Contributions algorithmiques au contrôle optimal stochastique à temps discret et horizon infini

Bruno Scherrer

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Contributions algorithmiques au contrôle optimal stochastique à temps discret et horizon infini

THÈSE

présentée et soutenue publiquement le 28 Juin 2016

pour l'obtention d'une

Habilitation de l'Université de Lorraine

(mention informatique)

par

Bruno Scherrer

Jury :	Didier Henrion (Président du jury)	DR CNRS, LAAS-CNRS
	Frédéric Bonnans	DR INRIA, Ecole Polytechnique
	Stéphane Gaubert	DR INRIA, Ecole Polytechnique
	Bruno Gaujal	DR INRIA, INRIA Grenoble Rhône-Alpes
	Anne Gegout-Petit	Professeur, Université de Lorraine
	Rémi Munos	DR INRIA & Google Deepmind
Rapporteurs :	Dimitri Bertsekas	Professeur, Massachusets Institute of Technology
	Stéphane Gaubert	DR INRIA, Ecole Polytechnique
	Bruno Gaujal	DR INRIA, INRIA Grenoble Rhône-Alpes

Institut Elie Cartan de Lorraine — UMR 7502

Mis en page avec la classe thesul.

Remerciements

Je remercie chaleureusement Dimitri Bertsekas, Stéphane Gaubert et Bruno Gaujal d'avoir accepté de rapporter ce manuscrit. Je remercie Didier Henrion d'avoir également accepté d'être rapporteur anonyme... Je remercie de plus Rémi Munos, Frédéric Bonnans, Anne Gegout-Petit et François Charpillet (même si ça n'a finalement pas été possible) d'avoir accepté de participer au jury.

S'il ne s'agit que d'une habilitation, c'est tout de même l'occasion idéale pour témoigner à ceux que j'ai pu côtoyer (de manière plus ou moins proche) ces dernières années ma reconnaissance pour tout ce qu'ils m'ont apporté.

Dimitri et Rémi, vos travaux ont été une source permanente d'inspiration, sans lesquels les recherches que je présente dans ce manuscrit n'auraient pas été possibles. Je vous suis reconnaissant de m'avoir laissé, ici et là, quelques pistes à creuser... Et je suis impatient de lire vos futures contributions.

Stéphane et Bruno, j'ai été très content, scientifiquement et humainement, de vous rencontrer ces dernières années. La dynamique des communautés scientifiques aurait dû nous faire nous rencontrer plus tôt ! J'ai principalement rédigé ce manuscrit pour vous raconter ce à quoi je m'occupais, et je suis heureux à l'idée que tout cela ait pu vous intéresser. Bruno (et Nicolas), merci pour votre accueil lors de ma dernière visite à Grenoble. Stéphane, je te remercie particulièrement de ne pas m'en vouloir d'avoir essayé (et parfois réussi) à te poser tant de lapins...

François, même si tu n'as pas pu te libérer pour la soutenance (pour l'une des meilleures raisons du monde), je te suis reconnaissant pour la confiance que tu m'as témoignée dès notre rencontre, et le soutien indéfectible que tu m'as apporté depuis. Je te dois beaucoup. Tu as été (et je suis sûr que tu es encore) un super chef d'équipe !

Matthieu, même si je ne suis pas toujours très réactif, tu auras été pour moi ces dernières années un super partenaire de discussions et de travail. Je suis impressionné par ton courage quand il s'agit de se lancer dans des calculs horribles et surtout par ton énorme créativité. Je suis vraiment très heureux de t'avoir rencontré et j'espère qu'on va continuer à bosser ensemble longtemps.

Je salue également les collègues (et ex-collègues) de la dream team de Lille avec qui j'ai eu la chance d'interagir (et parfois de travailler) : Mohammad, Bilal, Victor, Alessandro, Olivier, Julien, Philippe, Michal, Daniil. J'espère bientôt vous croiser.

Après ces nombreuses années au LORIA, où je ne compte plus les rencontres souvent devenues amitiés (le canal historique Cortex/Maia : Alain, Yann, Amine, Bernard, Laurent, Nico, Fred, Nazim, Vincent, Boris, Christophe, Jérémie, Maxime R, Olivier, Hervé, Joërg, Thomas G, Thomas V, Laure, Daniel, mais aussi David, Marc, Sylvain que je croise occasionnellement), j'ai été super bien accueilli à l'IECL. Samy, je te remercie de m'avoir ouvert la porte avec tant de bienveillance. Anne, Olivier et Antoine, je vous remercie de prolonger cette confiance, d'autant plus maintenant que vous connaissez mon côté "obstiné"... Je suis très heureux de faire aujourd'hui partie de l'équipe BIGS, de notre grande équipe de proba-stats, et du labo de maths. Merci aux copains qui sont alors devenus des collègues (Julien, Karim, Anne, Alex, Aline, Benoît, Kolehe, Olivier, Madalina, Jean-Seb, Vladimir, Oussama, Antoine, Bruno P, Jean-François S...) de m'avoir aidé (sans le savoir ?) dans cette évolution un peu étrange. Merci à tous les collègues que j'ai rencontrés depuis mon arrivée et qui contribuent également à faire du labo un lieu convivial : Romain, Nicolas, Sophie, Anne, Aurélie, Irène, Denis, Jean-Marie, Sandie, Tom, Renaud, Bruno D, Pierre-Emmanuel, Laurent, David, Frédéric, Damien, Xavier, Jean-François G, Jean-François W, Cécile, Wolfgang, Matei, Nicole, Aurélien, sans oublier mon actuel co-bureau Joseph avec qui nous menons parfois des expériences climatiques extrêmes... Merci à celles et ceux qui, parfois un peu (trop) dans l'ombre, rendent possible (et aussi souvent agréable) notre quotidien : Céline, Elodie, Paola, Didier, Nathalie, Laurence... Pardon à tous ceux que j'oublie.

Merci enfin à ceux qui, en dehors du boulot, font que nous sommes tant attachés à Nancy : Fabrice & Valérie, Julie & Samuel, Karim & Pascale, Anne & Alex, Olivier & Murièle, Claire & Olivier, Laurent & Christelle, Stéph & Bernard, Yann, Evelyne, Alain & Christine, Aline & Arsen, Amine & Armelle, Benoît & Marie, Isabelle & Hugo, Kolehe & Elsa, Madalina & Luis... avec une pensée tendre pour tous les nains qui nous entourent...

Je ne vois pas comment je pourrais terminer sans un mot pour vous : Régine, Marion et Arnaud, vous êtes tout ce qui compte.

Table des matières

Acronymes

MDP :	Processus de décision Markovien	Markov Decision Process
DP :	Programmation dynamique	Dynamic Programming
TD :	Différences temporelles	Temporal Differences
BR :	Résidu de Bellman	Bellman Residual
LSTD :	TD-moindres carrés	Least-Squares TD
LSBR :	BR-moindres carrés	Least-Squares BR
(A)VI :	Itérations sur les valeurs (approché)	(Approximate) Value Iteration
(A)PI :	Itérations sur les politiques (approché)	(Approximate) Policy Iteration
(A)MPI :	Itérations sur les politiques modifié (approché)	(Approximate) Modified Policy Iteration
CBMPI :	MPI basé sur un classifieur	Classification Based MPI
NSVI :	VI non-stationnaire	Non-stationary (A)VI
NSPI :	PI non-stationnaire	Non-stationary (A)PI
NSMPI :	MPI non-stationnaire	Non-stationary (A)MPI
CPI :	PI conservatif	Convervative PI
LSPI :	PI-moindres carrés	Least-Squares PI
PSDP :	Recherche de politique par DP	Policy Search by DP

Introduction

Ce document présente une sélection des activités de recherche que j'ai menées à l'*Institut National de Recherche en Informatique et Automatique (INRIA)* de Lorraine et au *Laboratoire Lorrain de Recherche en Informatique et ses Applications* à Nancy depuis mon doctorat en 2003, en collaboration avec Amine Boumaza, Olivier Buffet, Eugene Feinberg, Victor Gabillon, Bernard Girau, Matthieu Geist, Mohammad Ghavamzadeh, Jörg Hoffmann, Jefferson Huang, Alessandro Lazaric, Boris Lesner, Shie Mannor, Julien Perolat, Marek Petrik, Olivier Pietquin, Bilal Piot, Manel Tagorti, Christophe Thiéry et Cesar Torres-Huitzil.

Dans mes recherches, je considère un problème d'optimisation, le *contrôle optimal stochastique*, que l'on rencontre dans diverses disciplines (et sous diverses appellations !), notamment en *recherche opérationnelle*, en *automatique* (on parle alors plutôt de "commande optimale"), en *économie* ("théorie de la décision séquentielle"), et en *apprentissage automatique* ("apprentissage par renforcement"). Mes travaux se sont particulièrement porté sur la variation *à temps discret* de ce problème [1], dans lequel on considère l'évolution de l'état d'un système régie par une équation du type :

$$x_{t+1} = f(x_t, a_t, w_t) \qquad t = 0, 1, \dots .$$

Dans cette équation, t désigne le temps, x_t correspond à l'*état* du système, a_t est une *action* choisie (on parlera indifféremment de *contrôle*), w_t est un aléa, et f est une fonction qui caractérise l'évolution de l'état du système. A chaque instant t, le choix de l'action a_t dans l'état x_t donne lieu, en plus d'une transition vers l'état x_{t+1}, à une *récompense* (ou à l'opposé à un *coût*)

$$r(x_t, a_t, x_{t+1}) \in \mathbb{R},$$

qui mesure la qualité *instantanée* de la transition (x_t, a_t, x_{t+1}). Le problème du contrôle optimal stochastique consiste à déterminer une séquence d'actions $(a_0, a_1, \dots)$ (telle que a_t ne dépend que du passé $x_0, a_0, \dots, x_t$) de sorte à maximiser le cumul des récompenses (ou à minimiser le cumul des coûts), notamment en gérant optimalement le compromis entre court et long termes. Formellement, le modèle de *processus de décision Markovien*, que je décris plus précisément ci-après, a l'élégance de saisir l'essence de ce problème tout en donnant lieu à des méthodes de résolution numérique efficaces.

Nous considérons que le système dynamique vit dans un *espace d'états* X fini ou dénombrable.

1. Une exception est le travail réalisé avec Amine Boumaza, sur le lien entre navigation par potentiels harmoniques et le contrôle optimal (Boumaza et Scherrer, 2007), et sur un modèle jouet de fourmis (Boumaza et Scherrer, 2008).

Dans chaque état $x \in X$, l'action est choisie dans un *espace d'actions* A de taille finie [2]. L'action $a \in A$ dans un état x caractérise la *probabilité de transition* vers un nouvel état x' :

$$\mathbb{P}(x_{t+1} = x' | x_t = x, a_t = a),$$

autrement dit la dynamique est Markovienne (elle ne dépend pas de ce qui s'est passé avant l'instant t) et contrôlée (elle dépend de l'action choisie a). Chaque transition (x, a, x') donne lieu à une récompense $r(x, a, x')$, où $r : X \times A \times X \to \mathbb{R}$ est la *fonction de récompense* instantanée.

Dans ce contexte, nous cherchons une politique déterministe et stationnaire (une fonction $\pi : X \to A$ qui associe une action à chaque état [3]) qui maximise l'espérance de la somme actualisée des récompenses à partir de tout état x, quantité appelée la *valeur de la politique* π en l'état x :

$$v_\pi(x) := \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r(x_k, a_k, x_{k+1}) \middle| x_0 = x,\ \forall k \geq 0,\ a_k = \pi(x_k),\ x_{k+1} \sim \mathbb{P}(\cdot | x_k, a_k)\right]$$

où $\gamma \in [0, 1]$ peut être à la fois interprété comme un *taux d'actualisation* ou comme une probabilité de survie à chaque instant. Le modèle ci-dessus est appelé *processus de décision Markovien (MDP)* (Puterman, 1994; Bertsekas et Tsitsiklis, 1996), et le problème que nous avons décrit est le *contrôle optimal stochastique actualisé à temps discret et horizon infini.*

La *valeur optimale* à partir de l'état x est naturellement définie comme suit :

$$v_*(x) := \max_\pi v_\pi(x).$$

Pour toute politique π, on notera P_π le noyau de transition Markovien sur X si on choisit en tout état x l'action $\pi(x)$. Etant donné une numérotation des états, on peut représenter ce noyau comme une matrice stochastique ; pour tous états x et x', la valeur en la ligne correspondant à x et la colonne correspondant à x' est :

$$[P_\pi]_{x,x'} = \mathbb{P}(x_{t+1} = x' | x_t = x, a_t = \pi(x)).$$

Les fonctions valeur v_π et v_* peuvent également être vues comme des vecteurs de $\mathbb{R}^X$. Il est connu que la fonction valeur v_π d'une politique π est la solution de l'équation de Bellman linéaire suivante :

$$v_\pi = r + \gamma P_\pi v_\pi,$$

qui est un système linéaire ayant autant de lignes et d'inconnues qu'il y a d'états. De manière équivalente, on peut dire de la fonction v_π qu'elle est le point fixe de l'opérateur affine

$$T_\pi : v \mapsto r + \gamma P_\pi v.$$

Il est également connu que la fonction valeur optimale v_* satisfait l'équation de Bellman non linéaire suivante

$$v_* = \max_\pi (r + \gamma P_\pi v_*) = \max_\pi T_\pi v_*$$

2. Les hypothèses de finitude et de dénombrabilité, que j'ai généralement faites dans mes travaux, sont essentiellement énoncées pour simplifier l'écriture—cela permet de représenter un certain nombre d'objets d'intérêt sour la forme de vecteurs et de matrices (potentiellement de taille infinie)—et éviter des subtilités techniques d'intégration. L'extension d'un certain nombre de résultats à des situations plus générales seront souvent triviales dès lors que les opérateurs de Bellman qui nous introduisons plus loin sont bien définis et contractants.

3. Se restreindre aux politiques déterministes et stationnaires n'est pas une limitation, dans le sens où, pour le critère d'optimalité que nous allons considérer, on peut montrer qu'il existe au moins une politique déterministe et stationnaire qui est optimale.

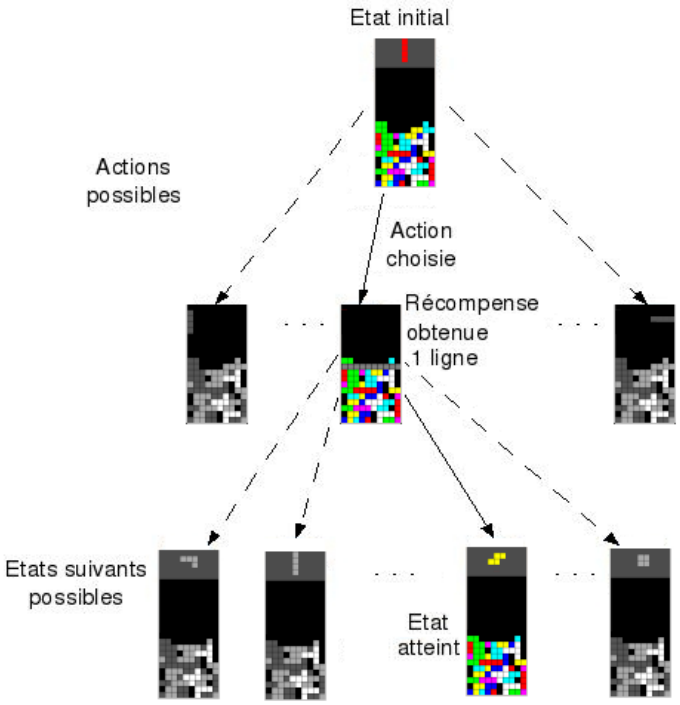


FIGURE 0.1 – Modélisation du jeu de Tetris par un MDP.

où l'opérateur max appliqué à un vecteur l'est composante par composante. En d'autres termes, v_* est le point fixe de l'opérateur non linéaire

$$T : v \mapsto \max_{\pi} T_\pi v.$$

Lorsque $\gamma < 1$, les opérateurs T_π et T sont γ-contractants pour la norme infinie $\|u\|_\infty = \max_{x \in X} |u(x)|$, ce qui assure (par le théorème de Banach) que les points fixes existent et sont uniques.

Finalement, pour toute fonction $v : X \to \mathbb{R}$, on dit d'une politique π qu'elle est *gloutonne par rapport à* v si elle satisfait

$$\pi \in \arg\max_{\pi'} T_{\pi'} v$$

ou de manière équivalente, $T_\pi v = Tv$. On note $\mathcal{G}(v)$ l'ensemble des politiques gloutonnes par rapport à v. Algorithmiquement, la détermination d'une politique gloutonne peut se faire en un temps quadratique en le nombre d'états et linéaire en le nombre d'actions. Les notions de *fonction valeur optimale* et de *politique gloutonne* sont fondamentales pour le contrôle optimal : en effet, n'importe quelle politique π_* qui est gloutonne par rapport à la fonction valeur optimale v_* est une politique optimale et sa valeur v_{π_*} est égale à v_*. Ainsi, le problème essentiel du contrôle optimal se réduit au calcul du point fixe de l'opérateur non-linéaire T.

Exemple 0.1. *Tetris est un jeu vidéo populaire créé en 1985 par Alexey Pajitnov. Le jeu se joue sur une grille de taille* 10×20. *Des pièces de différentes formes tombent. Le joueur doit choisir où poser chaque pièce : il peut la déplacer horizontalement ou la faire pivoter jusqu'à ce qu'elle se pose sur le mur existant. Chaque fois qu'une ligne horizontale est complétée, celle-ci disparaît et les pièces qui la surplombent descendent. Le but est de supprimer le plus grand*

nombre de lignes avant qu'il ne soit plus possible de poser aucune pièce sans sortir de la zone de jeu.

Au lieu de reproduire exactement le jeu original, Bertsekas et Ioffe (1996) ont proposé de se concentrer sur le problème de décision principal : où et dans quelle orientation poser la pièce courante. La figure 0.1 illustre le modèle de ce jeu dans le cadre des MDPs. L'état correspond à la configuration du mur et à la pièce courante à poser. Les actions sont l'ensemble des translations/rotations qu'on peut appliquer à cette pièce (il y en a au plus 10×4 *par état). La récompense est le nombre de lignes supprimées après avoir posé la pièce. Comme on s'intéresse ici à la maximisation du nombre de lignes, le facteur d'actualisation naturel est*[4] $\gamma = 1$. *On obtient un nouvel état en adjoignant au mur ainsi obtenu une nouvelle pièce tirée uniformément au hasard parmi les 7 pièces du jeu.*

De manière un peu plus détaillée, chaque transition d'états du jeu de Tetris est composée de 2 étapes : une transition déterministe du mur liée à l'endroit où on décide de poser la pièce, et une transition aléatoire non contrôlée lorsqu'une nouvelle pièce apparaît. On montre alors qu'on peut se restreindre au calcul des fonctions valeur sur l'ensemble des murs possibles (et non des états), et que l'équation d'optimalité de Bellman a la forme suivante :

$$\forall s \in S, \quad v_*(s) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \max_{a \in A(p)} r(s,p,a) + v_*(succ(s,p,a)),$$

où S *est l'ensemble de tous les murs possibles,* $\mathcal{P}$ *l'ensemble des pièces,* $A(p)$ *l'ensemble des translations/rotations possibles pour la pièce* p, $r(s,p,a)$ *et* $succ(s,p,a)$ *étant respectivement le nombre de lignes supprimées et le mur obtenu (déterministiquement) lorsqu'on pose la pièce* p *sur le mur* s *dans la configuration* a. *L'unique fonction qui satisfait cette équation donne, pour chaque mur possible* s, *le meilleur score moyen qui peut être atteint à partir de* s. *Pour tout état* (s,p), *on en déduit aisément la valeur optimale*

$$\max_a r(s,p,a) + v_*(succ(s,p,a)),$$

et une décision optimale $\pi_*(s,p)$ *:*

$$\pi_*(s,p) \in \operatorname*{argmax}_a r(s,p,a) + v_*(succ(s,p,a)).$$

Si les trois opérateurs T_π, T et $\mathcal{G}$, que nous venons d'introduire permettent de caractériser la solution au problème du contrôle optimal, ils sont aussi d'une grande utilité pour décrire des schémas algorithmiques. L'algorithme le plus simple conceptuellement est *Itérations sur les Valeurs (VI)*, qui approche le point fixe de T en calculant, à partir d'une initialisation quelconque v_0, les itérés :

$$v_{k+1} \leftarrow T v_k \qquad k = 0, 1, \ldots$$

4. D'un point de vue technique, on récupère les propriétés de contraction pour les opérateurs de Bellman via le fait que quelle soit la stratégie employée, la partie termine en un temps fini avec probabilité 1 (Burgiel, 1997). L'argument principal est suffisamment simple pour être énoncé ici : Burgiel (1997) montre qu'une séquence finie (relativement longue) de pièces fait nécessairement perdre le joueur ; comme dans la parabole du singe savant qui finira par écrire l'œuvre de Shakespeare, cette séquence de pièces apparaîtra avec probabilité 1 en un temps fini.

La propriété de contraction assure que cet algorithme converge asymptotiquement vers la valeur optimale v_*. Un autre algorithme standard, d'une nature sensiblement différente, est *Itérations sur les Politiques (PI)*, qui calcule à chaque étape k la valeur de la politique courante π_k, et en déduit une politique π_{k+1} meilleure uniformément, c'est-à-dire telle que $v_{\pi_{k+1}} \geq v_{\pi_k}$ (avec inégalité stricte en au moins un état tant que la politique politique optimale n'est pas atteinte) :

$$\begin{aligned} v_k \text{ est tel que } v_k &= T_{\pi_k} v_k \\ \pi_{k+1} &\leftarrow \text{un élément de } \mathcal{G}(v_k). \end{aligned}$$

On déduit facilement de la monotonie de la suite $(v_k)_k$ que dans le cas de MDP dont les espaces d'états et d'actions sont finis (le nombre de politiques est donc aussi fini), ce deuxième algorithme trouve la politique en un nombre fini d'itérations. Ainsi, cet algorithme est en général plus économe que VI en nombre d'itérations, mais chaque itération est un peu plus lourde, la détermination de v_k impliquant la résolution d'un système linéaire. Ces deux algorithmes standards, d'approches apparemment assez différentes (l'un itère sur les valeurs, l'autre sur les politiques) peuvent être unifiés en considérant l'algorithme *Itérations sur les Politiques Modifié (MPI)* introduit par Puterman et Shin (1978) :

$$\begin{aligned} \pi_{k+1} &\leftarrow \text{un élément de } \mathcal{G}(v_k) \\ v_{k+1} &\leftarrow (T_{\pi_{k+1}})^m v_k, \end{aligned} \tag{1}$$

où m est un paramètre entier strictement positif. Lorsque m vaut 1, il est facile de voir (en se rappelant que $\pi_{k+1} \in \mathcal{G}(v_k)$ est équivalent à $T_{\pi_{k+1}} v_k = T v_k$) que l'on retombe sur l'algorithme VI. De même, lorsque $m = \infty$, on retrouve l'algorithme PI dans la mesure où itérer infiniment avec l'opérateur $T_{\pi_{k+1}}$ contractant revient à estimer son unique point fixe. Le paramètre m permet ainsi de passer progressivement d'un algorithme à l'autre ; en pratique, une valeur bien choisie de m permet de gagner sur tous les fronts (Puterman et Shin, 1978) : on obtient une convergence plus rapide (comme PI) tout en gardant une complexité par itérations relativement faible (proche de celle de VI). D'autres unifications existent dans la littérature. Dans le cadre de la thèse de Christophe Thiéry, nous nous sommes intéressés à l'algorithme λ-*Itérations sur les Politiques* (Bertsekas et Ioffe, 1996; Bertsekas et Tsitsiklis, 1996) dont l'itération k peut être décrite de la manière suivante :

$$\begin{aligned} \pi_{k+1} &\leftarrow \text{un élément de } \mathcal{G}(v_k) \\ v_{k+1} &\leftarrow (1-\lambda) \sum_{i=0}^{\infty} \lambda^i (T_{\pi_{k+1}})^{i+1} v_k. \end{aligned} \tag{2}$$

Cet algorithme, paramétré par le réel $\lambda \in [0, 1]$, peut être vu comme effectuant une moyenne géométrique de paramètre λ de toutes les mises à jours possibles de MPI. Lorsque $\lambda = 0$, on retrouve VI ; lorsque $\lambda = 1$, on retombe sur PI. La somme infinie peut apparaître comme un obstacle à une implémentation pratique ; ce n'est en fait pas le cas, dans la mesure où l'on peut montrer que cette somme peut être de manière équivalente décrite comme le point fixe d'un opérateur linéaire similaire à $T_{\pi_{k+1}}$—je renvoie le lecteur au texte de Bertsekas et Ioffe (1996) pour plus de détails. Finalement, un dernier algorithme qui, lui, ne semble pas facilement implémentable, mais a le bon goût de généraliser l'ensemble des algorithmes mentionnés jusqu'ici, est Itérations sur les Politiques Optimiste [5] (Thiéry et Scherrer, 2010; Scherrer et Thiéry, 2010) :

$$\pi_{k+1} \leftarrow \text{un élément de } \mathcal{G}(v_k)$$

5. Le choix du mot "optimiste" vient de (Bertsekas et Tsitsiklis, 1996), qui parle d'optimisme dès lors qu'une variation de PI passe à une nouvelle politique avant d'avoir complètement fini d'évaluer la politique courante.

$$v_{k+1} \leftarrow \sum_{i=0}^{\infty} \lambda_i (T_{\pi_{k+1}})^{i+1} v_k,$$

où les $(\lambda_i)_{i \geq 0}$ sont une famille de coefficients positifs on nuls et dont la somme est 1.

Une fois effectué ce petit tour d'horizon des algorithmes de programmation dynamique pour le contrôle optimal, la sélection des activités de recherche que j'ai choisi de présenter [6] peut être décrite comme s'intéressant à l'analyse de ces algorithmes et d'un certain nombre de leurs variations. Par exemple, je me suis intéressé à la complexité, c'est-à-dire à l'étude du nombre d'itérations en fonction des paramètres du problème (nombre d'états, d'actions, facteur d'actualisation) : au premier chapitre, je décris quelques contributions sur ce thème.

Une autre partie—plus volumineuse—de mon activité de recherche, décrite dans les chapitres qui suivent, s'est portée sur la résolution des problèmes de grande taille, pour lesquels il n'est pas envisageable d'implémenter les schémas itératifs ci-dessus. Une option assez naturelle historiquement très bien documentée par Bertsekas et Tsitsiklis (1996) est alors de considérer des variations approchées de tous ces algorithmes. Les schémas d'approximation étant nombreux, l'approche consiste à la fois à faire des études générales de l'influence d'un bruit d'approximation sur les schémas itératifs décrits ci-dessus (l'idéal étant de le faire sur le dernier schéma qui généralise les précédents), et aussi à instancier des cas précis d'implémentations ; dans tous les cas, il s'agit de bien comprendre ce qui entre en jeu dans les approximations, notamment en dérivant des garanties théoriques de performance. Ceci nous donne souvent des informations précieuses quant au rôle des différents paramètres d'approximation, que l'on peut idéalement confirmer empiriquement. Ainsi, dans les chapitres 2 et 3, je m'intéresse au problème de l'estimation approchée de la valeur d'une politique, ce qui peut intervenir naturellement dans un schéma d'itérations sur les Politiques approché. Dans le chapitre qui suit (chapitre 4), je présente des résultats généraux de sensibilité au bruit du schéma algorithmique MPI qui, nous l'avons vu, généralise les algorithmes standards VI et PI. J'y décris notamment une implémentation particulière qui se base sur une série de problèmes de régression et de classification, pour lesquels nous avons dérivé des résultats analytiques intéressants, et qui a donné de très bons résultats empiriques sur le jeu de Tetris décrit plus haut. Enfin, dans les deux chapitres qui suivent, j'approfondis l'analyse sur les garanties de performance du chapitre 4. En effet, dans le chapitre 5, je montre comment l'utilisation de politiques non-stationnaires permet d'améliorer la dépendance des garanties vis-à-vis du facteur d'actualisation γ. Puis, dans l'ultime chapitre 6, j'effectue une comparaison d'un ensemble d'algorithmes de type PI, et montre analytiquement que certains jouissent de garanties avec de meilleures constantes dites de concentrabilité, et des simulations numériques montrent qu'ils sont effectivement préférables en pratique.

Les principales publications concernées par chacun des chapitres, sont listées ci-dessous.

— Chapitre 1 :
B. Scherrer. *Improved and Generalized Upper Bounds on the Complexity of Policy Iteration.* **Mathematics of Operations Research**, 2016.

— Chapitre 2 :
B. Scherrer. *Should one compute the temporal difference fix point or minimize the bellman residual ? the unified oblique projection view.* Dans **International Conference on Machine Learning**, 2010.

— Chapitre 3 :
M. Tagorti et B. Scherrer. *On the Rate of Convergence and Error Bounds for LSTD(λ).* Dans **International Conference on Machine Learning**, Lille, France, juillet 2015.
M. Geist et B. Scherrer *Off-policy Learning with Eligibility Traces : A Survey.* Dans

6. Plutôt que de présenter une liste exhaustive de mes activités, j'ai ainsi choisi de présenter dans ce document un ensemble de travaux qui—j'espère que le lecteur en conviendra—forme un tout cohérent.

Journal of Machine Learning Research, 2014.

— Chapitre 4 :
B. Scherrer, M. Ghavamzadeh, V. Gabillon, B. Lesner et M. Geist. *Approximate Modified Policy Iteration and its Application to the Game of Tetris.* **Journal of Machine Learning Research**, 2015.

— Chapitre 5 :
B. Scherrer et B. Lesner. *On the use of non-stationary policies for stationary infinite-horizon Markov decision processes.* Dans **Neural Information Processing Systems**, South Lake Tahoe, United States, décembre 2012.
B. Lesner et B. Scherrer. *Non-Stationary Approximate Modified Policy Iteration.* Dans **International Conference on Machine Learning**, Lille, France, juillet 2015.

— Chapitre 6 :
B. Scherrer. *Approximate Policy Iteration Schemes : A Comparison.* Dans **International Conference on Machine Learning**. Pékin, Chine, juin 2014.

Chapitre 1

Quelques majorants de la complexité d'Itérations sur les Politiques

Nous considérons dans ce chapitre le cas d'un MDP fini, ayant n états—on considèrera ici que l'espace d'états X est $\{1, 2, \ldots, n\}$ et que les fonctions valeur sont identifiables à des vecteurs de $\mathbb{R}^n$—et m actions par état. Nous allons décrire des majorants du nombre d'itérations de deux algorithmes de type Itérations sur les Politiques (PI), l'algorithme standard mentionné dans le chapitre introductif précédent, ainsi qu'une variation qui change la politique seulement en un seul état bien choisi (et pour lequel nous serons capables de dériver des résultats spécifiques). Ce travail a été publié en version courte (Scherrer, 2013a) puis étendue (Scherrer, 2016). Nous commençons par décrire précisément ces deux algorithmes.

1.1 Algorithmes

Soit π une politique. On appelle **avantage par rapport à la politique** π le vecteur :

$$a_\pi = \max_{\pi'} T_{\pi'} v_\pi - v_\pi = T v_\pi - v_\pi.$$

Par définition des opérateurs et de la fonction de valeur v_π, on peut voir que

$$a_\pi = \max_{\pi'} T_{\pi'} v_\pi - T_\pi v_\pi,$$

c'est-à-dire que les composantes de ce vecteur sont nécessairement positives ou nulles ; elles sont toutes nulles si et seulement si la politique est optimale. On appelle **ensemble des états modifiables** (en anglais, *switchable*) l'ensemble des états pour lequel l'avantage est strictement positif :

$$S_\pi = \{i,\ a_\pi(i) > 0\}.$$

Supposons maintenant que π n'est pas optimale ; cela implique que S_π est un ensemble non vide. Pour n'importe quel sous-ensemble Y de S_π, notons $\text{switch}(\pi, Y)$ une politique obtenue en remplaçant les actions de π par des actions gloutonnes par rapport à v_π dans les états contenus dans Y :

$$\forall i,\ \text{switch}(\pi, Y)(i) = \begin{cases} \text{un élément de } \arg\max_a [T_a v_\pi](i) & \text{si } i \in Y \\ \pi(i) & \text{si } i \notin Y. \end{cases}$$

Comme le montre le résultat suivant standard, n'importe quelle politique obtenue via cet opérateur sera meilleure que la politique π.

Lemme 1.1 (voir par exemple Puterman (1994))**.** *Soit π une politique non optimale. Si $\pi' = switch(\pi, Y)$ pour un sous-ensemble non vide Y de S_π, alors $v_{\pi'} \geq v_\pi$ et il existe au moins un état i tel que $v_{\pi'}(i) > v_\pi(i)$.*

Ce lemme constitue le fondement du schéma *Itérations sur les Politiques* (PI), qui génère une séquence de politiques (π_k) comme suit.

$$\pi_{k+1} \leftarrow \text{switch}(\pi_k, Y_k) \text{ pour un } Y_k \text{ tel que } \emptyset \subsetneq Y_k \subseteq S_{\pi_k}.$$

Selon la façon dont on choisit l'ensemble Y_k, on obtient différentes variations de PI. Nous considérons les deux variantes suivantes :

— Quand pour tout k, $Y_k = S_{\pi_k}$, c'est-à-dire qu'on change les actions dans tous les états où l'avantage est strictement positif, l'algorithme est connu sous le nom de PI de Howard (PI-Howard dans la suite) ; on peut notamment remarquer dans ce cas qu'à chaque étape, la politique est gloutonne par rapport à la valeur de la politique précédente, soit formellement $\pi_{k+1} \in \mathcal{G}(v_{\pi_k})$. C'est l'algorithme PI tel que nous l'avons décrit dans l'introduction.

— Quand pour tout k, Y_k est un singleton qui contient l'état $i_k \in \arg\max_i a_{\pi_k}(i)$, c'est-à-dire qu'on change une seule action dans l'état qui a un avantage maximal par rapport à π_k, on appellera cet algorithme PI-Simplexe [1].

Comme ces algorithmes génèrent une séquence de politiques dont les valeurs forment une suite strictement croissante (par le lemme 1.1), toute variation de PI converge vers la politique et la valeur optimales en un nombre d'itérations qui est plus petit que m^n, le nombre total de politiques du problème. En pratique, les algorithmes de type PI convergent en très peu d'itérations. Sur des MDP aléatoires, la convergence a lieu en un temps qui est souvent sous-linéaire en n. Nous allons exhiber des majorants du nombre d'itérations requis par PI-Howard et PI-Simplexe qui sont beaucoup plus fins que m^n.

1.2 Majorants de la complexité pour un γ fixé

Dans cette section, nous commençons par décrire quelques résultats de la littérature—voir (Ye, 2011) pour un état de l'art plus étoffé—sur le nombre d'itérations requis par PI-Howard et PI-Simplexe, ainsi qu'un certain nombre d'améliorations et d'extensions originales. Nous renvoyons le lecteur à (Scherrer, 2016) pour les preuves de la plupart résultats énoncés.

Une observation importante concernant les deux algorithmes, qui sera centrale dans les résultats que nous allons décrire, est que la séquence qu'ils génèrent satisfait une certaine propriété de contraction [2]. Pour tout vecteur u de $\mathbb{R}^n$, rappelons que $\|u\|_\infty = max_{1 \leq i \leq n} |u(i)|$ désigne la norme infinie de u. Soit $\mathbb{1}$ le vecteur dont toutes les composantes sont égales à 1.

Lemme 1.2. *La séquence $(\|v_* - v_{\pi_k}\|_\infty)_{k \geq 0}$ générée par PI-Howard est contractante de coefficient γ.*

Lemme 1.3. *La séquence $(\mathbb{1}^T(v_* - v_{\pi_k}))_{k \geq 0}$ générée par PI-Simplexe est contractante de coefficient $1 - \frac{1-\gamma}{n}$.*

1. L'appellation PI-Simplexe est motivée par le fait que la variation de PI est alors équivalente à l'algorithme du simplexe (utilisant la règle du pivot le plus grand) appliqué à une formulation de type programmation linéaire du problème (Ye, 2011).

2. Une séquence $(x_k)_{k \geq 0}$ de réels positifs est dite contractante de coefficient $\alpha < 1$ si et seulement si pour tout $k \geq 0$, $x_{k+1} \leq \alpha x_k$.

Si cette observation est bien connue pour PI-Howard, elle n'a à notre connaissance jamais été mentionnée explicitement dans la littérature pour PI-Simplexe. Ces propriétés de contractions ont pour immédiate conséquence le résultat suivant [3].

Corollaire 1.4. *Soit* $V_{\max} = \frac{\max_\pi \|r_\pi\|_\infty}{1-\gamma}$ *un majorant de* $\|v_\pi\|_\infty$ *pour toute politique* π. *Afin d'obtenir une politique* ϵ*-optimale, c'est-à-dire une politique* π_k *satisfaisant* $\|v_* - v_{\pi_k}\|_\infty \leq \epsilon$, *PI-Howard a besoin d'au plus* $\left\lceil \frac{\log \frac{V_{\max}}{\epsilon}}{1-\gamma} \right\rceil$ *itérations, tandis que PI-Simplexe a besoin d'au plus* $\left\lceil \frac{n \log \frac{nV_{\max}}{\epsilon}}{1-\gamma} \right\rceil$ *itérations.*

Comme ces majorants dépendent d'un terme de précision ϵ, cela implique que PI-Howard et PI-Simplexe sont des algorithmes *faiblement polynomiaux* pour un facteur d'actualisation γ fixé. Un résultat particulièrement important a récemment été obtenu par Ye (2011), qui a fourni des majorants qui ne dépendent plus de la précision ϵ, ce qui implique que PI-Howard et PI-Simplexe sont *fortement polynomiaux* pour un facteur d'actualisation γ fixé.

Théorème 1.5 (Ye (2011))**.** *PI-Simplexe et PI-Howard terminent après au plus* $n(m-1)\left\lceil \frac{n}{1-\gamma} \log\left(\frac{n^2}{1-\gamma}\right)\right\rceil$ *itérations.*

La démonstration se base sur le fait que ces algorithmes sont des instances particulières de l'algorithme du simplexe appliqué à une formulation de type programme linéaire du problème. En utilisant des arguments plus directs, Hansen *et al.* (2013) ont récemment amélioré le majorant d'un facteur $O(n)$ pour PI-Howard.

Théorème 1.6 (Hansen *et al.* (2013))**.** *PI-Howard termine après au plus* $(nm+1)\left\lceil \frac{1}{1-\gamma} \log\left(\frac{n}{1-\gamma}\right)\right\rceil$ *itérations.*

Nos deux premiers résultats, publiés dans (Scherrer, 2016), qui sont des conséquences des propriétés de contractions décrites plus haut (lemmes 1.2 et 1.3) sont énoncés dans les théorèmes ci-dessous.

Théorème 1.7. *PI-Howard termine après au plus* $n(m-1)\left\lceil \frac{1}{1-\gamma} \log\left(\frac{1}{1-\gamma}\right)\right\rceil$ *itérations.*

Théorème 1.8. *PI-Simplexe termine après au plus* $n(m-1)\left\lceil \frac{n}{1-\gamma} \log\left(\frac{n}{1-\gamma}\right)\right\rceil$ *itérations.*

Nous allons détailler la preuve du théorème 1.7 pour plusieurs raisons : tout d'abord, cela permet de se rendre compte de l'élégance de l'argumentation proposée par Ye (2011) (notre résultat est légèrement plus fin, mais l'essentiel est très proche de la preuve originelle) ; ensuite, la preuve du théorème 1.8 pour PI-Simplexe est similaire ; enfin, la preuve donnée ci-dessous est particulièrement simple, et elle aurait à mon avis sa place parmi les résultats standards d'analyse d'Itérations sur les Politiques.

3. Pour PI-Howard, nous avons : $\|v_* - v_{\pi_k}\|_\infty \leq \gamma^k \|v_* - v_{\pi_0}\|_\infty \leq \gamma^k V_{\max}$. Ainsi, une condition suffisante pour avoir $\|v_* - v_{\pi_k}\|_\infty < \epsilon$ est $\gamma^k V_{\max} < \epsilon$, ce qui est vrai dès lors que $k \geq \frac{\log \frac{V_{\max}}{\epsilon}}{1-\gamma} > \frac{\log \frac{V_{\max}}{\epsilon}}{\log \frac{1}{\gamma}}$. Pour PI-Simplexe, nous avons $\|v_* - v_{\pi_k}\|_\infty \leq \mathbb{1}^T(v_* - v_{\pi_k}) \leq \gamma^k \mathbb{1}^T(v_* - v_{\pi_0}) \leq \gamma^k n V_{\max}$, et on conclue de manière similaire à PI-Howard.

Preuve du théorème 1.7. On commence par dériver l'identité suivante : pour toute paire de politiques π et π',

$$\begin{aligned} v_{\pi'} - v_\pi &= (I - \gamma P_{\pi'})^{-1} r_{\pi'} - v_\pi && \{v_\pi = T_\pi v_\pi \;\Rightarrow\; v_\pi = (I - \gamma P_\pi)^{-1} r_\pi\} \\ &= (I - \gamma P_{\pi'})^{-1}(r_{\pi'} + \gamma P_{\pi'} v_\pi - v_\pi) \\ &= (I - \gamma P_{\pi'})^{-1}(T_{\pi'} v_\pi - v_\pi). && (1.1) \end{aligned}$$

Alors, pour tout k, on a :

$$\begin{aligned} v_* - T_{\pi_k} v_* &= (I - \gamma P_{\pi_k})(v_* - v_{\pi_k}) && \{\text{Equation (1.1)}\} \\ &\leq v_* - v_{\pi_k}. && \{v_* - v_{\pi_k} \geq 0 \text{ et } P_{\pi_k} \geq 0\} \end{aligned}$$

Comme $v_* - T_{\pi_k} v_*$ est positif, on peut prendre la norme infinie et obtenir :

$$\begin{aligned} \|v_* - T_{\pi_k} v_*\|_\infty &\leq \|v_* - v_{\pi_k}\|_\infty \\ &\leq \gamma^k \|v_{\pi_*} - v_{\pi_0}\|_\infty && \{\text{Lemme 1.2}\} \\ &= \gamma^k \|(I - \gamma P_{\pi_0})^{-1}(v_* - T_{\pi_0} v_*)\|_\infty && \{\text{Equation (1.1)}\} \\ &\leq \frac{\gamma^k}{1-\gamma} \|v_* - T_{\pi_0} v_*\|_\infty. && \{\|(I - \gamma P_{\pi_0})^{-1}\|_\infty = \frac{1}{1-\gamma}\} \end{aligned}$$

Par définition de la norme infinie, il existe un état s_0 tel que $v_*(s_0) - [T_{\pi_0} v_*](s_0) = \|v_* - T_{\pi_0} v_*\|_\infty$. On en déduit que pour tout k,

$$\begin{aligned} v_*(s_0) - [T_{\pi_k} v_*](s_0) &\leq \|v_* - T_{\pi_k} v_*\|_\infty \\ &\leq \frac{\gamma^k}{1-\gamma} \|v_* - T_{\pi_0} v_*\|_\infty \\ &= \frac{\gamma^k}{1-\gamma} (v_*(s_0) - [T_{\pi_0} v_*](s_0)) \end{aligned}$$

Par conséquent, l'action $\pi_k(s_0)$ doit être différente de $\pi_0(s_0)$ quand $\frac{\gamma^k}{1-\gamma} < 1$, c'est-à-dire dès lors que k satisfait :

$$k \geq k^* = \left\lceil \frac{\log \frac{1}{1-\gamma}}{1-\gamma} \right\rceil > \left\lceil \frac{\log \frac{1}{1-\gamma}}{\log \frac{1}{\gamma}} \right\rceil.$$

En d'autres termes, si une politique n'est pas optimale, alors l'une de ses actions non-optimales est éliminée définitivement après au plus k^* itérations. En répétant cette argument pour toutes les actions non-optimales (qui sont au plus au nombre de $n(m-1)$), on obtient le résultat. □

Notre résultat pour PI-Howard est un facteur $O(\log n)$ plus fin que celui de Hansen *et al.* (2013), qui était à notre connaissance le meilleur résultat publié dans la littérature. Notre résultat pour PI-Simplexe est très légèrement meilleur (d'un facteur 2) que celui de Ye (2011), et utilise des arguments plus directs. En utilisant une preuve un peu plus complexe, on peut améliorer le résultat pour PI-Simplexe d'un facteur $O(\log n)$:

Théorème 1.9. *PI-Simplexe termine après au plus* $n^2(m-1)\left(1 + \frac{2}{1-\gamma} \log\left(\frac{1}{1-\gamma}\right)\right)$ *itérations.*

Par rapport à PI-Howard, le nombre d'itérations requis par PI-Simplexe est ainsi un facteur $O(n)$ plus grand. Cependant, chaque itération de PI-Simplexe (qui change seulement une action) a en général une complexité moindre qu'une itération de PI-Howard : en effet, la mise à jour peut-être réalisée en temps $O(n^2)$ à l'aide de la formule de Sherman-Morrisson, alors qu'une itération

de PI-Howard, qui nécessite de calculer la valeur d'une politique qui peut être arbitrairement différente de la politique précédente, demande en général un temps $O(n^3)$. Globalement, on pourra retenir que les deux algorithmes ont une complexité similaire.

Il est facile de voir que la dépendance linéaire en n du majorant pour PI-Howard est optimale. Nous conjecturons que la dépendance linéaire en m pour les deux bornes est également optimale. Il est peut-être possible d'améliorer la dépendance vis-à-vis du terme $\frac{1}{1-\gamma}$, mais la supprimer est impossible pour PI-Howard et peu vraisemblable pour PI-Simplexe. Fearnley (2010) a décrit un MDP pour lequel PI-Howard met un temps exponentiel en n pour $\gamma = 1$ et Hollanders *et al.* (2012) ont ensuite expliqué que cette complexité était la même lorsque γ est dans le voisinage de 1. Bien que des résultats similaires pour PI-Simplexe ne sont à notre connaissance pas documentés dans la littérature, Melekopoglou et Condon (1994) ont considéré quatre variations de PI qui changent seulement une action par itération, et exhibent des MDP sur lesquels ces algorithmes terminent en un temps exponentiel en n lorsque $\gamma = 1$.

1.3 Majorants de la complexité indépendants de γ

Dans la suite, nous décrivons des majorants qui ne dépendent pas du facteur d'actualisation γ, mais qui seront obtenus sous des hypothèses structurelles du MDP. Sur ce sujet, Post et Ye (2012) ont récemment dérivé une borne polynomiale pour les MDP déterministes.

Théorème 1.10 (Post et Ye (2012))**.** *Si le MDP est déterministe, alors PI-Simplexe termine après au plus $O(n^5 m^2 \log^2 n)$ itérations.*

Étant donnée une politique π d'un MDP déterministe, les états appartiennent soit à un cycle, soit à un chemin induit par π. Le cœur de l'analyse repose sur les lemmes que nous énonçons ci-après, qui montrent qu'au cours des itérations, des cycles sont créés régulièrement et qu'à chaque fois qu'un nouveau cycle apparaît, un progrès significatif vers la solution est effectué ; en conséquence, un progrès significatif a lieu régulièrement.

Lemme 1.11 (Post et Ye (2012))**.** *Supposons que le MDP est déterministe. Après au plus $O(n^2 m \log n)$ itérations, soit PI-Simplexe termine soit un nouveau cycle apparaît.*

Lemme 1.12 (Post et Ye (2012))**.** *Supposons que le MDP est déterministe. Lorsque PI-Simplexe passe de π à π' où π' implique un nouveau cycle, on a*

$$\mathbb{1}^T(v_{\pi_*} - v_{\pi'}) \leq \left(1 - \frac{1}{n}\right) \mathbb{1}^T(v_{\pi_*} - v_\pi).$$

Ces observations suffisent à prouver [4] que PI-Simplexe termine après au plus $O(n^4 m^2 \log \frac{n}{1-\gamma}) = \tilde{O}(n^4 m^2)$ itérations. Éliminer complètement la dépendance en le facteur d'actualisation γ—le terme en $O(\log \frac{1}{1-\gamma})$—requiert un travail minutieux supplémentaire (voir Post et Ye (2012)), et se traduit par un facteur supplémentaire en $O(n \log(n))$.

D'un point de vue plus technique, la preuve de Post et Ye (2012) se fonde singulièrement sur des propriétés du vecteur $x_\pi = (I - \gamma P_\pi^T)^{-1}\mathbb{1}$, qui représente une mesure actualisée de la présence en chacun des états lorsqu'on considère une trajectoire induite par la politique π à partir d'un état initial aléatoire uniforme :

$$\forall i \in X, \quad x_\pi(i) = n \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(i_t = i \mid i_0 \sim U,\ a_t = \pi(i_t)),$$

4. Ceci peut être fait par des arguments similaires à la démonstration du théorème 1.9 (voir la Section 6 de Scherrer (2016)).

où nous avons noté U la distribution uniforme sur X. Pour toute politique π et tout état i, on a trivialement $x_\pi(i) \in \left[1, \frac{n}{1-\gamma}\right]$. La démonstration exploite de plus le fait que $x_\pi(i)$ appartient à l'intervalle $[1, n]$ lorsque i est sur un chemin de π, tandis que $x_\pi(i)$ appartient à l'intervalle $\left[\frac{1}{1-\gamma}, \frac{n}{1-\gamma}\right]$ lorsque i est sur un cycle de π. Dans (Scherrer, 2016), nous avons montré qu'il est possible de généraliser l'approche de Post et Ye (2012) aux MDP stochastiques. Étant donnée une politique π d'un MDP stochastique, les états sont soit *récurrents*, soit *transients* (ces deux catégories généralisant respectivement celles de cycles et de chemins pour les MDP déterministes). Ceci nous amène à formuler l'hypothèse suivante.

Hypothèse 1.13. *Soit $\tau_t \geq 1$ et $\tau_r \geq 1$ les plus petits coefficients tels que pour toute politique π, pour tout état i transient pour π,*

$$(1 \leq) x_\pi(i) \leq \tau_t,$$

et pour tout état i récurrent pour π,

$$\frac{n}{(1-\gamma)\tau_r} \leq x_\pi(i) \left(\leq \frac{n}{1-\gamma}\right).$$

La constante τ_t (respectivement τ_r) peut être vue comme une mesure du temps nécessaire pour quitter les états transients (respectivement pour revisiter les états dans les classes récurrentes). En particulier, quand γ tend vers 1, on peut voir que τ_t est un majorant de $\mathcal{L}$, le temps aléatoire nécessaire pour quitter les états transients, car pour toute politique π,

$$\begin{aligned}\lim_{\gamma\to 1} \tau_r \geq \frac{1}{n} \lim_{\gamma\to 1} \sum_{i \text{ transient pour } \pi} x_\pi(i) &= \sum_{t=0}^{\infty} \mathbb{P}(i_t \text{ transient pour } \pi \mid i_0 \sim U,\ a_t = \pi(i_t)) \\ &= \mathbb{E}\left[\, \mathcal{L} \mid i_0 \sim U,\ a_t = \pi(i_t)\right].\end{aligned}$$

De manière similaire, lorsque γ tend vers 1, $\frac{1}{\tau_r}$ est la fréquence asymptotique minimale [5] dans les états récurrents étant donné que la chaîne est initialisée uniformément sur X, car pour toute politique π et tout état récurrent i,

$$\begin{aligned}\lim_{\gamma\to 1} \frac{1-\gamma}{n} x_\pi(i) &= \lim_{\gamma\to 1} (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(i_t = i \mid i_0 \sim U,\ a_t = \pi(i_t)) \\ &= \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{P}(i_t = i \mid i_0 \sim U,\ a_t = \pi(i_t)).\end{aligned}$$

Une fois l'hypothèse 1.13 formulée, nous pouvons généraliser les lemmes 1.11-1.12 comme suit.

Lemme 1.14. *Supposons que le MDP satisfait l'hypothèse 1.13. Après au plus $n^2 m \lceil \tau_t \log(n(\tau_t + 1)) \rceil$ itérations, soit PI-Simplexe termine soit une nouvelle classe récurrente apparaît.*

Lemme 1.15. *Supposons que le MDP satisfait l'hypothèse 1.13. Lorsque PI-Simplexe passe de π à π' où π' implique une nouvelle classe récurrente, on a*

$$\mathbb{1}^T (v_{\pi_*} - v_{\pi'}) \leq \left(1 - \frac{1}{\tau_r}\right) \mathbb{1}^T (v_{\pi_*} - v_\pi).$$

5. Si le MDP est apériodique et irréductible, de sorte qu'il admet une politique stationnaire ν_π pour toute politique π, on peut voir que

$$\frac{1}{\tau_r} = \min_{\pi,\ i \text{ récurrent pour } \pi} \nu_\pi(i).$$

De ces observations générales découle le résultat suivant.

Théorème 1.16. *Si le MDP satisfait l'hypothèse 1.13, alors PI-Simplexe termine après au plus*

$$n^2(m-1)\left(\lceil \tau_r \log(n\tau_r)\rceil + \lceil \tau_r \log(n\tau_t)\rceil\right)\left[(m-1)\lceil n\tau_t \log(n\tau_t)\rceil + \lceil n\tau_t \log(n^2\tau_t)\rceil\right] = \tilde{O}\left(n^3m^2\tau_t\tau_r\right)$$

itérations.

Remarque 1.17. *Ce nouveau résultat est une généralisation stricte de celui énoncé pour les MDP déterministes. En effet, dans le cas déterministe, on a $\tau_t = \tau_r = n$ et il est facile de voir que les lemmes 1.14, 1.15 et le théorème 1.16 impliquent les lemmes 1.11, 1.12 et le théorème 1.10.*

Une conséquence immédiate de ce résultat est que PI-Simplexe est *fortement polynomial* pour un ensemble de MDP qui est significativement plus grand que les MDP déterministes considérés au théorème 1.10.

Corollaire 1.18. *Pour toute famille de MDP telle que τ_t et τ_r sont polynomiaux en n et m (uniformément pour tout γ), PI-Simplexe termine en un temps polynomial en n et m.*

On pourrait se demander si un résultat analogue peut être dérivé pour PI-Howard. Malheureusement, et comme Post et Ye (2012) le mentionnent brièvement, la technique de preuve développée pour PI-Simplexe ne semble pas s'adapter aisément à PI-Howard, parce que les changements de plusieurs actions lors d'une itération peuvent interférer les uns avec les autres de sorte à rendre l'amélioration de la politique très petite. Nous pouvons détailler un peu plus précisément ce qui, dans l'approche décrite plus haut, pose problème. D'un côté, il est possible d'écrire des variations des lemmes 1.11 et 1.14 pour PI-Howard.

Lemme 1.19. *Supposons que le MDP est déterministe. Après au plus n itérations, soit PI-Howard termine soit un nouveau cycle apparaît.*

Lemme 1.20. *Supposons que le MDP satisfait l'hypothèse 1.13. Après au plus $nm\lceil \tau_t \log n\tau_t\rceil$ itérations, soit PI-Howard termine soit une nouvelle classe récurrente apparaît.*

Cependant, de l'autre côté, nous n'avons pas réussi à adapter les lemmes 1.12 ou 1.15. De fait, il semble peu vraisemblable qu'un résultat similaire à celui énoncé dans le lemme 1.12 soit vrai pour PI-Howard. Dans un exemple déterministe récemment décrit par Hansen et Zwick (2010) pour montrer que PI-Howard pouvait requérir au moins $O(n^2)$ itérations, des cycles apparaissent à chaque itération mais la séquence des fonctions valeur satisfait [6] pour toute itération $k < \frac{n^2}{4} + \frac{n}{4}$ et état i,

$$v_*(i) - v_{\pi_{k+1}}(i) \geq \left[1 - \left(\frac{2}{n}\right)^k\right](v_*(i) - v_{\pi_k}(i)).$$

Contrairement au lemme 1.12, on voit ici que lorsque k grandit, la contraction est (exponentiellement rapidement) de moins en moins forte. Par rapport à PI-Simplexe, cela suggère notamment que PI-Howard pourrait (dans le pire cas) souffrir de subtiles pathologies. Plus largement, déterminer le nombre d'itérations nécessaire pour PI-Howard est un problème qui résiste aux efforts des chercheurs depuis maintenant presque 30 ans : il a été originellement énoncé par Schmitz (1985). Dans le cas le plus simple (déterministe), le problème est aujourd'hui encore

6. Ce MDP a un nombre pair d'états $n = 2p$. Dans Hansen et Zwick (2010), le but est de minimiser l'espérance de la somme actualisée des coûts. La fonction valeur optimale satisfait $v_*(i) = -p^N$ pour tout i, avec $N = p^2 + p$. Les politiques générées par PI-Howard ont des fonctions valeur satisfaisant $v_{\pi_k}(i) \in [p^{N-k-1}, p^{N-k}]$. On en déduit que pour toute itération k et tout état i, $\frac{v_*(i)-v_{\pi_{k+1}}(i)}{v_*(i)-v_{\pi_k}(i)} \geq \frac{1+p^{-k-2}}{1+p^{-k}} = 1 - \frac{p^{-k}-p^{-k-2}}{1+p^{-k}} \geq 1 - p^{-k}(1-p^{-2}) \geq 1 - p^{-k}$.

ouvert : le meilleur minorant connu est la borne en $O(n^2)$ de Hansen et Zwick (2010) que nous venons de mentionner, alors que le meilleur majorant est $O(\frac{m^n}{n})$—valable pour les MDPs en général—prouvé par Mansour et Singh (1999).

Comme lot de consolation, nous avons été à même d'adapter dans (Scherrer, 2016) l'analyse décrite plus haut sous une hypothèse structurelle supplémentaire.

Hypothèse 1.21. *L'espace d'états X peut être partitionné en deux ensembles $\mathcal{T}$ et $\mathcal{R}$ tels que pour toute politique π, les états de $\mathcal{T}$ sont transients et ceux de $\mathcal{R}$ sont récurrents.*

En effet, sous cette hypothèse, il est possible de prouver pour PI-Howard une variation du lemme 1.15 introduit pour PI-Simplexe.

Lemme 1.22. *Supposons que le MDP satisfait les Hypothèses 1.13 et 1.21. Lorsque PI-Howard passe de π à π' où π' implique une nouvelle classe récurrente, on a*

$$\mathbb{1}^T(v_{\pi_*} - v_{\pi'}) \leq \left(1 - \frac{1}{\tau_r}\right) \mathbb{1}^T(v_{\pi_*} - v_\pi).$$

Et on peut en déduire le résultat suivant original (qui s'applique d'ailleurs également à PI-Simplexe).

Théorème 1.23. *Si le MDP satisfait les Hypothèses 1.13 et 1.21, alors PI-Howard et PI-Simplexe terminent après au plus $n(m-1)\left(\lceil n\tau_t \log n\tau_t \rceil + \lceil \tau_r \log n\tau_r \rceil\right)$ itérations.*

Il est important de noter que l'hypothèse 1.21 est plutôt restrictive. Elle implique que les deux algorithmes convergent sur les états récurrents indépendemment de ce qui se passe sur les états transients, ce qui permet de réduire l'analyse à deux étapes : 1) l'analyse de la convergence sur les états récurrents ; 2) l'analyse de la convergence sur les états transients (sachant que la convergence a eu lieu sur les états récurrents). L'étude de la première phase (convergence sur les états récurrents) est grandement facilitée par le fait que, dans ce cas, une classe récurrente apparaît à chaque itération (ceci est à contraster avec les lemmes 1.11, 1.14, 1.19 et 1.20 dont la fonction est de montrer que des cycles ou des classes récurrentes apparaissent en un temps raisonnable). De plus, l'analyse de la deuxième phase (convergence sur les états transients) est similaire à celle du cas avec facteur d'actualisation γ (théorèmes 1.7 et 1.9). En d'autres termes, si ce dernier résultat contribue à nous éclairer sur l'efficacité pratique usuelle de PI-Howard et PI-Simplexe, une analyse plus générale de PI-Howard est encore à faire. Sur ce point, on notera pour finir que l'analyse de Akian et Gaubert (2013) permet d'obtenir un résultat similaire au théorème 1.23, mais sous une hypothèse plus légère (il existe un état vers lequel on retourne en temps moyen fini).

1.4 Bilan et perspectives

Dans ce chapitre, nous avons décrit plusieurs contributions à la compréhension de deux algorithmes de type PI. Lorsque le facteur d'actualisation γ est fixé, nos résultats (théorèmes 1.7 et 1.9) permettent de supprimer un facteur $\log n$ inutile dans les analyses précédentes. Sous des hypothèses structurelles du MDP, nous avons proposé des majorants indépendants de γ : étant données une mesure τ_t du temps maximal pour quitter les zones transientes et une mesure τ_r du temps maximal pour revisiter les états dans les classes récurrentes sous l'ensemble des politiques, nous avons montré (théorème 1.16) que PI-Simplexe termine après au plus $\tilde{O}\left(n^3 m^2 \tau_t \tau_r\right)$, itérations, ce qui généralise un résultat récent de Post et Ye (2012) sur les PDMs déterministes dans lesquels on a $\tau_t = \tau_r = n$.

Une question naturelle qui se pose après cette étude de complexité, est de savoir dans quelle mesure des résultats similaires pourraient être également obtenus pour d'autres algorithmes de programmation dynamique, par exemple dans un schéma de type Itérations sur les Politique Modifié (équation (1) page 9) qui contient PI comme cas particulier, précisément lorsque $m = \infty$. Malheureusement, comme nous l'avons montré dans (Feinberg *et al.*, 2014), il est assez facile de voir que la simple garantie de convergence en temps fini disparaît dès qu'on s'éloigne un tant soit peu de PI ; on exhibe en effet une instance de MDP pour laquelle le nombre d'itérations pour obtenir la politique optimale peut être rendu aussi grand que l'on veut pour n'importe quelle valeur du paramètre d'interpolation $m < \infty$.

Comme nous l'avons déja signalé, l'analyse de la complexité de la variation la plus standard de PI, PI-Howard, qui est un problème ouvert depuis près de 30 ans, constitue la principale perspective de ce travail. Dans cette direction, une première question sur laquelle j'ai commencé à travailler avec Romain Hollanders, est de voir dans quelle mesure des schémas de type PI qui changent plusieurs actions à chaque étape ($|S_k| > 1$) peuvent jouir de garanties similaires à celle dérivées pour PI-Simplexe. Plus généralement, la question de la complexité de ces schémas PI pourrait (en tant que cas particulier) éclairer la question de la complexité de la programmation linéaire, grand classique des grandes questions ouvertes en informatique. Sur ce thème, une première étape envisageable consiste à voir dans quelle mesure une analyse de type *complexité lisse* (Spielman et Teng, 2004), qui a porté ses fruits pour la programmation linéaire, est envisageable pour PI-Howard.

Chapitre 2

Différences temporelles ou résidu de Bellman ? L'interprétation "projection oblique"

Dans ce chapitre et le suivant, nous allons nous focaliser sur le problème de l'estimation de la fonction valeur d'une politique fixe π, c'est-à-dire sur le problème linéaire

$$v_\pi = T_\pi v_\pi,$$

lorsque la taille de l'espace d'états $n \in \mathbb{N} \cup \{\infty\}$ est trop grande pour qu'on puisse envisager une résolution exacte. Résoudre ce problème peut notamment s'inscrire dans un schéma approché de type Itérations sur les Politiques. Pour alléger les notations, nous laisserons ici de côté l'indice π des objets intervenant dans l'équation de Bellman : r, P, v ; notamment, T désignera ici l'opérateur de Bellman affine caractérisant la valeur de cette politique (et non l'opérateur d'optimalité). L'équation linéaire satisfaite par la fonction de valeur, $v = Tv = r + \gamma Pv$, peut être en théorie résolue analytiquement :

$$v = (I - \gamma P)^{-1} r = L^{-1} r$$

avec $L = I - \gamma P$; en pratique néanmoins, un grand espace d'états peut rendre cette résolution hors d'atteinte numérique. Nous allons considérer deux méthodes d'*approximation linéaire* relativement populaires de la fonction valeur : le calcul du *point fixe des différences temporelles* (TD) (Sutton, 1988; Tsitsiklis et Van Roy, 1997; Bertsekas et Tsitsiklis, 1996; Sutton et Barto, 1998) que Antos *et al.* (2008); Farahmand *et al.* (2008); Sutton *et al.* (2009) ont présentée comme la minimisation du résidu d'une équation de Bellman projetée, et la *minimisation du résidu de l'équation de Bellman* (BR) (Schweitzer et Seidmann, 1985; Baird, 1995), sous-entendu ici non projetée. Dans ce chapitre qui a donné lieu à une publication (Scherrer, 2010), nous présentons des contributions analytiques et empiriques qui permettent de mieux comprendre les avantages et les inconvénients de chacune d'elles. Nous montrons notamment qu'elle peuvent être toutes deux analysées dans le cadre des *projections obliques*, ce qui permet—à moindres efforts que Yu et Bertsekas (2010) qui a précédemment considéré cette problématique—de calculer des garanties sur l'erreur d'approximation.

2.1 Méthodes de projections

Nous considérons ici que nous cherchons à approcher la fonction valeur v par une approximation $\hat{v}$ qui vit dans un espace plus petit. Un cadre simple et bien compris est celui de la

paramétrisation linéaire :

$$\forall i,\ \hat{v}(i) = \sum_{j=1}^{d} w_j \phi_j(i),$$

où d est typiquement beaucoup plus petit que n, les $(\phi_j)_{1\leq j\leq d}$ sont des fonctions de bases qui idéalement rendent compte de la forme générale de v, et les $(w_j)_{1\leq j\leq d}$ sont des poids qui caractérisent la valeur approchée $\hat{v}$. Pour tout i et j, notons ϕ_j le vecteur de $\mathbb{R}^n$ correspondant à la $j^{\text{ième}}$ fonction de base et $\phi(i)$ le vecteur de $\mathbb{R}^d$ donnant les d valeurs des d fonctions de base en l'état i. Pour tout vecteur (ou toute matrice) X, notons X' son (sa) transposée. La matrice de taille $n \times d$ suivante

$$\Phi = (\phi_1 \dots \phi_m) = \begin{pmatrix} \phi(i_1)' \\ \vdots \\ \phi(i_N)' \end{pmatrix}$$

permet d'écrire la paramétrisation de manière condensée :

$$\hat{v} = \Phi w,$$

où $w = (w_1, ..., w_m)'$ est le vecteur de *poids*. Nous noterons désormais $\mathrm{Im}\,(\Phi)$ le sous-espace de $\mathbb{R}^n$ engendré par les vecteurs $(\phi_j)_{1\leq j\leq d}$ que nous supposerons deux à deux linéairement indépendants.

Une approximateion $\hat{v}$ de v peut être obtenue en minimisant $\hat{v} \mapsto \|\hat{v} - v\|$ pour une certaine norme $\|\cdot\|$, c'est-à-dire en projetant v sur $\mathrm{Im}\,(\Phi)$ orthogonalement par rapport à un produit scalaire induisant $\|\cdot\|$. D'une manière très générale, tout matrice semi-définie positive Q de taille $n \times n$ induit la norme quadratique $\|\cdot\|_Q$ sur $\mathbb{R}^n$ définie par

$$\|v\|_Q = \sqrt{v'Qv}.$$

Il est alors connu que la projection orthogonale correspondante, que nous noterons $\Pi_{\|\cdot\|_Q}$, a la forme analytique suivante :

$$\Pi_{\|\cdot\|_Q} = \Phi \pi_{\|\cdot\|_Q}, \quad \text{où } \pi_{\|\cdot\|_Q} = (\Phi' Q \Phi)^{-1} \Phi' Q$$

est l'application linéaire de $\mathbb{R}^n$ vers $\mathbb{R}^d$ qui renvoie les coordonnées de la projection d'un point dans la base $(\phi_1, \dots, \phi_m)$. Avec ces notations, on a notamment les identités suivantes :

$$\pi_{\|\cdot\|_Q} \Phi = I \quad \text{et} \quad \pi_{\|\cdot\|_Q} \Pi_{\|\cdot\|_Q} = \pi_{\|\cdot\|_Q}.$$

Dans le cadre considéré d'un MDP avec une politique fixe, qui modélise donc une chaîne de Markov dont chaque transition a une récompense, on considère habituellement une norme et une projection particulières. Notons $\mu = (\mu_i)$ une loi sur X qui met une masse non nulle sur tout état ($\mu > 0$). Soit D_μ la matrice diagonale ayant les éléments de μ sur la diagonale. Considérons alors la projection orthogonale de $\mathbb{R}^n$ sur $\mathrm{Im}\,(\Phi)$ par rapport à la norme quadratique à poids μ :

$$\|v\|_\mu = \sqrt{\sum_{j=1}^{N} \mu_i {v_i}^2} = \sqrt{v' D_\mu v}.$$

Pour alléger, nous noterons cette projection $\Pi := \Pi_{\|\cdot\|_\mu} = \Phi\pi$ avec $\pi := \pi_{\|\cdot\|_\mu} = (\Phi' D_\mu \Phi)^{-1} \Phi' D_\mu$. Idéalement, étant donnée cette projection, on souhaiterait calculer l'approximation *optimale* :

$$\hat{v}_{opt} = \Phi w_{opt} \text{ avec } w_{opt} = \pi v = \pi L^{-1} r.$$

Ceci peut être fait avec une approche de type Monte-Carlo dans laquelle on simulerait des trajectoires initialisées selon la distribution μ—ce qui serait alors très proche des algorithmes TD(1) (Sutton et Barto, 1998) et LSTD(1) (Bertsekas et Tsitsiklis, 1996; Boyan, 2002)—mais une telle approche requiert en général de simuler de longues trajectoires et souffre d'une forte variance. Les méthodes de projection sur lesquelles nous allons nous concentrer, sont des alternatives qui ont seulement besoin d'échantillons de transitions simples (de couples d'états (i_t, i_{t+1}) et de récompenses r_t) réalisées par la chaîne de Markov.

La méthode du point fixe des différences temporelles (TD) Le principe de l'approche TD (Sutton, 1988; Tsitsiklis et Van Roy, 1997; Bertsekas et Tsitsiklis, 1996; Sutton et Barto, 1998) est de chercher un point fixe de l'opérateur ΠT, c'est-à-dire une valeur $\hat{v}_{TD}$ dans $\mathrm{Im}\,(\Phi)$ satisfaisant

$$\hat{v}_{TD} = \Pi T \hat{v}_{TD}.$$

Sous l'hypothèse que l'inverse de la matrice ci-dessous existe [1], on peut montrer [2] que $\hat{v}_{TD} = \Phi w_{TD}$ avec

$$w_{TD} = (\Phi' D_\mu L \Phi)^{-1} \Phi' D_\mu r$$

Comme remarqué par Antos *et al.* (2008); Farahmand *et al.* (2008); Sutton *et al.* (2009), lorsque l'inverse existe, ce calcul est équivalent à celui de calculer la valeur $\hat{v} \in \mathrm{Im}\,(\Phi)$ qui minimise (jusqu'à 0) l'*erreur TD* [3]

$$E_{TD}(\hat{v}) := \|\hat{v} - \Pi T \hat{v}\|_\mu.$$

La méthode de minimisation du résidu de Bellman (BR) Le principe de cette seconde approche BR (Schweitzer et Seidmann, 1985; Baird, 1995) est de chercher une valeur $\hat{v} \in \mathrm{Im}\,(\Phi)$ qui minimise la norme du résidu de Bellman, c'est-à-dire la quantité

$$E_{BR}(\hat{v}) := \|\hat{v} - T\hat{v}\|_\mu.$$

Comme $\hat{v}$ est de la forme Φw, on peut voir que $E_{BR}(\hat{v}) = \|\Phi w - \gamma P \Phi w - r\|_\mu = \|\Psi w - r\|_\mu$ avec la notation $\Psi = L\Phi$. En utilisant des arguments standards de type moindres carrés, on peut montrer que le minimum est atteint pour $\hat{v}_{BR} = \Phi w_{BR}$ avec

$$w_{BR} = (\Psi' D_\mu \Psi)^{-1} \Psi' D_\mu r. \tag{2.1}$$

Notons que dans ce cas, l'inverse existe toujours (Schoknecht, 2002).

Exemple 2.1. *Considérons le MDP ayant 2 états tel que* $P = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$. *Notons les récompenses* r_1 *et* r_2. *On a ainsi* $v(1) = r_1 + \frac{\gamma r_2}{1-\gamma}$ *et* $v(2) = \frac{r_2}{1-\gamma}$. *Considérons l'approximation à l'aide d'une unique fonction de base telle que* $\Phi = (1\ 2)'$, *avec une distribution uniforme* $\mu = (.5\ .5)'$. *Comme* $\Phi' D_\mu \Phi = \frac{5}{2}$, *on a* $\pi = \left(\frac{1}{5}\ \frac{2}{5}\right)$, *et le poids de la meilleure approximation est par conséquent* $w_{opt} = \pi v = \frac{1}{5} r_1 + \frac{2+\gamma}{5(1-\gamma)} r_2$. *Cet exemple a été initialement proposé par Tsitsiklis et Van Roy (1997); Bertsekas et Tsitsiklis (1996) afin de montrer qu'un algorithme de type Itérations sur les Valeurs avec approximation linéaire et échantillonnage peut diverger si les échantillons ne*

1. Ce n'est pas nécessairement le cas, comme nous le verrons dans l'exemple 2.1.
2. Nous ne détaillons pas ici dans la mesure où la Section 2.3 généralisera cette dérivation.
3. Cette remarque est aussi vraie si l'on remplace $\|\cdot\|_\mu$ par n'importe quelle norme équivalente $\|\cdot\|$. Cette observation, simple, a permis à Sutton *et al.* (2009) de proposer des algorithmes de type gradient dans le cas *off-policy*, c'est-à-dire où les échantillons ne proviennent pas de la même politique que celle que l'on souhaite évaluer.

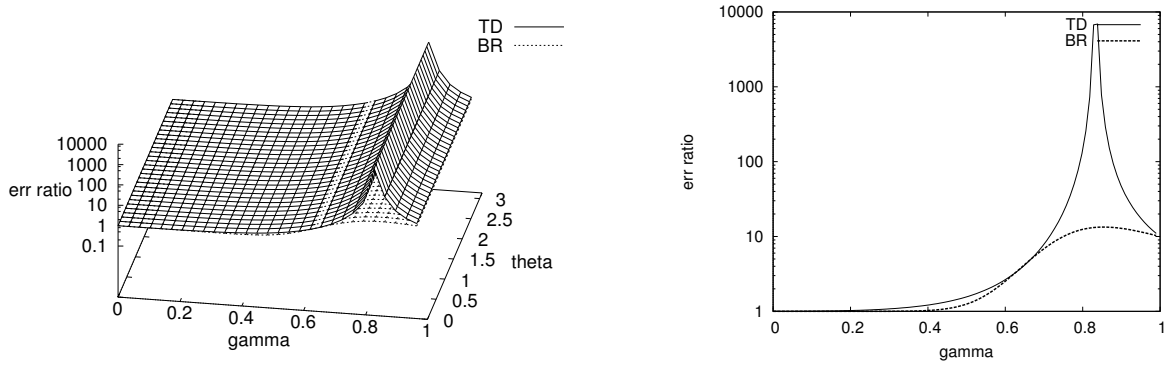


FIGURE 2.1 – Rapports (en échelle logarithmique) entre d'une part les projections TD et BR, et d'autre part la meilleure approximation dans l'exemple 2.1, en fonction du facteur d'actualisation γ et du paramètre θ de la récompense (Gauche). Il s'avère que ces surfaces sont indépendantes de θ, donc nous traçons le graphe montrant uniquement la dépendance en γ (Droite).

sont pas tirés selon la loi stationnaire de la politique considérée. Dans ces travaux, les auteurs considèrent le cas $r_1 = r_2 = 0$ *si bien qu'on a divergence quand bien même la fonction de valeur* $v(0) = v(1) = 0$ *appartient à l'espace* $Im(\Phi)$*. Dans le cas* $r_1 = r_2 = 0$*, les méthodes TD et BR calculent la bonne solution (nous verrons un peu plus loin que c'est une propriété générale lorsque la fonction de valeur appartient à* $Im(\Phi)$*). Nous avons donc étendu ce modèle en prenant* $(r_1, r_2) \neq (0, 0)$*. Comme la valeur est une fonction linéaire de la récompense, on considère sans perte de généralité la forme* $(r_1, r_2) = (\cos\theta, \sin\theta)$*.*

Considérons tout d'abord TD : on a $\Phi' D_\mu = \left(\frac{1}{2}\ 1\right)$*,* $(I - \gamma P)\Phi = (1 - 2\gamma\ 1 - \gamma)$*, ainsi* $(\Phi' D_\mu \Psi) = \frac{5}{2} - 3\gamma$ *et* $\Phi' D_\mu r = \frac{r_1}{2} + r_2$*. En conséquence, le poids de l'approximation TD est* $w_{TD} = \frac{r_1 + 2r_2}{5 - 6\gamma}$*. On remarque ici que la valeur* $\gamma = 5/6$ *est singulière. Considérons maintenant BR : on peut voir que* $(\Psi' D_\mu \Psi)^{-1} = \frac{(1-2\gamma)^2 + (2-2\gamma)^2}{2}$ *et* $\Psi' D_\mu r = \frac{(1-2\gamma)r_1 + (2-2\gamma)r_2}{2}$*. Ainsi le poids de l'approximation BR est* $w_{BR} = \frac{(1-2\gamma)r_1 + (2-2\gamma)r_2}{(1-2\gamma)^2 + (2-2\gamma)^2}$*.*

Pour toutes ces approximations, on peut calculer l'erreur quadratique e par rapport à la vraie valeur v. Pour tout poids $w \in \{w_{opt}, w_{TD}, w_{BR}\}$*,* $e(w) = \|v - \Phi w\|_\mu^2 = \frac{1}{2}(v(1) - w)^2 + \frac{1}{2}(v(2) - 2w)^2$*. A la figure 2.1, nous avons tracé les rapports* $\frac{e(w_{TD})}{e(w_{opt})}$ *et* $\frac{e(w_{BR})}{e(w_{opt})}$ *sur une échelle logarithmique (ils sont pas construction plus grands que 1) par rapport à* θ *et* γ*. Il s'avère que ces surfaces sont indépendantes de* θ *; cette figure montre également le graphe en fonction de la variable* γ *seulement. On peut observer que pour n'importe quel choix de récompense, la méthode BR est plus précise que la méthode TD. De plus, quand* γ *est dans le voisinage de* $\frac{5}{6}$*, le rapport correspondant à l'erreur TD tends vers l'infini tandis que celui de BR reste borné. Cet exemple montre qu'il existe des MDPs où la projection BR peut être arbitrairement meilleure que la projection TD. Il serait ici prématuré de conclure que BR est toujours préférable à TD. La littérature contient plusieurs arguments en faveur de TD, dont celui que nous considérons dans l'exemple suivant.*

Exemple 2.2. *Sutton* et al. *(2009) ont décrit un MDP à 3 états où la méthode TD calcule la meilleure projection $\hat{v}_{opt}$ tandis que ce n'est pas le cas de la méthode BR. L'idée derrière ce MDP peut être décrit d'une manière générale*[4]*. Supposons que nous avons un MDP ayant $k+l$ états, et dont l'équation de Bellman a une structure triangulaire par blocs :*

$$v_1 = \gamma P_1 v_1 + r_1$$
$$v_2 = \gamma P_{21} v_1 + P_{22} v_2 + r_2$$

où $v_1 \in \mathbb{R}^k$ et $v_2 \in \mathbb{R}^l$, la concaténation des vecteurs v_1 et v_2 formant la fonction valeur. Supposons de plus que l'espace d'approximation $Im(\Phi)$ est $\mathbb{R}^k \times S_2$ où S_2 est un sous-espace de $\mathbb{R}^l$. Avec TD, on obtient la fonction de valeur exacte sur les k premières composantes, tandis que ce n'est pas le cas pour BR : la satisfaction du premier bloc d'équations ci-dessus est détériorée de sorte à réduire l'erreur sur le second bloc (qui implique également v_1). Dans un contexte d'optimisation du contrôle où la valeur serait utilisée pour déduire une politique gloutonne, ce genre d'exemples peut avoir des conséquences dramatiques : v_1 est relié aux récompenses d'états futurs accessibles à partir de ceux correspondants à v_2, et les récompenses futures sont l'essentiel lorsqu'il s'agit de prendre des décisions.

Globalement, les deux approches génèrent des biais de différents types, et distribuent les erreurs différemment. Pour approfondir leur comparaison, l'essentiel de la suite de ce chapitre est consacré à des arguments plus analytiques.

2.2 Une relation et des problèmes de stabilité

Bien que plusieurs travaux de la littérature aient comparé les deux méthodes (Schoknecht, 2002; Lagoudakis et Parr, 2003b; Munos, 2003; Yu et Bertsekas, 2010), la simple observation suivante n'avait à notre connaissance jamais été soulignée.

Théorème 2.3. *L'erreur BR est une borne supérieure de l'erreur TD, et plus précisément :*

$$\forall \hat{v} \in Im(\Phi), E_{BR}(\hat{v})^2 = E_{TD}(\hat{v})^2 + \|T\hat{v} - \Pi T\hat{v}\|_\mu^2.$$

Démonstration. Ceci est simplement une conséquence du théorème de Pythagore, dans la mesure où $\Pi T\hat{v} - T\hat{v}$ est orthogonal à $\mathrm{Im}(\Phi)$ et $\hat{v} - \Pi T\hat{v}$ appartient à $\mathrm{Im}(\Phi)$. □

Ceci implique que si on est capable de rendre l'erreur BR petite, alors l'erreur TD sera également petite. Dans le cas limite où l'erreur BR est nulle, l'erreur TD le sera également.

L'une des motivations historiques pour minimiser l'erreur BR est liée à la garantie de Williams et Baird (1993) :

$$\forall \hat{v},\ \|v - \hat{v}\|_\infty \leq \frac{1}{1-\gamma} \|T\hat{v} - \hat{v}\|_\infty.$$

Dans la mesure où l'on contrôle l'erreur de projection seulement en norme quadratique, c'est la variation suivante de ce résultat[5] qui est pertinente :

$$\forall \hat{v},\ \|v - \hat{v}\|_\mu \leq \frac{\sqrt{C(\mu)}}{1-\gamma} \|T\hat{v} - \hat{v}\|_\mu$$

4. La suite de la description est fortement inspirée d'une communication personnelle avec Janey Yu.

5. La preuve est une conséquence de l'inégalité de Jensen et les arguments sont similaires à ceux donnés par Munos (2003).

où $C(\mu) := \max_{i,j} \frac{p_{ij}}{\mu_i}$ est un *coefficient de concentrabilité*, qui peut être vu comme une mesure de stochasticité de la chaîne de Markov sous-jacente [6]. Ce résulte montre qu'il est raisonnable de minimiser l'erreur BR, dans la mesure où elle permet de contrôler indirectement l'erreur d'approximation $\|v - \hat{v}_{BR}\|_\mu$.

Du côté de TD, il n'existe pas de résultat similaire. Le simple fait que nous ayons construit l'exemple 2.1 où la valeur estimée par la projection TD est numériquement instable montre qu'on ne peut pas prouver un tel résultat. Le théorème 2.3 que nous venons d'énoncer permet d'en dire un peu plus. En minimisant l'erreur TD, on minimise seulement une partie de l'erreur BR, tandis qu'on ignore l'autre, $\|Tv - \Pi Tv\|_\mu^2$, qui peut être interprétée comme une mesure d'adéquation entre la projection Π et l'opérateur de Bellman T. Dans l'exemple 2.1, l'erreur d'approximation de TD tend vers l'infini parce que ce terme d'adéquation diverge.

Un regard complémentaire sur la potentielle instabilité de TD, a été qualifié de *problème d'incompatibilité de norme* (Bertsekas et Tsitsiklis, 1996; Guestrin *et al.*, 2001), et peut être revisité via la notion de coefficient de concentrabilité. Les matrices stochastiques P satisfont $\|P\|_\infty = 1$, ce qui fait que l'opérateur de Bellman T est contractant de coefficient γ et que son point fixe est bien défini. La projection orthogonale par rapport à la norme $\|\cdot\|_\mu$ est telle que $\|\Pi\|_\mu = 1$. Ainsi P et Π sont de norme 1, mais pour des normes différentes. Malheureusement, une borne générale (optimale) pour les projections linéaires est $\|\Pi\|_\infty \leq \frac{1+\sqrt{n}}{2}$ (Thompson, 1996) et on peut montrer [7] que $\|P\|_\mu \leq \sqrt{C(\mu)}$, ce qui peut en général être également de l'ordre $O(\sqrt{n})$. En conséquence, les normes $\|\Pi P\|_\infty$ et $\|\Pi P\|_\mu$ peuvent toutes deux être plus grandes que 1, et le point fixe de l'équation de Bellman projetée n'est en général pas bien défini. Une exception remarquable à ce genre de problème, où ΠP est de norme 1, est lorsqu'on peut prouver que $\|P\|_\mu = 1$, comme par exemple lorsque μ est une mesure stationnaire de P (Bertsekas et Tsitsiklis, 1996; Tsitsiklis et Van Roy, 1997). Dans ce cas, on peut en déduire (Bertsekas et Tsitsiklis, 1996; Tsitsiklis et Van Roy, 1997) que

$$\|v - \hat{v}_{TD}\|_\mu \leq \frac{1}{\sqrt{1-\gamma^2}} \|v - \hat{v}_{opt}\|_\mu. \tag{2.2}$$

Une autre exception notable est lorsque $\|\Pi\|_{max} = 1$, comme dans le cas d'approximation linéaire de type "averager" (Gordon, 1995). Cependant, en général, la stabilité numérique de l'approche TD est difficile à garantir.

2.3 Une vision unifiée via les projections obliques

Dans l'approche TD, on cherche à trouver le point fixe de la composition d'une projection orthogonale Π et de l'opérateur de Bellman T. Nous allons ici supposer que nous utilisons une projection Π *non nécessairement orthogonale* sur $\mathrm{Im}\,(\Phi)$, c'est-à-dire une application linéaire qui satisfait $\Pi^2 = \Pi$ est dont le domaine d'arrivée est $\mathrm{Im}\,(\Phi)$. De la manière la plus générale, ces opérateurs sont appelés des *projections obliques* et ont la forme analytique suivante :

$$\Pi_X = \Phi \pi_X \quad \text{avec } \pi_X = (X'\Phi)^{-1} X'.$$

6. Si μ est uniforme et n est fini, alors il existe une telle constante $C(\mu) \in (1, n)$, où l'on rappelle que n est la taille de l'espace d'états. Dans un tel cas, $C(\mu)$ est minimal lorsque la chaîne de Markov est telle que tous les états suivants sont choisis selon la loi uniforme, et maximal dès qu'il existe une transition déterministe qui va concentrer la masse en un état. Nous reviendrons sur ces coefficients dans les chapitres suivants, notamment dans le dernier chapitre de ce manuscrit.

7. Pour tout x, $\|Px\|_\mu^2 \leq \|x\|_{\mu P}^2 \leq C(\mu)\|x\|_\mu^2$. L'argument pour la première inégalité implique l'inégalité de Jensen et est de nouveau proche ce ce qui est fait dans (Munos, 2003).

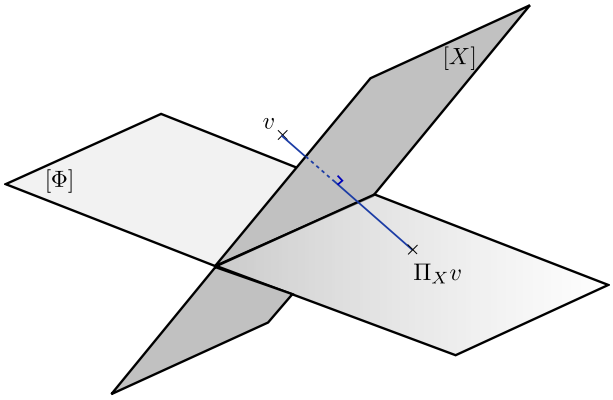


FIGURE 2.2 – Projection oblique de v sur $\mathrm{Im}\,(\Phi)$ orthogonalement à $\mathrm{Im}\,(X)$.

Ici, le paramètre X caractérise la direction avec laquelle on effectue la projection. Précisement, Π_X est la projection sur $\mathrm{Im}\,(\Phi)$ orthogonalement à $\mathrm{Im}\,(X)$ (une illustration est donnée à la figure 2.2). De même que dans le cas des projections orthogonales, nous avons les identités $\pi_X \Phi = I$ et $\pi_X \Pi_X = \pi_X$. Rappelons que nous avons noté $L = I - \gamma P$. Le résultat principal de notre analyse comparée des approches TD et BR est énoncé ci-dessous.

Théorème 2.4. *Notons $X_{TD} = D_\mu \Phi$ et $X_{BR} = D_\mu L \Phi$.*

1. *Le point fixe de l'approche TD et le projeté de résidu minimal BR sont, respectivement pour $X = X_{TD}$ et $X = X_{BR}$, le point fixe de l'équation projetée*

$$\hat{v}_X = \Pi_X T \hat{v}_X.$$

2. *Quand ce point fixe existe, la solution de cette équation projetée est la projection de la (vraie) valeur v sur $Im\,(\Phi)$ orthogonalement à $Im\,(L'X)$, c'est-à-dire que*

$$\hat{v}_X = \Pi_{L'X}\ v.$$

Démonstration. On commence par montrer le deuxième point. Notant $\hat{v}_X = \Phi w_X$, l'équation du point fixe est :

$$\Phi w_X = \Pi_X (r + \gamma P \Phi w_x).$$

En multipliant par π_X, on obtient

$$w_X = \pi_X (r + \gamma P \Phi w_x),$$

dont on déduit

$$w_X = (I - \gamma \pi_X P \Phi)^{-1} \pi_X r.$$

En utilisant la définition de π_X, on a

$$\begin{aligned} w_X &= (I - \gamma (X'\Phi)^{-1} X' P \Phi)^{-1} (X'\Phi)^{-1} X' r \\ &= \left[(X'\Phi)(I - \gamma (X'\Phi)^{-1} X' P \Phi) \right]^{-1} X' r \\ &= (X'(I - \gamma P)\Phi)^{-1} X' r \qquad (2.3) \\ &= (X' L \Phi)^{-1} X' L v \\ &= \pi_{L'X}\ v \end{aligned}$$

où nous avons utilisé après l'équation (2.3) le fait que $r = Lv$.

Prouvons maintenant le premier point. Le fait que TD soit un cas particulier avec $X = D_\mu\Phi$ est trivial par construction dans la mesure où Π_X est la projection orthogonale par rapport à $\|\cdot\|_\mu$. Enfin, lorsque $X = D_\mu L\Phi$, il suffit d'observer les équations (2.1) et (2.3) et la défintion de $\Psi = L\Phi$ pour voir que $w_X = w_{BR}$. □

Au delà de l'interprétation géométrique qu'on peut apprécier en tant que telle, une conséquence directe et pragmatique du théorème 2.4 est qu'on peut facilement dériver des bornes d'erreur pour les méthodes TD, BR, et de manière générale pour toute autre méthode basée sur une projection sur $\mathrm{Im}\,(\Phi)$ orthogonalement à $\mathrm{Im}\,(X)$ avec X quelconque. Pour toute matrice M, notons $\sigma(M)$ son rayon spectral. On a le résultat suivant.

Théorème 2.5. *Pour n'importe quel choix de X, l'erreur d'approximation satisfait*

$$\begin{aligned}\|v - \hat{v}_X\|_\mu &\leq \|\Pi_{L'X}\|_\mu \|v - \hat{v}_{opt}\|_\mu \\ &= \sqrt{\sigma(ABCB')}\|v - \hat{v}_{opt}\|_\mu\end{aligned} \tag{2.4}$$

où $A = \Phi' D_\mu \Phi$, $B = (X'L\Phi)^{-1}$ et $C = XLD_\mu^{-1}L'X$ sont des matrices de taille $d \times d$.

Ainsi, pour tout X, l'amplification de la plus petite erreur de projection possible $\|v - \hat{v}_{opt}\|_\mu$ dépend de la norme de la projection oblique associée, qui peut être estimée comme le rayon spectral d'un produit de matrices de (petite) taille $d \times d$. Voici un corollaire simple de ce résultat : si la fonction valeur v appartient à l'espace $\mathrm{Im}\,(\Phi)$ sur lequel on effectue la projection (dans ce cas $v = \hat{v}_{opt}$), alors toutes les méthodes de point fixes basées sur des projections obliques (dont TD et BR) la calculent exactement ($\hat{v}_X = v$).

Démonstration du théorème 2.5. Le théorème 2.4 implique que

$$v - \hat{v}_X = (I - \Pi_{L'X})v = (I - \Pi_{L'X})(I - \Pi_{D_\mu\Phi})v.$$

où nous avons utilisé le fait que $\Pi_{L'X}\Pi_{D_\mu\Phi} = \Pi_{D_\mu\Phi}$, vu que $\Pi_{L'X}$ et $\Pi_{D_\mu\Phi}$ sont toutes deux des projections sur $\mathrm{Im}\,(\Phi)$. En prenant la norme, on obtient

$$\begin{aligned}\|v - \hat{v}_X\|_\mu &\leq \|I - \Pi_{L'X}\|_\mu \|v - \Pi_{D_\mu\Phi} v\|_\mu \\ &= \|\Pi_{L'X}\|_\mu \|v - \hat{v}_{opt}\|_\mu\end{aligned}$$

où nous avons utilisé la définition de $\hat{v}_{opt}$, et le fait que $\|I - \Pi_{L'X}\|_\mu = \|\Pi_{L'X}\|_\mu$ lorsque $\Pi_{L'X}$ est une projection non-triviale (Szyld, 2006). On en déduit l'équation (2.4).

Afin d'évaluer cette norme en termes de matrices de taille $d \times d$, on utilise le lemme suivant.

Lemme 2.6 (Yu et Bertsekas (2010))**.** *Soient Y une matrice de taille $n \times d$ et Z une matrice de taille $d \times n$. On a la relation suivante :*

$$\|YZ\|_\mu^2 = \sigma\left((Y'D_\mu Y)(ZD_\mu^{-1}Z')\right).$$

Appliqué à la matrice de projection $\Pi_{L'X} = \Phi\pi_{L'X}$, on obtient

$$\begin{aligned}\|\Pi_{L'X}\|_\mu^2 &= \|\Phi\pi_{L'X}\|_\mu^2 \\ &= \sigma[(\Phi' D_\mu \Phi)(\pi_{L'X} D_\mu^{-1}(\pi_{L'X})')] \\ &= \sigma[\Phi' D_\mu \Phi (X'L\Phi)^{-1} X'LD_\mu^{-1}L'X(\Phi'L'X)^{-1}] \\ &= \sigma[ABCB'].\end{aligned}$$

□

Le théorème 2.4 que nous venons de décrire est intimement lié à un article de Schoknecht (2002), dans lequel l'auteur dérive la caractérisation suivante des méthodes TD et BR.

Théorème 2.7 (Schoknecht (2002))**.** *Les solutions calculées par les méthodes TD et BR sont les projections orthogonales de la valeur v respectivement induites par la seminorme*[8] $\|\cdot\|_{Q_{TD}}$ *avec* $Q_{TD} = L'D_\mu\Phi\Phi'D_\mu L$ *et par la norme* $\|\cdot\|_{Q_{BR}}$ *with* $Q_{BR} = L'D_\mu L$*.*

Cette caractérisation en termes de "projection orthogonale" par rapport à des (semi-)normes et notre caractérisation originale en terme de "projection oblique" sont en fait équivalentes. D'une part, pour l'approche BR, il est facile de voir que $\Pi_{\|\cdot\|_{Q_{BR}}} = \Pi_{L'X_{BR}}$. D'autre part, pour TD, en écrivant $Y = L'X_{TD}$, on a simplement besoin de remarquer que

$$\begin{aligned}\Pi_{L'X_{TD}} &= \Pi_Y \\ &= \Phi(Y'\Phi)^{-1}Y' \\ &= \Phi(Y'\Phi)^{-1}(\Phi'Y)^{-1}(\Phi'Y)Y' \\ &= \Phi(\Phi'YY'\Phi)^{-1}\Phi'YY' \\ &= \Pi_{\|\cdot\|_{Q_{TD}}}.\end{aligned}$$

L'analyse de Schoknecht (2002) suggère que les approches TD et BR sont optimales pour des critères différents, puisque les deux visent à calculer une approximation $\hat{v} \in \mathrm{Im}\,(\Phi)$ qui minimise $\|\hat{v} - v\|$ pour une certaine (semi-)norme $\|\cdot\|$. De manière complémentaire, notre résultat suggère plutot qu'aucune des deux n'est optimale, dans la mesure où aucune des deux ne correspond à la direction de projection $X^* := L'^{-1}D_\mu\Phi$ pour laquelle $\hat{v}_{X^*} = \Pi_{L'X^*}v = \Pi_{D_\mu\Phi}v = \hat{v}_{opt}$.

L'étude que nous venons de présenter ici, et notamment les théorèmes 2.4 et 2.5, revisitent le travail de Yu et Bertsekas (2010), dans lequel les auteurs dérivent des bornes d'erreur similaire pour les approches TD et BR. Notre approche ressemble beaucoup à la leur : 1) nous dérivons une relation linéaire entre la fonction de valeur v, son approximation $\hat{v}$, et la meilleure approximation $\hat{v}_{opt}$, puis 2) nous exprimons la norme des matrices impliquées en fonction du rayon spectral de petites matrices, via d'ailleurs le lemme 2.6 que nous empruntons à Yu et Bertsekas (2010). D'un point de vue quantitatif, nos bornes sont identiques à celles dérivées par Yu et Bertsekas (2010). Ceci a deux conséquences immédiates : comme montré par Yu et Bertsekas (2010), 1) notre borne est optimale dans le sens où il existe une fonction récompense pour laquelle l'inégalité est en fait une égalité, et 2) elle est toujours plus fine que la borne donnée à l'équation (2.2) de Bertsekas et Tsitsiklis (1996); Tsitsiklis et Van Roy (1997). Cependant, notre approche est qualitativement différente : en soulignant la relation de type *projection oblique* entre la fonction de valeur v et son approximation $\hat{v}$, nous donnons une intuition géométrique sur les deux méthodes, et cela simplifie grandement les résultats et les démonstrations pour les obtenir, les arguments dans Yu et Bertsekas (2010) étant en effet beaucoup plus complexes.

Enfin, il y a une différence importante entre nos résultats et les deux travaux similaires que nous venons de mentionner. L'analyse que nous proposons est *unifiée* pour les deux approches TD et BR (et elle s'étend même immédiatement à de potentielles nouvelles approches qui utiliseraient d'autres choix du paramètre X), tandis que dans (Schoknecht, 2002) et (Yu et Bertsekas, 2010), les résultats sont prouvés indépendemment pour chaque méthode.

2.4 Comparaison empirique

Pour approfondir la comparaison des approches TD et BR, nous avons effectué quelques simulations numériques. Nous considérons des espaces d'états de dimensions $n = 2, 3, .., 30$. Pour chaque n, nous considérons des projections sur des espaces de dimensions $k = 1, 2, .., n$.

8. C'est une seminorme car la matrice Q_{TD} est en général seulement semi-définie positive (étant donné que $\Phi\Phi'$ a un rang au plus d, qui est strictement plus petit que n). La projection correspondante est néanmoins bien définie (dans le sens "chaque point a exactement un projeté") dès lors que $\mathrm{Im}\,(\Phi) \cap \{x; \|x\|_{Q_{TD}} = 0\} = \{0\}$.

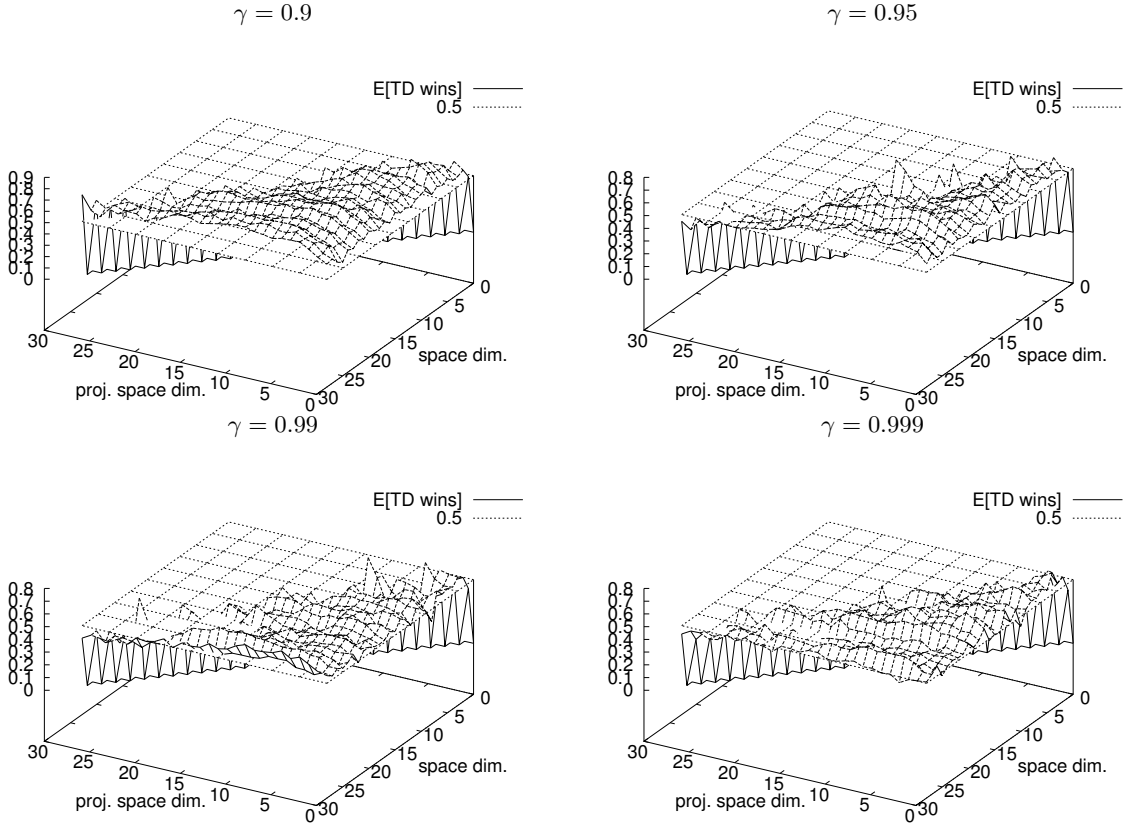


FIGURE 2.3 – Proportion de fois où l'approche TD est meilleure.

Pour chaque couple (n, k), nous générons aléatoirement 20 projections (les entrées de la matrice Φ, et de la mesure μ qui caractérisent cette projection sont des variables aléatoires uniformes i.i.d. sur $[0, 1]$) et 20 chaînes de Markov : pour chaque état i, il y a une probabilité p_i (choisie uniformément dans $[0, 1]$) d'aller dans l'état $i+1$ et une probabilité $1-p_i$ de rester dans i, l'état n étant absorbant. La récompense est un vecteur aléatoire (entrées également uniformes i.i.d sur $[0, 1]$). Pour les 20×20 combinaisons de projections/chaînes de Markov, nous calculons la vraie valeur v, la meilleure projection $\hat{v}_{opt}$ selon la norme $\|\cdot\|_\mu$, le point fixe $\hat{v}_{TD}$ de l'approche TD, et le resultat $\hat{v}_{BR}$ de l'approximation BR. Nous en déduisons la meilleure erreur $e = \|v - \hat{v}_{opt}\|_\mu$, et les erreurs d'approximations $e_{TD} = \|v - \hat{v}_{TD}\|_\mu$ et $e_{BR} = \|v - \hat{v}_{BR}\|_\mu$ des deux approches. Nous calculons également les bornes théoriques du théorème 2.5, notées ici b_{TD} et b_{BR}. Chacune des ces expériences est effectuée pour 4 valeurs du facteur d'actualisation γ : 0.9, 0.95, 0.99, et 0.999.

A partir des données brutes obtenues par ces expériences, nous avons effectué, pour chaque couple (n, k), des statistiques que nous décrivons maintenant. Tous les graphes que nous présentons montrent la dimension de l'espace n et celle de la projection k respectivement sur les axes x et y. A chaque figure, nous présentons les résultats pour les 4 valeurs du facteurs d'actualisation. La figure 2.3 donne la proportion de problèmes où la méthode TD renvoie une meilleure approximation que la méthode BR ($\mathbb{E}[\mathbb{1}_{e_{TD}<e_{BR}}]$). Cette proportion est systématiquement plus grande que $\frac{1}{2}$, ce qui signifie que le point fixe TD est généralement meilleur que la solution BR. La figure 2.4 donne la proportion du nombre de fois où les bornes du théorème 2.5 permettent de prédire correctement laquelle des deux méthodes est la mieux ($\mathbb{E}[\mathbb{1}_{[e_{TD}<e_{BR}]=[b_{TD}<b_{BR}]}]$). Sauf lorsque l'espace de projection est de grande dimension (et donc très proche de l'espace initial), ces bornes n'ont pas l'air suffisamment fines pour prédire correctement la meilleure méthode. La figure 2.5 montre $\mathbb{E}[e_{TD}/e_{BR}]$, l'espérance du rapport entre les deux erreurs d'approximation.

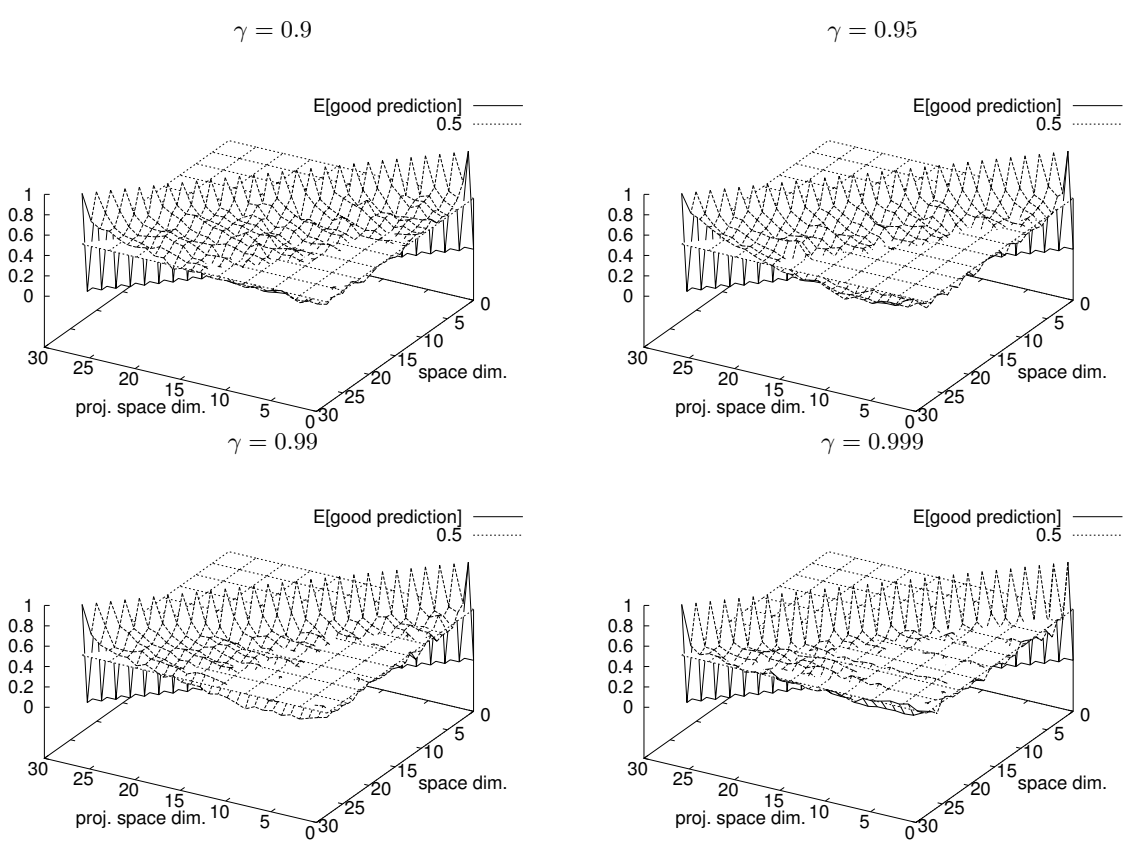


FIGURE 2.4 – Proportions du nombre de bonnes predictions de la meilleure méthode via le Théorème 2.5

On observe qu'en général, cette espérance est plus grande que 1, c'est-à-dire qu'en moyenne l'approche BR est meilleure que l'approche TD. De prime abord, ceci peut paraître contradictoire avec notre interprétation de la figure 2.3 ; l'explication est la suivante : lorsque la minimisation BR est meilleure que TD, c'est par une plus grande marge que lorsque c'est le contraire. Nous pensons que cela correspond aux situations où TD est instable numériquement. Pour confirmer ce point, la figure 2.6 montre, séparément pour chaque méthode, l'espérance de l'erreur relative d'approximation par rapport à la meilleure approximation ($\mathbb{E}[e_{TD}/e]$ et $\mathbb{E}[[e_{BR}/e]$). Sur toutes les surfaces correspondant à TD, on peut voir de nombreux pics (correspondant aux instabilités numériques), tandis que les surfaces pour BR sont beaucoup plus lisses.

2.5 Bilan et perspectives

Dans ce chapitre, nous avons présenté les méthodes du point fixe TD et de la minimisation BR pour approcher la valeur d'une politique dans un MDP. Nous avons décrit en détail deux exemples (exemples 2.1 et 2.2) originaux. Dans le premier, l'approche BR est systématiquement meilleure, tandis que dans le second (qui généralise l'esprit de l'exemple de Sutton *et al.* (2009)) le problème est mieux approché par TD. Le théorème 2.3 souligne une relation étroite entre les critères optimisés par les deux méthodes : il montre que la minimisation effectuée par l'approche BR implique la minimisation de l'erreur TD ainsi que d'un terme supplémentaire d'*adéquation*, qui apparaît crucial pour la stabilité numérique.

La principale contribution, énoncée au théorème 2.4, fournit un point de vue original pour comparer les deux méthodes de projections. Ainsi, les deux approches peuvent être toutes deux

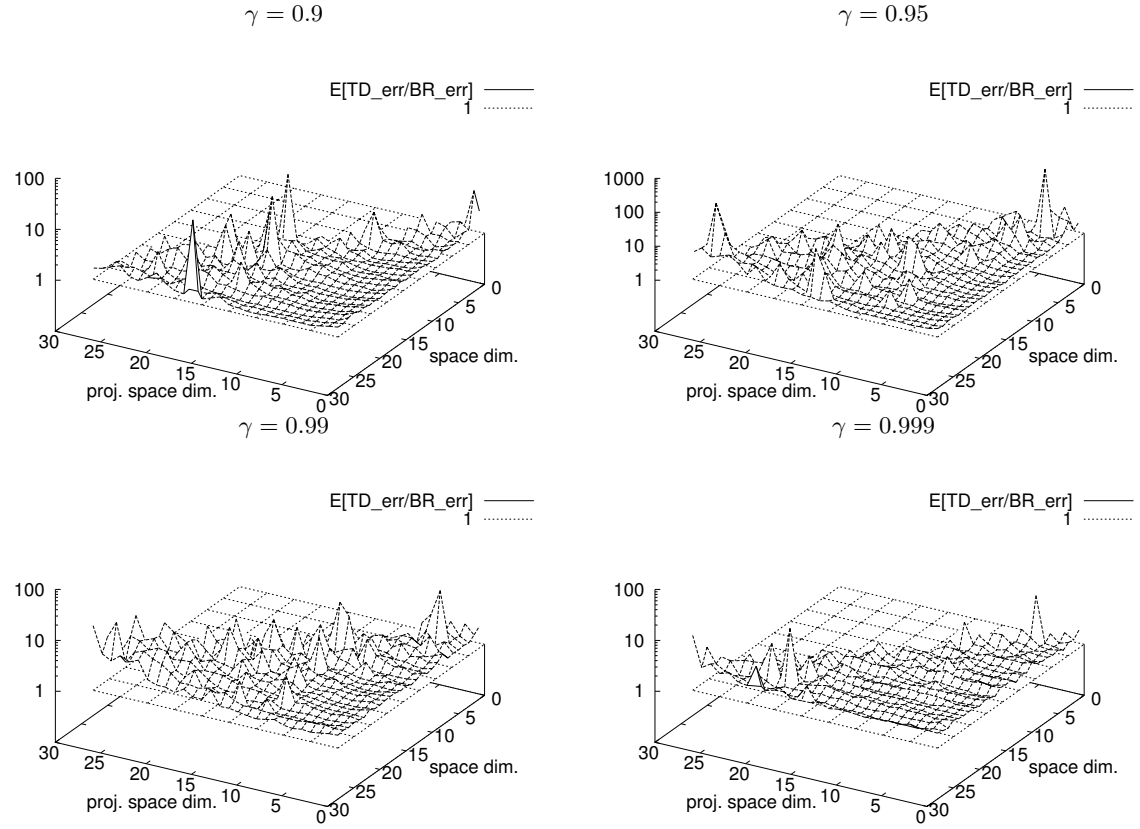


FIGURE 2.5 – Espérance de e_{TD}/e_{BR}.

vues comme des méthodes de point fixe (ce fait était à notre connaissance nouveau pour BR). De plus, les solutions de ces deux méthodes sont des projections obliques de la vraie valeur v (ceci était également à notre connaissance nouveau). Finalement, cette caractérisation permet de dériver des bornes d'erreurs optimales (théorème 2.5). Notre analyse est étroitement liée à celles de Schoknecht (2002) et Yu et Bertsekas (2010) ; en comparaison, notre approche est unifiée, et permet une dérivation beaucoup plus simple des bornes d'erreur.

Concernant la question de choisir en pratique parmi les deux approches, l'analyse suggère que l'approche BR est la plus *raisonnable* des deux, car c'est la seule à être assortie d'une performance de garantie. Empiriquement, nous pouvons faire les constats suivants :

1. la méthode TD renvoie souvent un meilleur résultat que BR ;
2. l'approche TD est numériquement instable, et renvoie parfois une très mauvaise approximation ;
3. En moyenne, BR est meilleure que TD.

En d'autres termes, on pourra retenir que l'approche du point fixe TD est plus risquée que la minimisation BR.

Une observation, un peu décevante, est que les bornes du théorème 2.5, bien qu'optimales (parmi les bornes indépendantes de la fonction récompense), n'ont pas l'air très efficaces pour décider *a priori* laquelle des deux méthodes est la meilleure en pratique (c'est ce qu'on a pu observer à la figure 2.4). Etendre l'analyse d'une manière qui permettrait de prendre en compte la fonction récompense, de même qu'exploiter la vision originale unifiée des bornes (théorèmes 2.4 et 2.5)—par exemple en interpolant entre les deux méthodes—constituent deux perspectives naturelles.

L'approche TD que nous avons considérée est un cas particulier d'une famille de méthodes, TD(λ) (Bertsekas et Tsitsiklis, 1996; Tsitsiklis et Van Roy, 1997), avec $\lambda = 0$. Une implémentation particulière de cette méthode paramétrée par λ sera d'ailleurs l'objet du prochain chapitre. Les valeurs de $\lambda > 0$ permettent de contrôler l'une des faiblesses de TD : on peut montrer que les instabilités numériques disparaissent et que l'on se rapproche de la meilleure projection $\hat{v}_{opt}$ lorsque λ tend vers 1. Etudier ce cadre plus général dans le cadre des projections obliques et dériver des bornes d'erreurs générales constitue une perspective naturelle.

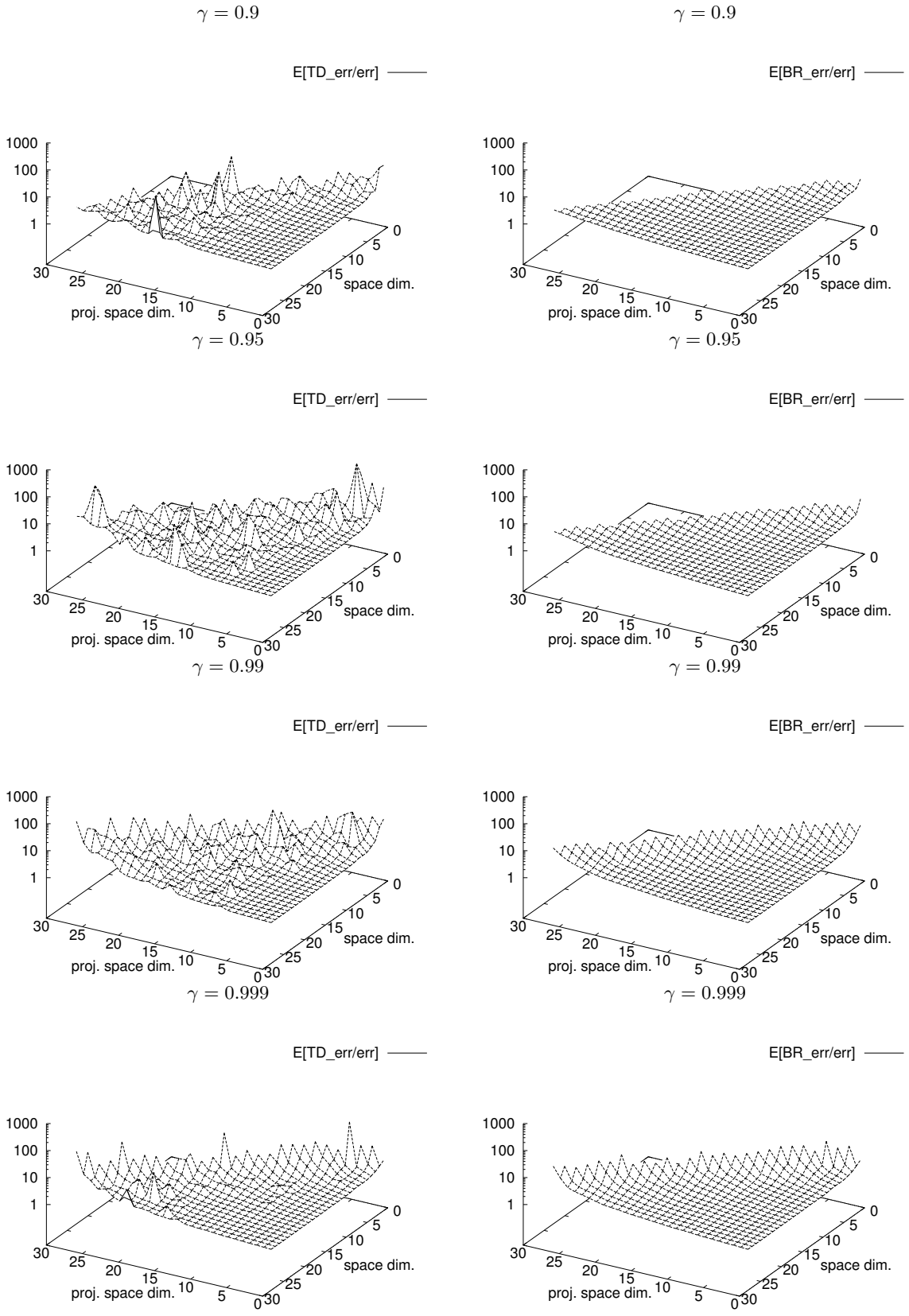


FIGURE 2.6 – Espérance de e_{TD}/e (Gauche) et espérance de e_{BR}/e (Droite).

Chapitre 3

Vitesse de convergence et borne d'erreur pour LSTD(λ)

Comme dans le chapitre précédent, nous allons ici encore nous concentrer sur l'estimation de la valeur d'une politique fixée. Nous allons considérer une généralisation de l'approche du point fixe TD ; cela permettra notamment d'illustrer la mise en œuvre pratique du calcul du point fixe d'une équation de Bellman projetée. Nous considérons l'algorithme LSTD(λ) (Least-Squares Temporal-Difference) avec traces d'éligibilité initialement proposé par Boyan (2002), où λ est un paramètre librement choisi dans $[0, 1]$. Lorsque $\lambda = 0$, le problème d'estimation que cet algorithme essaye de résoudre est le problème TD que nous avons considéré dans le chapitre précédent. Étudier le cas $\lambda \neq 0$ peut être intéressant car ce paramètre intervient dans la qualité de la borne d'erreur entre la valeur asymptotique calculée par l'algorithme et la valeur effective v. En effet, comme le montre le théorème 3.4 (reproduit à la section 3.2) dû à Tsitsiklis et Van Roy (1997), en faisant varier λ de 0 à 1, on passe de la projection oblique décrite au précédent chapitre à la (meilleure) projection orthogonale. Nedic et Bertsekas (2002) ont prouvé la convergence presque sûre de l'algorithme LSTD(λ) quand le nombre d'échantillons utilisé tend vers l'infini. Plus récemment, dans le cas $\lambda = 0$ et pour un nombre fini n d'échantillons, Lazaric *et al.* (2012) ont obtenu une borne d'erreur de l'ordre $\tilde{O}(\frac{1}{\sqrt{n}})$ [1] qui est valide avec forte probabilité. Une analyse similaire dans le cas $\lambda > 0$ n'avait, à notre connaissance, pas encore été proposée dans la littérature ; c'est l'objet de la contribution que nous décrivons dans ce chapitre, travail réalisé dans le cadre de la thèse de Manel Tagorti et publié dans (Tagorti et Scherrer, 2015).

3.1 L'algorithme LSTD(λ)

Nous utilisons un certain nombre de notations communes avec celles du précédent chapitre ; nous les rappelons ici par souci de clarté. La principale différence ici sera qu'on utilisera la notation n pour désigner le nombre d'échantillons utilisés par l'algorithme. Nous aurons de plus besoin d'un certain nombre d'hypothèses techniques que nous décrivons maintenant. Nous rappelons que comme dans le reste du manuscrit concernant les problèmes de grande taille, on considère le cas d'un espace d'états fini ou dénombrable [2]. On suppose que la chaîne de Markov (correspondant à la politique qu'on souhaite évaluer) est ergodique [3]. Par conséquent elle admet

1. La notation $f(n) = \tilde{O}(g(n))$ signifie $f(n) = O(g(n) \log^k g(n))$ pour $k \geq 0$.

2. On se restreint ici au cas fini/dénombrable essentiellement car ceci facilite la présentation de notre analyse. Une extension au cas continu est à notre avis possible moyennant quelques hypothèses techniques supplémentaires.

3. Dans le cas dénombrable, une chaîne de Markov est ergodique si et seulement si elle est irréductible et apériodique, c'est-à-dire si et seulement si : $\forall (x, y) \in X^2,\ \exists n_0,\ \forall n \geq n_0,\ P^n(x, y) > 0$.

une unique mesure invariante qu'on notera μ. Pour tout $K \in \mathbb{R}$, on note $\mathcal{B}(X, K)$ l'ensemble des fonctions mesurables définies sur X et majorées en valeur absolue par K. On suppose ici que la fonction récompense r est dans $\mathcal{B}(X, R_{\max})$ pour $R_{\max} \in \mathbb{R}$. On souhaite ici encore approcher la solution du système linéaire $v = Tv$ où T est l'opérateur de Bellman affine caractérisant la valeur de la politique. Il est clair que $v \in \mathcal{B}(X, V_{\max})$ avec $V_{\max} = \frac{R_{\max}}{1-\gamma}$. On souhaite approcher v par une paramétrisation linéaire $\hat{v} = \Phi\theta$, avec la matrice de *bases* Φ de dimension $|X| \times d$ et $\theta \in \mathbb{R}^d$. On suppose que, pour tout $j \in \{1, ..., d\}$, la *fonction de base* $\phi_j : X \to \mathbb{R}$ appartient à $\mathcal{B}(X, L)$, pour un L fini. Pour tout $x \in X$, on note $\phi(x) = (\phi_1(x), ..., \phi_d(x))'$ le vecteur donnant les d valeurs des fonctions de base en l'état x. On fera ici encore l'hypothèse suivante.

Hypothèse 3.1. *Les fonctions de base $(\phi_j)_{j\in\{1,...,d\}}$ sont linéairement indépendantes.*

Comme dans le chapitre précédent, on considère la projection Π sur l'espace engendré $\mathrm{Im}\,(\Phi)$ suivant la norme quadratique $\|.\|_\mu$:

$$\|v\|_\mu = \sqrt{\sum_{x\in X} \mu(x)v(x)^2}.$$

La matrice de cette projection Π a la forme analytique suivante :

$$\Pi = \Phi(\Phi' D_\mu \Phi)^{-1}\Phi' D_\mu, \tag{3.1}$$

où D_μ est la matrice diagonale composée des éléments $\mu(i)$.

Le but de l'algorithme LSTD(λ) est de calculer la solution de l'équation $v = \Pi T^\lambda v$, où l'opérateur T^λ est défini comme la moyenne géométrique de paramètre λ des puissances T^i de l'opérateur de Bellman T :

$$\forall\lambda \in (0,1),\ \forall v,\ T^\lambda v = (1-\lambda)\sum_{i=0}^{\infty} \lambda^i T^{i+1} v. \tag{3.2}$$

Cet opérateur est identique à celui, présenté en introduction, qui intervient dans la description de l'algorithme λ-Itérations sur les politiques—équation (2) page 9. Dans le cas $\lambda = 0$, on a $T^\lambda = T$. On peut montrer que l'opérateur T^λ est contractant de module $\frac{(1-\lambda)\gamma}{1-\lambda\gamma} \leq \gamma$ pour la norme $\|\cdot\|_\mu$; en effet, comme T^i est affine, et vu que $\|P\|_\mu = 1$ étant donné que μ est la distribution stationnaire du processus (Tsitsiklis et Van Roy, 1997; Nedic et Bertsekas, 2002), pour tous vecteurs u et v, on a

$$\begin{aligned}
\|T^\lambda u - T^\lambda v\|_\mu &\leq (1-\lambda)\|\sum_{i=0}^{\infty} \lambda^i (T^{i+1}u - T^{i+1}v)\|_\mu \\
&= (1-\lambda)\|\sum_{i=0}^{\infty} \lambda^i (\gamma^{i+1}P^{i+1}u - \gamma^{i+1}P^{i+1}v)\|_\mu \\
&\leq (1-\lambda)\sum_{i=0}^{\infty} \lambda^i \gamma^{i+1}\|u - v\|_\mu \\
&= \frac{(1-\lambda)\gamma}{1-\lambda\gamma}\|u - v\|_\mu.
\end{aligned}$$

On sait que $\|\Pi\|_\mu = 1$ (Tsitsiklis et Van Roy, 1997), et on a par conséquent $\|\Pi T^\lambda\|_\mu < 1$. Par le théorème du point fixe de Banach, l'équation $v = \Pi T^\lambda v$ admet une et une seule solution, qu'on notera $v_{LSTD(\lambda)}$, étant donné que c'est la valeur vers laquelle l'algorithme LSTD(λ)

converge asymptotiquement (Nedic et Bertsekas, 2002). Puisque $v_{LSTD(\lambda)}$ appartient au sous-espace $\mathrm{Im}\,(\Phi)$, il existe $\theta \in \mathbb{R}^d$ tel que :

$$v_{LSTD(\lambda)} = \Phi\theta = \Pi T^\lambda \Phi\theta.$$

En remplaçant Π et T^λ par leurs expressions respectives—équations (3.1) et (3.2)—on peut montrer (Nedic et Bertsekas, 2002) que résoudre $v = \Pi T^\lambda v$ revient à résoudre l'équation $A\theta = b$, avec pour tout i,

$$A = \Phi' D_\mu (I - \gamma P)(I - \lambda\gamma P)^{-1}\Phi = \mathbb{E}_{X_{-\infty}\sim\mu}\left[\sum_{k=-\infty}^{i} (\gamma\lambda)^{i-k}\phi(X_k)(\phi(X_i) - \gamma\phi(X_{i+1}))'\right] \tag{3.3}$$

$$\text{et} \quad b = \Phi' D_\mu (I - \gamma\lambda P)^{-1} r = \mathbb{E}_{X_{-\infty}\sim\mu}\left[\sum_{k=-\infty}^{i} (\gamma\lambda)^{i-k}\phi(X_k) r(X_i)\right], \tag{3.4}$$

Pour tout x, $\phi(x)$ est de dimension d, donc la matrice A est une matrice de taille $d \times d$ et b un vecteur de taille d. Sous l'hypothèse 3.1, on peut montrer que la matrice A est inversible (Nedic et Bertsekas, 2002) et donc $v_{LSTD(\lambda)} = \Phi A^{-1} b$ est bien défini.

On va maintenant décrire plus précisément l'algorithme LSTD(λ). Étant donnée une trajectoire $X_1, ..., X_n$ générée par la chaîne de Markov, les expressions respectives de A et b sous forme d'espérance—équations (3.3) et (3.4)—suggèrent de construire les estimateurs suivants :

$$\begin{aligned} \hat{A} &= \frac{1}{n-1}\sum_{i=1}^{n-1} z_i(\phi(X_i) - \gamma\phi(X_{i+1}))' \\ \text{et} \quad \hat{b} &= \frac{1}{n-1}\sum_{i=1}^{n-1} z_i r(X_i), \\ \text{où} \quad z_i &= \sum_{k=1}^{i} (\lambda\gamma)^{i-k}\phi(X_k) \end{aligned} \tag{3.5}$$

est communément appelée la *trace d'éligibilité*. L'algorithme LSTD(λ) renvoie alors l'estimation (pour un nombre d'échantillons fini) $\hat{v}_{LSTD(\lambda)} = \Phi\hat{\theta}$ de $v_{LSTD(\lambda)}$ avec [4] $\hat{\theta} = \hat{A}^{-1}\hat{b}$. Nedic et Bertsekas (2002) ont montré, en utilisant une variante de la loi des grands nombres, la convergence presque sûre de $\hat{A}$ et $\hat{b}$ respectivement vers A et b. Ceci implique la convergence presque sûre de $\hat{v}_{LSTD(\lambda)}$ vers $v_{LSTD(\lambda)}$. Le résultat principal présenté dans ce chapitre et publié avec Manel Tagorti dans (Tagorti et Scherrer, 2015) est d'approfondir cette analyse : nous allons dériver une borne sur la vitesse de convergence de $\hat{v}_{LSTD(\lambda)}$ vers $v_{LSTD(\lambda)}$, et en déduire une borne sur l'erreur d'approximation globale $\|\hat{v}_{LSTD(\lambda)} - v\|_\mu$ de l'algorithme.

3.2 Résultat principal

Cette section contient notre résultat principal. Une hypothèse clé de notre analyse consiste à supposer la chaîne de Markov β-mélangeante [5].

4. On verra dans le théorème 3.3 que $\hat{A}$ est inversible avec forte probabilité à partir d'un certain nombre d'échantillons.

5. Une chaîne de Markov stationnaire et ergodique est *toujours* β-mélangeante.

Hypothèse 3.2. *Le processus $(X_n)_{n\geq 1}$ est β-mélangeant, c'est-à-dire que son $i^{ème}$ coefficient*

$$\beta_i = \sup_{t\geq 1} \mathbb{E}\left[\sup_{B\in\sigma(X_{t+i}^{\infty})} \left|P(B|\sigma(X_1^t)) - P(B)\right|\right]$$

tend vers 0 quand i tend vers l'infini, avec $X_l^j = \{X_l, ..., X_j\}$ pour $j \geq l$ et $\sigma(X_l^j)$ la tribu engendrée par X_l^j. De plus, le processus $(X_n)_{n\geq 1}$ est exponentiellement β-mélangeant de paramètres $\overline{\beta} > 0$, $b > 0$, et $\kappa > 0$ dans le sens suivant : $\beta_i \leq \overline{\beta} e^{-bi^{\kappa}}$.

Intuitivement le coefficient β_i mesure le degré de dépendance entre les échantillons de la séquence séparés par i pas de temps (plus le coefficient est petit, plus les échantillons sont indépendants). On va maintenant énoncer le théorème principal qui fournit la vitesse de convergence de l'algorithme LSTD(λ).

Théorème 3.3. *On fait les hypothèses 3.1 et 3.2 et on suppose que $X_1 \sim \mu$. Pour tout $n \geq 1$ et $\delta \in (0,1)$, on définit les fonctions :*

$$I(n,\delta) = 32\Lambda(n,\delta)\max\left\{\frac{\Lambda(n,\delta)}{b}, 1\right\}^{\frac{1}{\kappa}} \quad avec \quad \Lambda(n,\delta) = 2\left(\log\left(\frac{8n^2}{\delta}\right) + \log(\max\{4e^2, n\overline{\beta}\})\right),$$

et l'entier strictement positif :

$$m_n^{\lambda} = \left\lceil \frac{\log(n-1)}{\log\left(\frac{1}{\lambda\gamma}\right)} \right\rceil.$$

Pour tout δ, soit $n_0(\delta)$ le plus petit entier tel que pour tout $n \geq n_0(\delta)$,

$$\frac{2dL^2}{(1-\gamma)\nu}\left[\frac{2}{\sqrt{n-1}}\sqrt{(m_n^{\lambda}+1)\, I(n-1,\delta)} + \frac{1}{(n-1)(1-\lambda\gamma)} + \frac{2}{n-1}m_n^{\lambda}\right] < 1 \tag{3.6}$$

où ν est la plus petite valeur propre de la matrice de Gram $\Phi' D_{\mu}\Phi$. Alors, pour tout δ, avec une probabilité supérieure ou égale à $1-\delta$, pour tout $n \geq n_0(\delta)$, $\hat{A}$ est inversible et

$$\|v_{LSTD(\lambda)} - \hat{v}_{LSTD(\lambda)}\|_{\mu} \leq \frac{4V_{max}dL^2}{\sqrt{n-1}(1-\gamma)\nu}\sqrt{(m_n^{\lambda}+1)\, I(n-1,\delta)} + h(n,\delta)$$

avec $h(n,\delta) = O\left(\frac{1}{n}\log\frac{1}{\delta}\right)$.

La constante ν est strictement positive sous l'hypothèse 3.1. Pour tout δ, il est clair que l'entier $n_0(\delta)$ existe et est fini puisque le terme à gauche de l'équation (3.6) tend vers 0 quand n tend vers l'infini. Comme m_n^{λ} et $I(n-1,\delta)$ sont $\tilde{O}(1)$, on peut déduire que LSTD(λ) estime $v_{LSTD(\lambda)}$ avec une vitesse de l'ordre $\tilde{O}\left(\frac{1}{\sqrt{n}}\right)$. Comme la fonction $\lambda \mapsto m_n^{\lambda}$ est croissante, notre borne sur la vitesse de convergence se détériore lorsque λ augmente. Cette détérioration est compensée par le fait que la qualité asymptotique de $v_{LSTD(\lambda)}$ s'améliore lorsqu'on augmente le paramètre λ, comme le montre le résultat suivant de la littérature.

Théorème 3.4 (Tsitsiklis et Van Roy (1997))**.** *L'erreur d'approximation vérifie*[6] *:*

$$\|v - v_{LSTD(\lambda)}\|_{\mu} \leq \frac{1-\lambda\gamma}{1-\gamma}\|v - \Pi v\|_{\mu}.$$

6. Cette borne peut être améliorée, comme l'a suggéré V. Papavassilou (Tsitsiklis et Van Roy, 1997) ; en utilisant le théorème de Pythagore, on obtient

$$\|v - v_{LSTD(\lambda)}\|_{\mu} \leq \frac{1-\lambda\gamma}{\sqrt{(1-\gamma)(1+\gamma-2\lambda\gamma)}}\|v - \Pi v\|_{\mu}.$$

Par souci de simplicité, nous garderons la forme du théoreme 3.4.

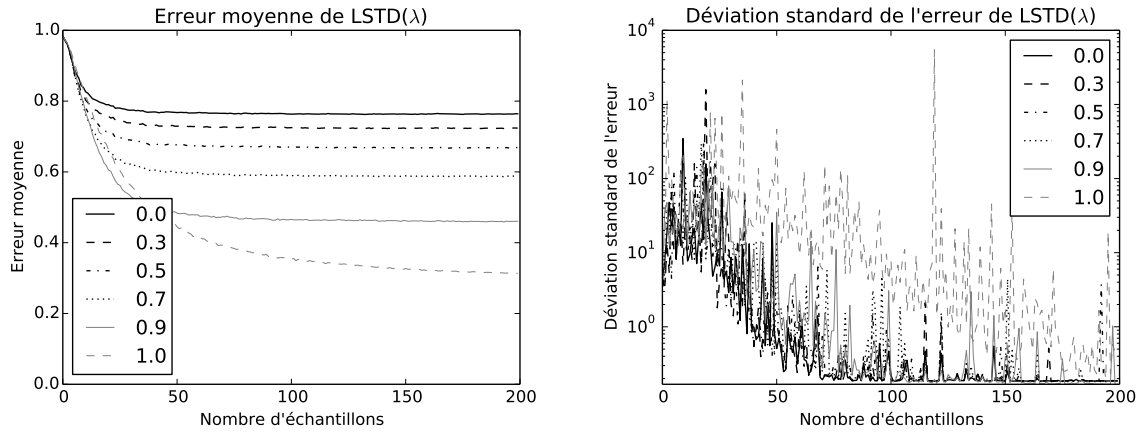


FIGURE 3.1 – **Courbes d'apprentissage pour différentes valeurs de λ.** On génère 10000 MDPs Garnet aléatoires (Archibald *et al.*, 1995) avec 15 états, un facteur de branchement 2, des récompenses uniformes et aléatoires avec $\gamma = 0.99$. Pour chaque MDP, on génère une base aléatoire de dimension 4 (en prenant des matrices aléatoires avec des entrées uniformes et aléatoires). Pour toutes ces valeurs de $\lambda \in \{0.0, 0.3, 0.5, 0.7, 0.9, 1.0\}$, on montre (à gauche) la moyenne de l'erreur et (à droite) la déviation standard de cette erreur en fonction du nombre d'échantillons.

En particulier lorsque $\lambda = 1$, on a $\frac{1-\lambda\gamma}{1-\gamma} = 1$ et l'on en déduit que LSTD(1) estime la projection orthogonale Πv de v. En utilisant l'inégalité triangulaire, on déduit des théorèmes 3.3 et 3.4 une borne sur l'erreur globale.

Corollaire 3.5. *Sous les hypothèses et les notations du théorème 3.3, pour tout δ, avec une probabilité $1-\delta$, pour tout $n \geq n_0(\delta)$, l'erreur globale de LSTD(λ) vérifie :*

$$\|v - \hat{v}_{LSTD(\lambda)}\|_\mu \leq \frac{1-\lambda\gamma}{1-\gamma}\|v - \Pi v\|_\mu + \frac{4V_{max}dL^2}{\nu\sqrt{n-1}(1-\gamma)}\sqrt{(m_n^\lambda + 1)\, I(n-1,\delta)} + h(n,\delta).$$

Cette borne requiert un nombre suffisamment grand ($n \geq n_0(\delta)$) d'échantillons. Pour une valeur fixe de δ, ce nombre augmente lorsque λ augmente. Par ailleurs, lorsque δ tend vers 0, $n_0(\delta)$ tend vers l'infini. L'existence d'une telle condition n'est pas surprenante dans la mesure où l'on s'intéresse à une version non-régularisée de LSTD(λ), ce qui fait que la matrice $\hat{A}$ peut être singulière pour un n ou un δ trop petit.

Comme il a déjà été mentionné précédemment, la valeur $\lambda = 1$ minimise la borne sur l'erreur d'approximation $\|v - v_{LSTD(\lambda)}\|_\mu$ (le premier terme à droite dans le corollaire 3.5) alors que la valeur $\lambda = 0$ minimise la borne sur l'erreur d'estimation $\|v_{LSTD(\lambda)} - \hat{v}_{LSTD(\lambda)}\|_\mu$ (le second terme). Pour tous δ et $n \geq n_0(\delta)$, il existe une valeur optimale λ^* du paramètre qui minimise l'erreur globale, faisant un compromis entre ces deux erreurs. Lorsque le nombre d'échantillons n tend vers l'infini, la valeur optimale λ^* tend vers 1. En général cependant, le choix optimal λ^* depend aussi bien des paramètres du processus β-mélangeant (b, κ et $\overline{\beta}$) que de la qualité de l'espace de projection $\|v - \Pi v\|_\mu$, qui sont des quantités inconnues en pratique. La figure 3.1 illustre via quelques simulations numériques l'influence de λ et n sur l'erreur d'approximation globale. Empiriquement, la meilleure valeur de λ semble bien être une fonction monotone du nombre d'échantillons n, qui tend vers 1 asymptotiquement.

3.3 Principales idées de la preuve

Dans cette section, j'esquisse les grandes lignes justifiant les résultats de la section précédente. Les détails des arguments sont donnés dans l'article (Tagorti et Scherrer, 2015). Tout d'abord, nous commençons par dériver (via des arguments d'algèbre linéaire) un lemme sur la sensibilité de la solution $v_{LSTD(\lambda)}$ à une déviation déterministe des estimés $\hat{A}$ et $\hat{b}$ par rapport à leurs limites respectives A et b.

Lemme 3.6. *Soient $\epsilon_A = \hat{A} - A$, $\epsilon_b = \hat{b} - b$, et ν la plus petite valeur propre de $\Phi^T D_\mu \Phi$. Pour tout $\lambda \in (0,1)$,*

$$\|v_{LSTD(\lambda)} - \hat{v}_{LSTD(\lambda)}\|_\mu \leq \frac{1-\lambda\gamma}{(1-\gamma)\sqrt{\nu}} \|(I + \epsilon_A A^{-1})^{-1}\|_2 \|\epsilon_A \theta - \epsilon_b\|_2,$$

où $\theta = A^{-1}b$. De plus, si pour 2 réels ϵ et C on a $\|\epsilon_A\|_2 \leq \epsilon < C \leq \frac{1}{\|A^{-1}\|_2}$, alors $\hat{A}$ est inversible et

$$\|(I + \epsilon_A A^{-1})^{-1}\|_2 \leq \frac{1}{1 - \frac{\epsilon}{C}}.$$

Ce lemme suggère de contrôler les termes $\|\epsilon_A\|_2 = \|\hat{A} - A\|_2$ et $\|\epsilon_b\|_2 = \|\hat{b} - b\|_2$. Nous allons expliquer comment faire cela avec forte probabilité. Comme les deux termes $\hat{A}$ et $\hat{b}$ ont la même structure, on considère une matrice qui a la forme générale suivante :

$$\hat{G} = \frac{1}{n-1} \sum_{i=1}^{n-1} G_i \quad \text{avec} \quad G_i = z_i (\tau(X_i, X_{i+1}))^T$$

où z_i est la trace définie à l'équation (3.5) et τ est une fonction de X^2 dans $\mathbb{R}^k$. Soit $\|.\|_F$ la norme de Frobenius : pour tout $M \in \mathbb{R}^{d \times k}$, $\|M\|_F^2 = \sum_{l=1}^{d} \sum_{j=1}^{k} (M_{l,j})^2$. Le second élément de notre analyse est une inégalité de concentration pour le processus $\hat{G}$ basé sur une trace z_i définie à partir d'une chaîne de Markov β-mélangeante.

Lemme 3.7. *On fait les hypothèses 3.1 et 3.2 et on suppose que $X_1 \sim \mu$. On rappelle que $\phi = (\phi_1, \ldots, \phi_d)$ est telle que pour tout j, $\phi_j \in \mathcal{B}(X, L)$; on suppose de plus que pour tout $1 \leq j \leq d$, $\tau_j \in \mathcal{B}(X^2, L')$. Soient m_n^λ et $I(n,\delta)$ comme définis au théorème 3.3. Soit $J(n,\delta) = I(n, 4n^2\delta)$. Alors, pour tout δ dans $[0,1]$, avec probabilité au moins $1-\delta$,*

$$\left\| \frac{1}{n-1} \sum_{i=1}^{n-1} G_i - \frac{1}{n-1} \sum_{i=1}^{n-1} \mathbb{E}[G_i] \right\|_2 \leq \frac{2\sqrt{d \times k} L L'}{(1-\lambda\gamma)\sqrt{n-1}} \sqrt{(m_n^\lambda + 1)\, J(n-1, \delta)} + \epsilon(n),$$

avec $\epsilon(n) = 2m_n^\lambda \frac{\sqrt{d \times k} L L'}{(n-1)(1-\lambda\gamma)}$.

La preuve de ce résultat est assez technique. Il y a deux difficultés concernant les estimés G_i utilisés pour calculer G : 1) G_i est une fonction $\sigma(X^{i+1})$-mesurable du vecteur non-stationnaire $(X_1, \ldots, X_{i+1})$ et est par conséquent *non-stationnaire* ; 2) Pour tout i, les G_i sont calculés à partir d'une unique trajectoire de la chaîne de Markov et sont statistiquement *fortement dépendants.* Pour gérer la première difficulté (la non-stationarité), on considère une trace tronquée avec profondeur m

$$z_i^m = \sum_{k=\max(i-m+1,1)}^{i} (\lambda\gamma)^{i-k} \phi(X_k),$$

et on approche $\hat{G}$ par le processus $\hat{G}^m$ défini par :

$$\hat{G}^m = \frac{1}{n-1}\sum_{i=1}^{n-1} G_i^m, \quad \text{avec} \quad G_i^m = z_i^m(\tau(X_i, X_{i+1}))^T.$$

En effet, G_i^m est maintenant une fonction $\sigma(X^{m+1})$-mesurable du vecteur stationnaire $Z_i = (X_{i-m+1}, X_{i-m+2}, \ldots, X_{i+1})$, ce vecteur étant stationnaire dans la mesure où $X_1 \sim \mu$. Pour gérer le deuxième difficulté (la dépendance des échantillons), pour n'importe quelle profondeur de troncature m de la trace, on utilise l'hypothèse de β-mélange pour transformer les échantillons G_i^m en blocs d'échantillons indépendants en utilisant la "technique de blocs" de Yu (1994) d'une manière similaire à ce qui a été fait par Lazaric *et al.* (2012) pour LSTD(0). On conclut alors en utilisant une inégalité de concentration de la littérature pour un processus basé sur des échantillons i.i.d. En plus des traces tronquées à profondeur m, un ingrédient spécifique de notre analyse de LSTD(λ) est que nous devons montrer que le processus $(Z_i)_{i\geq 1} = (X_{i-m+1}, X_{i-m+2}, \ldots, X_{i+1})_{i\geq 1}$, à partir duquel le processus tronqué G_i^m est défini, hérite de la propriété de β-mélange du processus originel $(X_i)_{i\geq 1}$; c'est l'objet du lemme ci-dessous.

Lemme 3.8. *Si $(X_n)_{n\geq 1}$ est un processus β-mélangeant, alors $(Z_n)_{n\geq m} = (X_{n-m+1}, X_{n-m+2}, \ldots, X_{n+1})_{n\geq m}$ est un processus β-mélangeant tel que le $i^{ième}$ coefficient β_i^Z satisfait $\beta_i^Z \leq \beta_{i-m}^X$.*

Le lemme 3.7 s'obtient alors en choisissant une profondeur de trace $m = m_n^\lambda$, ce qui permet de faire en sorte que la distance entre $\hat{G}$ et $\hat{G}^m$ est bornée par $\epsilon(n)$ (comme défini dans le lemme), et est donc négligeable par rapport à l'analyse de déviation obtenue via la "technique de blocs" de Yu (1994).

La fin de la preuve du théorème 3.3 se fait en appliquant l'inégalité de concentration du lemme 3.7 aux quantités d'intérêt du lemme 3.6, c'est-à-dire à $\|\epsilon_A\|_2$ et $\|\epsilon_A\theta - \epsilon_b\|_2$. Le passage du terme $J(n,\delta) = I(n, 4n^2\delta)$ du lemme 3.7 à $I(n,\delta)$ dans le theorème 3.3 se fait en bornant la probabilité de la réunion des événements "$\|\epsilon_A\|_2$ et $\|\epsilon_A\theta - \epsilon_b\|_2$ sont suffisamment petits après n échantillons" sur tous les valeurs de n. On notera en particulier que la condition sur le nombre minimal d'échantillons $n_0(\delta)$ provient de la deuxième partie du lemme 3.6.

3.4 Bilan et perspectives

Nous avons étudié la vitesse de convergence de l'algorithme LSTD(λ) en termes du nombre d'échantillons n et du paramètre λ. Le théorème 3.3 montre, sous une hypothèse de β-mélange, que la vitesse de convergence est de l'ordre $\tilde{O}(\frac{1}{\sqrt{n}})$. Le corollaire 3.5 est, à notre connaissance, le premier résultat analytique qui rend compte de l'influence du paramètre λ pour un algorithme approché de type *différences temporelles* par rapport à la qualité du choix de l'espace linéaire choisi pour approcher la valeur et du nombre d'échantillons. Les études précédentes du rôle du paramètre λ étaient à notre connaissance empiriques (Sutton et Barto, 1998) ou réduites à une représentation exacte de la fonction valeur (Kearns et Singh, 2000).

La forme de notre résultat au corollaire 3.5 est légèrement plus forte que celle de Lazaric *et al.* (2012) dans le cas $\lambda = 0$: pour une propriété P, notre résultat est de la forme

$$''\forall\delta, \exists n_0(\delta), \forall n \geq n_0(\delta),\ P \text{ est vraie avec probabilité } 1-\delta''$$

alors que celui de Lazaric *et al.* (2012) est de la forme

$$''\forall n,\ \forall\delta,\ P \text{ est vraie avec probabilité } 1-\delta''.$$

En particulier, notre formulation a l'avantage d'impliquer le précédent résultat d'analyse de LSTD(λ) dérivé par Nedic et Bertsekas (2002) : la convergence presque sûre de $\hat{v}_{LSTD(\lambda)}$ vers $v_{LSTD(\lambda)}$.

Sous des hypothèses similaires, la borne obtenue par Lazaric *et al.* (2012) dans le cas particulier $\lambda = 0$, a la forme suivante :

$$\|\tilde{v}_{LSTD(0)} - v\|_\mu \leq \frac{4\sqrt{2}}{1-\gamma}\|v - \Pi v\|_\mu + O\left(\sqrt{\frac{d\log d}{\nu n}}\right),$$

où $\tilde{v}_{LSTD(0)}$ est la troncature (avec seuils $\{-V_{\max}, V_{\max}\}$) de $\hat{v}_{LSTD(0)}$. Avec notre analyse, nous obtenons pour $\lambda = 0$:

$$\|\hat{v}_{LSTD(0)} - v\|_\mu \leq \frac{1}{1-\gamma}\|v - \Pi v\|_\mu + \tilde{O}\left(\frac{d}{\nu\sqrt{n}}\right).$$

D'un côté, le terme correspondant à l'erreur d'approximation est meilleur d'un facteur $4\sqrt{2}$ avec notre analyse : notre borne est en particulier meilleure asymptotiquement. Contrairement à notre approche, l'analyse de Lazaric *et al.* (2012) n'implique pas la convergence de $\hat{v}_{LSTD(0)}$ vers $v_{LSTD(0)}$; leur arguments, basé sur un modèle de *régression avec design Markovien*, consiste à *directement* borner l'erreur globale. D'un autre côté, notre borne dépend linéairement de la dimension d de l'espace sur lequel on projette et de $\frac{1}{\nu}$ tandis que le résultat obtenu par Lazaric *et al.* (2012) a une dépendance en $O\left(\sqrt{d\log d/(n\nu)}\right)$; notre borne semble donc sous-optimale par rapport à d et ν. Un élément technique, mentionné plus haut, pour expliquer cette différence, est que Lazaric *et al.* (2012) considèrent une version tronquée de $v_{LSTD(0)}$. En effet, une lecture attentive de notre preuve dans (Tagorti et Scherrer, 2015) montre que le terme supplémentaire $\sqrt{d/\nu}$ vient d'une borne uniforme en x de $v_{LSTD(\lambda)}(x)$. Il n'est pas clair qu'on puisse utiliser le même genre d'astuce avec notre approche, mais cela constitue une piste à creuser.

Une condition critique dans l'analyse de LSTD(0) par Lazaric *et al.* (2012) est que le terme de bruit, dans leur modèle de *régression avec design Markovien*, est une différence de martingales par rapport à la filtration associée à la chaîne de Markov. Dès que $\lambda > 0$, cette propriété cesse d'être vraie. Nous pensons que la technique que nous avons utilisée pour prouver l'inégalité de concentration du lemme 3.7—la troncature de la trace à une profondeur m et la considération du processus $(Z_n) = (X_{i-m+1}, X_{i-m}, \ldots, X_{i+1})$ par blocs de taille m—constitue une piste pour corriger cette difficulté technique. Si cela se confirmait, il faudrait noter cependant qu'une telle extension du travail de l'approche de Lazaric *et al.* (2012) pour $\lambda > 0$ souffrirait toujours du facteur $4\sqrt{2}$ dans la borne finale.

Au sujet de la dépendence par rapport aux paramètres d et ν, il est intéressant de mentionner que la borne obtenue par Pires et Szepesvári (2012) pour une version régularisée de LSTD(0) dépend linéairement de d et $\|\theta\|_2$, qui peut être borné par $V_{\max}/\sqrt{\nu}$, soit une dépendance similaire en d mais meilleure en $\frac{1}{\nu}$. Dans Antos *et al.* (2008), la borne ne dépend pas de $\frac{1}{\nu}$ mais la vitesse de convergence est en $\tilde{O}\left(\frac{1}{n^{\frac{1}{4}}}\right)$, ce qui est plus lent que ce que nous obtenons ici. Dans un cadre de *régression avec design déterministe*—la régression pure peut être vue comme correspondant au cas où $\gamma = 0$—, les bornes correspondantes ne dépendent pas du paramètre ν (Györfi *et al.*, 2002). La question de savoir si l'on peut avoir le meilleur des mondes—la meilleure borne asymptotique sans le facteur $4\sqrt{2}$, la meilleure vitesse de convergence en n et la meilleure dépendance en d et $\frac{1}{\nu}$—constitue une perspective naturelle et intéressante.

Pires et Szepesvári (2012) ont considéré des versions pénalisées de systèmes linéaires estimés de manière bruitée, et expliqué comment appliquer leur approche à LSTD(0). Une telle pénalisation permet de contrôler la norme du paramètre estimé $\hat{\theta}$ dans les situations où la

matrice $\hat{A}$ est presque singulière. Ceci a l'avantage de supprimer la condition sur le nombre d'échantillons pour assurer l'inversibilité de $\hat{A}$, et permet de dériver des bornes qui sont valides pour toutes valeurs du seuil de probabilité δ et du nombre d'échantillons n (tandis que dans notre approche sans régularisation, le nombre minimum d'échantillons $n_0(\delta)$ tend vers l'infini lorsque δ tend vers 0). La pénalisation la plus simple que l'on aurait envie de rajouter à LSTD(λ) est celle où l'on rajoute un terme ρI à l'estimé $\hat{A}$ (Nedic et Bertsekas, 2002). Cela revient à résoudre le problème ci-dessous :

$$\hat{\theta}_\rho = \arg\min_\theta \left\{ \|\hat{A}\theta - \hat{b}\|_2^2 + \rho\|\theta\|_2^2 \right\}. \tag{3.7}$$

Cette forme de régularisation—erreur et pénalisation au carré—n'est pas traitée par Pires et Szepesvári (2012). Il n'est cependant pas très compliqué de suivre une approche similaire à celle décrite par Pires et Szepesvári (2012) pour borner le terme de résidu $\|A\hat{\theta}_\rho - b\|_2$. Combiné à notre analyse—nous avons toujours besoin des inégalités de concentration (lemme 3.7) et d'une partie de notre analyse de sensibilité (lemme 3.6) pour passer du résidu à une borne sur la fonction valeur—, ceci mène au résultat suivant :

Théorème 3.9. *On se place sous les hypothèses et notations du théorème 3.3 et du lemme 3.7. Pour tout $\delta \in (0,1)$ et n, on considère l'estimation $\hat{v}^{\rho_{n,\delta}}_{LSTD(\lambda)} = \Phi\hat{\theta}_\rho$ obtenue en résolvant le problème à l'équation* (3.7) *avec*

$$\rho_{n,\delta} = \frac{4dL^2}{(1-\lambda\gamma)\sqrt{n-1}}\sqrt{(m_n^\lambda+1)\,J(n-1,\delta_n)} + \frac{2dL^2}{(1-\lambda\gamma)^2(n-1)} + 2m_n^\lambda\frac{\sqrt{d\times k}LL'}{(1-\lambda\gamma)(n-1)} = O\left(\frac{\log n}{\sqrt{n}}\log\frac{1}{\delta}\right).$$

Alors, avec probabilité au moins $1-\delta$, et pour tout n,

$$\|\hat{v}^{\rho_{n,\delta}}_{LSTD(\lambda)} - v_{LSTD(\lambda)}\|_\mu \leq \frac{4V_{max}\sqrt{d}L(3+\sqrt{d}L)}{\sqrt{n-1}(1-\gamma)\sqrt{\nu}}\sqrt{(m_n^\lambda+1)\,I(n-1,\delta)} + h(n,\delta).$$

Du côté négatif, cette borne (la constante) est moins bonne que celle que nous avons dérivée dans le cas non régularisé. Du côté positif, elle a l'avantage d'être valide uniformément pour tout δ et n.

Deux perspectives intéréssantes et assez immédiates, partiellement considérées dans le manuscrit de thèse de Manel (Tagorti, 2015) sont les suivantes :

1. Etendre notre analyse au cas de politiques périodiques non-stationnaires, dans la mesure où nous verrons aux chapitres 5 et 6 que cela permet d'améliorer la borne de performance de l'algorithme itérations sur les politiques ;
2. Considérer d'autres hypothèses de mélange que le β-mélange (par exemple la notion de m-mélange).

Une question un peu plus ardue serait d'envisager une analyse similaire à celle que nous avons menée pour une version *off-policy* de LSTD(λ)—c'est-à-dire lorsqu'on suit la chaîne de Markov induite par une politique pour en évaluer une autre. La convergence d'une telle variante a été récemment prouvée par Yu (2010), mais il faut noter que le caractère *off-policy* fait perdre les propriétés de contraction des opérateurs, et l'analyse s'appuie alors sur la théorie des processus de Feller (qui vivent dans des espaces topologiques) rendant une étude en nombre d'échantillons finis vraisemblablement plus complexe.

L'algorithme LSTD(λ) n'est pas la seule manière de calculer une approximation linéaire de la fonction valeur, voire même d'estimer le point fixe d'une équation de Bellman projetée $v = \Pi T^\lambda v$. Dans le précédent chapitre, nous avons évoqué l'approche de minimisation du résidu de Bellman (BR). Avec Matthieu Geist, nous avons proposé un cadre algorithmique qui permet

de décrire—voire dans plusieurs cas de généraliser—de manière unifiée de nombreux algorithmes de la littérature. Tous ces algorithmes peuvent être vus comme mettant à jour, échantillon après échantillon, le paramètre caractérisant la position dans l'espace linéaire : après le $i^{\text{ième}}$ échantillon, le nouveau paramètre θ_i est calculé de sorte à se rapprocher du minimum de la fonction

$$\omega \mapsto \sum_{j=1}^{i} \left(\hat{T}_{j,i}^{\lambda} \hat{V}_{\xi_{ij}} - \hat{V}_{\omega}(s_j) \right)^2,$$

où $\hat{T}_{j,i}^{\lambda}$ est une approximation empirique de l'opérateur T^{λ}, et ξ_{ij} est remplacé par un élément de l'ensemble $\{\theta_i, \theta_{i-1}, \theta_j, \omega\}$. Selon le choix de ξ_{ij} et la définition précise de $\hat{T}_{j,i}^{\lambda}$ on retombe sur les algorithmes LSTD(λ) (Nedic et Bertsekas, 2002), LSPE(λ) (Nedic et Bertsekas, 2002), FPKF (Choi et Van Roy, 2006) et BRM (Engel, 2005; Geist et Pietquin, 2013) si θ_i est *exactement* le minimum de la fonction ci-dessus, et sur TD(λ) (Sutton et Barto, 1998), TDC(λ) / GQ(λ) Sutton *et al.* (2009), GTD2 (Sutton *et al.*, 2009) et gBRM (Williams et Baird, 1993) si θ_i est obtenu par un pas de type descente de gradient stochastique. Bien qu'ayant une vision algorithmique unifiée, nous n'avons pas été à même de proposer une analyse unifiée, mais seulement quelques résultats partiels. A l'exception de LSTD(λ) et LSPE(λ) qui ont été analysés par Nedic et Bertsekas (2002), la convergence de la plupart de ces algorithmes dans le cas *off-policy* reste à notre connaissance un problème ouvert, et constitue un sujet naturel pour des travaux futurs.

Chapitre 4

Itérations sur les Politiques Modifié avec approximations

Après deux chapitres sur l'estimation de la fonction valeur pour une politique fixe, nous allons revenir dans ce chapitre et ceux qui suivent sur des schémas algorithmiques de programmation dynamique approchée pour estimer la valeur et la politique *optimales* dans des problèmes de trop grande taille pour qu'on puisse envisager une résolution exacte. J'ai commencé à travailler sur ce sujet en considérant une version approchée de l'algorithme λ-Itérations sur les Politiques (mentionné dans l'introduction page 9) dans (Scherrer, 2007, 2013b) ainsi que dans le cadre de la thèse de Christophe Thiéry (Thiéry et Scherrer, 2010). Nous avons logiquement considéré ensuite la version la plus générale, Itérations sur les Politiques Optimiste (également mentionnée dans l'introduction, page 9) dans (Scherrer et Thiéry, 2010). J'ai choisi ici de décrire les travaux concernant des implémentations approchées d'Itérations sur les Politiques Modifié (MPI, pour Modified Policy Iteration) que nous avons proposées dans (Scherrer *et al.*, 2012, 2015)—avec Mohammad Ghavamzadeh, Victor Gabillon, Boris Lesner et Matthieu Geist—pour trois raisons : le schéma algorithmique est plus simple conceptuellement que celui de λ-Itérations sur les Politiques, plus intéressant dans le sens où il généralise un grande nombre d'instances algorithmiques de la littérature, et il a mené à une implémentation qui a donné de très bons résultats empiriques, que nous évoquerons à la fin de ce chapitre.

Comme nous l'avons déja évoqué dans l'introduction de ce manuscrit, MPI, intialement proposé par Puterman et Shin (1978), est une procédure itérative pour calculer la valeur et la politique optimale d'un MDP. Partant d'une valeur v_0 arbitraire, on génère la séquence de paires politiques-valeurs

$$\pi_{k+1} \leftarrow \text{un élément de } \mathcal{G}v_k \qquad \text{(étape gloutonne)} \tag{4.1}$$

$$v_{k+1} \leftarrow (T_{\pi_{k+1}})^m v_k \qquad \text{(étape d'évaluation)} \tag{4.2}$$

où l'on rappelle que $\mathcal{G}v_k$ est l'ensemble des politiques gloutonnes par rapport à v_k, T_{π_k} est l'opérateur de Bellman associé à la politique π_k, et où $m \geq 1$ est un paramètre entier libre. MPI généralise les deux algorithmes les plus connus de la programmation dynamique : Itérations sur les Valeurs (VI) et Itération sur les Politiques (PI) pour les valeurs $m = 1$ et $m = \infty$ respectivement. Lorsqu'on l'implémente de manière exacte, les itérations de MPI sont plus légères que celles de PI (d'une manière similaire à VI) et l'algorithme hérite en partie de la convergence rapide (en termes de nombre d'itérations) de PI (Puterman, 1994). Pour des problèmes de grande taille, des versions approchées de VI (AVI pour Approximate VI) et PI (API pour Approximate PI) ont été l'objet d'une riche littérature (voir par exemple Bertsekas et Tsitsiklis 1996; Szepesvári 2010). D'un côté, AVI génère une nouvelle fonction valeur en approchant l'application de l'opérateur T d'optimalité de Bellman sur la valeur courante (Singh et Yee, 1994;

Gordon, 1995; Bertsekas et Tsitsiklis, 1996; Munos, 2007; Ernst *et al.*, 2005; Antos *et al.*, 2007; Munos et Szepesvári, 2008). D'un autre côté, API estime généralement de manière approchée la valeur de la politique courante—par exemple en utilisant les méthodes que nous avons décrites dans les deux précédents chapitres—et génère une nouvelle politique gloutonne par rapport à cette approximation (Bertsekas et Tsitsiklis, 1996; Munos, 2003; Lagoudakis et Parr, 2003b; Lazaric *et al.*, 2012) ; des approches alternatives (Lagoudakis et Parr, 2003a; Fern *et al.*, 2006; Lazaric *et al.*, 2010) essaient de directement produire une politique gloutonne par rapport à la valeur de la précédente. Le but de ce chapitre central (et aussi un peu plus long que les autres) est de montrer que, comme dans le cas exact, des versions approchées de MPI (AMPI, pour Approximate MPI) constituent une alternative intéressante à AVI et API.

4.1 Une famille d'algorithmes

Dans cette section, nous décrivons trois implémentations approchées du schéma algorithmique MPI. Ces implémentations reposent sur un espace de fonctions $\mathcal{F}$ pour représenter les fonctions valeurs, et pour le troisième, sur un espace de politiques Π pour représenter les politiques gloutonnes. Comme il s'agit d'algorithmes qui approchent le schéma *itératif* de MPI, nous nous concentrons dans ce qui suit sur la description de l'*itération* k de ces implémentations.

AMPI-V La première implémentation qui est aussi probablement la plus naturelle, désignée AMPI-V, est décrite à la figure 4.1. Dans AMPI-V, nous supposons que les fonctions valeurs $(v_k)_{k\geq 1}$ sont représentées dans un espace fonctionnel $\mathcal{F} \subseteq \mathbb{R}^X$. En n'importe quel état x, l'action $\pi_{k+1}(x)$ qui est gloutonne par rapport à v_k peut être estimée de la manière suivante :

$$\pi_{k+1}(x) \in \arg\max_{a\in A} \frac{1}{M}\sum_{j=1}^{M} \big(\widehat{T}_a^{(j)} v_k\big)(x), \tag{4.3}$$

$$\text{avec} \qquad \big(\widehat{T}_a^{(j)} v_k\big)(x) = r_a^{(j)} + \gamma v_k(x_a^{(j)}),$$

où pour tout $a \in A$ et $1 \leq j \leq M$, $r_a^{(j)}$ et $x_a^{(j)}$ sont réalisations de la récompense et de l'état suivant lorsque l'action a est choisie en l'état x.[1] Ainsi, approcher l'action gloutonne en un état x requiert $M|A|$ échantillons de transitions.

L'algorithme fonctionne de la manière suivante. Etant donnée une loi μ sur X, on construit un ensemble d'états $\mathcal{D}_k = \{x^{(i)}\}_{i=1}^N$, $x^{(i)} \overset{iid}{\sim} \mu$. On note $\widehat{\mu}$ la distribution empirique correspondante. A partir de chaque état $x^{(i)} \in \mathcal{D}_k$, on génère une trajectoire de longueur m,

$$\big(x^{(i)}, a_0^{(i)}, r_0^{(i)}, x_1^{(i)}, \ldots, a_{m-1}^{(i)}, r_{m-1}^{(i)}, x_m^{(i)}\big),$$

où $a_t^{(i)}$ est l'action $\pi_{k+1}(x_t^{(i)})$ estimée selon l'équation (4.3), et $r_t^{(i)}$ et $x_{t+1}^{(i)}$ sont des réalisations de récompenses et d'états suivants induites par ces actions. Pour chaque $x^{(i)}$, on calcule alors la quantité :

$$\widehat{v}_{k+1}(x^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m v_k(x_m^{(i)}), \tag{4.4}$$

qui est un estimateur non biaisé de $\big[(T_{\pi_{k+1}})^m v_k\big](x^{(i)})$. Finalement, la nouvelle valeur v_{k+1} est calculée comme le meilleur régresseur dans $\mathcal{F}$ de l'ensemble des estimateurs $(\widehat{v}_{k+1}(x^{(i)}))$,

1. Quand M tend vers l'infini, cette estimation de l'action tend effectivement vers une action gloutonne.

entrées : un espace de fonctions $\mathcal{F} \subset \mathbb{R}^X$, une loi μ sur X
initialisation : soit $v_0 \in \mathcal{F}$ arbitraire
pour $k = 0, 1, \ldots$
- ***génération des trajectoires et calcul des estimés :***

échantillonner l'ensemble des états $\mathcal{D}_k = \{x^{(i)}\}_{i=1}^N$, $x^{(i)} \overset{\text{iid}}{\sim} \mu$
pour tous $x^{(i)} \in \mathcal{D}_k$
générer une trajectoire (utilisant l'équation (4.3) pour chaque action)
$\widehat{v}_{k+1}(x^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m v_k(x_m^{(i)})$
fin pour
- ***calcul de la nouvelle valeur :***

$v_{k+1} \in \operatorname*{argmin}_{v \in \mathcal{F}} \widehat{\mathcal{L}}_k^{\mathcal{F}}(\widehat{\mu}; v)$ **(régression)** (voir l'équation (4.5))
fin pour

FIGURE 4.1 – Pseudo-code de l'algorithme AMPI-V.

c'est-à-dire que c'est une fonction $v \in \mathcal{F}$ qui minimise l'erreur empirique

$$\widehat{\mathcal{L}}_k^{\mathcal{F}}(\widehat{\mu}; v) = \frac{1}{N} \sum_{i=1}^{N} \big(\widehat{v}_{k+1}(x^{(i)}) - v(x^{(i)})\big)^2, \tag{4.5}$$

dans le but de minimiser l'erreur

$$\mathcal{L}_k^{\mathcal{F}}(\mu; v) = \Big|\Big|\big[(T_{\pi_{k+1}})^m v_k\big] - v\Big|\Big|_{2,\mu}^2.$$

La politique π_{k+1} étant approximativement gloutonne par rapport à v_k, l'algorithme essaye ainsi de suivre au plus près ce que ferait l'algorithme MPI s'il était implémenté de manière exacte ; en particulier, si μ chargeait tous les états, $N = M = \infty$ et $\mathcal{F}$ contenait la totalité des fonctions valeurs possibles, les itérations d'AMPI-V seraient identiques à celles de MPI. Chaque itération d'AMPI-V requiert N trajectoires de taille m, et dans chaque trajectoire, chacune des $|A|$ actions requiert M échantillons pour calculer l'équation (4.3). Au total, une itération requiert d'échantillonner $Nm(M|A|+1)$ transitions du MDP. Dans le cas particulier $m = 1$, on retombe sur l'algorithme de la littérature fitted-value iteration (Munos et Szepesvári, 2008).

AMPI-Q L'évaluation d'une action gloutonne étant assez coûteuse dans AMPI-V, l'algorithme que nous allons décrire à présent, AMPI-Q, utilise une astuce usuelle de la littérature "apprentissage par renforcement", l'emploi d'une fonction valeur états-actions [2] $q : X \times A \to \mathbb{R}$. La figure 4.2 contient le pseudo-code de cet algorithme. L'opérateur de Bellman linéaire associé à une politique π est alors défini par :

$$\forall x, a, \quad [T_\pi q](x, a) = \mathbb{E}\big[r(x, a) + \gamma q(x', \pi(x')) \mid x' \sim \mathbb{P}(\cdot|x, a)\big],$$

et l'opérateur glouton est tel que

$$\pi \in \mathcal{G}q \iff \forall x, \quad \pi(x) \in \arg\max_{a \in A} q(x, a).$$

2. Comme expliqué par Bertsekas et Tsitsiklis (1996), on peut voir cette astuce comme transformant le MDP originel contenant l'ensemble des états X en un nouveau MDP équivalent dans lequel l'espace d'états est $X \cup (X \times A)$. Ainsi, dans un état x de ce nouveau MDP, l'action a implique une transition déterministe vers l'état (x, a) ; puis, à partir de l'état (x, a), on effectue une transition non contrôlée vers un nouvel état x' dont la loi est celle induite par l'action a dans l'état x dans le MDP originel. Découper les transitions en 2 étapes—une transition déterministe non-contrôlée, puis une transition stochastique non contrôlée—facilite un certain nombre de calculs, notamment celui des actions gloutonnes qui deviennent la maximisation d'une fonction qui ne contient pas de terme "espérance".

Dans AMPI-Q, les fonctions valeurs états-actions sont représentées dans un espace de fonctions $\mathcal{F} \subseteq \mathbb{R}^{X \times A}$, et l'action $\pi_{k+1}(x)$, qui est gloutonne par rapport à la valeur q_k en l'état x est simplement calculée comme suit :

$$\pi_{k+1}(x) \in \arg\max_{a \in A} \; q_k(x, a). \tag{4.6}$$

L'*étape d'évaluation* est similaire à celle d'AMPI-V, à la différence que maintenant nous travaillons à partir de paires états-actions. Etant donnée une loi μ sur $\mathcal{S} \times A$, on construit un ensemble de paires états-actions $\mathcal{D}_k = \{(x^{(i)}, a^{(i)})\}_{i=1}^{N}$, $(x^{(i)}, a^{(i)}) \overset{iid}{\sim} \mu$. On note ici encore $\widehat{\mu}$ la distribution empirique correspondante. Pour chaque $(x^{(i)}, a^{(i)}) \in \mathcal{D}_k$, on génére une trajectoire de taille m,

$$\big(x^{(i)}, a^{(i)}, r_0^{(i)}, x_1^{(i)}, a_1^{(i)}, \cdots, x_m^{(i)}, a_m^{(i)}\big),$$

où la première action est $a^{(i)}$, et les actions suivantes $a_t^{(i)}$ pour $1 \leq t \leq m$ sont les actions $\pi_{k+1}(x_t^{(i)})$ gloutonnes par rapport à q_k calculées par l'équation (4.6), et $r_t^{(i)}$ et $x_{t+1}^{(i)}$ sont des réalisations de récompenses et d'états suivants induites par ces actions. Pour chaque $(x^{(i)}, a^{(i)}) \in \mathcal{D}_k$, on calcule alors la quantité :

$$\widehat{q}_{k+1}(x^{(i)}, a^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m q_k(x_m^{(i)}, a_m^{(i)}),$$

qui est un estimateur non biaisé de $\big[(T_{\pi_{k+1}})^m q_k\big](x^{(i)}, a^{(i)})$. Finalement, la nouvelle valeur q_{k+1} est calculée comme le meilleur régresseur dans $\mathcal{F}$ des estimateurs $(\widehat{q}_{k+1}(x^{(i)}, a^{(i)}))$, c'est-à-dire que c'est la fonction $q \in \mathcal{F}$ qui minimise l'erreur empirique

$$\widehat{\mathcal{L}}_k^{\mathcal{F}}(\widehat{\mu}; q) = \frac{1}{N} \sum_{i=1}^{N} \big(\widehat{q}_{k+1}(x^{(i)}, a^{(i)}) - q(x^{(i)}, a^{(i)})\big)^2, \tag{4.7}$$

dans le but de minimiser l'erreur

$$\mathcal{L}_k^{\mathcal{F}}(\mu; Q) = \Big|\Big|\big[(T_{\pi_{k+1}})^m q_k\big] - q\Big|\Big|_{2,\mu}^2.$$

La politique π_{k+1} est ici *exactement* gloutonne par rapport à q_k, ainsi AMPI-Q essaye de suivre au plus près ce que ferait l'algorithme MPI défini sur des fonctions valeurs états-actions. Si μ chargeait tous les états, $N = \infty$ et $\mathcal{F}$ contenait toutes les fonctions valeurs états-actions possibles, on génèrerait les même itérés que la version exacte de MPI. Chaque itération d'AMPI-Q requiert Nm échantillons de transition, ce qui est plus économe qu'AMPI-V. Cependant, AMPI-Q utilise un espace $\mathcal{F}$ de fonctions valeur états-actions, ce qui est plus complexe que l'espace requis par AMPI-V. Dans le cas particulier $m = 1$, on retombe sur l'algorithme de la littérature fitted-Q iteration (Ernst *et al.*, 2005; Antos *et al.*, 2007).

entrées : un espace de fonctions $\mathcal{F} \subset \mathbb{R}^{X \times A}$, une loi μ sur $X \times A$
initialisation : soit $q_0 \in \mathcal{F}$ arbitraire
pour $k = 0, 1, \ldots$
- ***génération des trajectoires et calcul des estimés :***

échantillonner l'ensemble des paires états-actions $\mathcal{D}_k = \{(x^{(i)}, a^{(i)}\}_{i=1}^N,\ (x^{(i)}, a^{(i)}) \overset{\text{iid}}{\sim} \mu$
pour tous $(x^{(i)}, a^{(i)}) \in \mathcal{D}_k$
générer une trajectoire (utilisant l'équation (4.6) pour chaque action)
$\widehat{q}_{k+1}(x^{(i)}, a^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m q_k(x_m^{(i)}, a_m^{(i)})$,
fin pour
- ***calcul de la nouvelle valeur :***

$q_{k+1} \in \underset{Q \in \mathcal{F}}{\operatorname{argmin}}\, \widehat{\mathcal{L}}_k^{\mathcal{F}}(\widehat{\mu}; Q)$ **(régression)** (voir l'équation (4.7))
fin pour

FIGURE 4.2 – Pseudo-code de l'algorithme AMPI-Q.

CBMPI La troisième et dernière implémentation que nous présentons utilise une représentation explicite pour les politiques (π_k), en plus de la représenation pour les fonction valeur (v_k), et est désignée par CBMPI (pour Classification Based MPI). L'idée est similaire aux algorithmes de type API basés sur des classifieurs (Lagoudakis et Parr, 2003a; Fern *et al.*, 2006; Lazaric *et al.*, 2010) dans lesquels la politique gloutonne est choisie dans l'espace des politiques Π via un problème de classification plutôt que via l'action gloutonne d'une fonction valeur obtenue par régression (comme dans AMPI-V et AMPI-Q). Afin de décrire CBMPI, nous récrivons le schéma algorithmique MPI (équations (4.1) et (4.2)) sous la forme suivante équivalente :

$$v_k = (T_{\pi_k})^m v_{k-1} \qquad \text{(étape d'évaluation)} \tag{4.8}$$
$$\pi_{k+1} = \mathcal{G}\big[(T_{\pi_k})^m v_{k-1}\big] \qquad \text{(étape gloutonne)} \tag{4.9}$$

Il est intéressant de remarquer qu'avec cette reformulation, v_k et π_{k+1} sont toutes deux des fonctions de $(T_{\pi_k})^m v_{k-1}$. L'implémentation CBMPI est une version approchée de cette reformulation. Comme décrit à la figure 4.3, CBMPI commence avec une politique $\pi_1 \in \Pi$ et une valeur $v_0 \in \mathcal{F}$ arbitraires. A l'itération k, on a à disposition une politique π_k et une valeur v_{k-1}. Une nouvelle valeur v_k est calculée comme une bonne approximation dans $\mathcal{F}$ de $(T_{\pi_k})^m v_{k-1}$. Ceci est fait en construisant un problème de régression dont $(T_{\pi_k})^m v_{k-1}$ est la cible. On construit un ensemble d'états $\mathcal{D}_k$ en échantillonnant N états i.i.d. à partir d'une distribution μ sur X : $\mathcal{D}_k = \{x^{(i)}\}_{i=1}^N,\ x^{(i)} \overset{\text{iid}}{\sim} \mu$. On note $\widehat{\mu}$ la loi empirique correspondante. Pour chaque $x^{(i)} \in \mathcal{D}_k$, on génère une trajectoire de taille m :

$$\big(x^{(i)}, a_0^{(i)}, r_0^{(i)}, x_1^{(i)}, \ldots, a_{m-1}^{(i)}, r_{m-1}^{(i)}, x_m^{(i)}\big)$$

où pour tout t, $a_t^{(i)} = \pi_k(x_t^{(i)})$, et $r_t^{(i)}$ et $x_{t+1}^{(i)}$ sont des réalisations de récompenses et de transitions induites par ce choix d'action. A partir de cette trajectoire, on calcule la quantité similaire à l'équation (4.4))

$$\widehat{v}_k(x^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m v_{k-1}(x_m^{(i)}), \tag{4.10}$$

qui est une estimateur non biaisé de $\big[(T_{\pi_k})^m v_{k-1}\big](x^{(i)})$. On utilise alors ces estimateur pour construire un ensemble d'apprentissage $\big\{\big(x^{(i)}, \widehat{v}_k(x^{(i)})\big)\big\}_{i=1}^N$, qui est utilisé par régresser la nouvelle fonction v_k. De manière similaire à l'algorithme AMPI-V, on calcule la fonction $v \in \mathcal{F}$ qui

entrées : un espace de fonctions $\mathcal{F} \subset \mathbb{R}^X$, un espace de polique $\Pi \subset A^X$, une loi μ sur X
initialisation : soit $\pi_1 \in \Pi$ et $v_0 \in \mathcal{F}$ arbitraires
pour $k = 1, 2, \ldots$
- ***génération des trajectoires et calcul des estimés :***

échantillonner l'ensemble des états $\mathcal{D}_k = \{x^{(i)}\}_{i=1}^N,\ x^{(i)} \overset{\text{iid}}{\sim} \mu$
pour tous $x^{(i)} \in \mathcal{D}_k$
générer une trajectoire et estimer $\widehat{v}_k(x^{(i)})$ (utilisant l'équation (4.10))
fin pour
échantillonner l'ensemble des états $\mathcal{D}'_k = \{x^{(i)}\}_{i=1}^{N'},\ x^{(i)} \overset{\text{iid}}{\sim} \mu$
pour tous $x^{(i)} \in \mathcal{D}'_k$ et $a \in A$
pour $j = 1$ à M
générer une trajectoire et estimer $R_k^j(x^{(i)}, a)$ (utilisant l'équation (4.15))
fin pour
$\widehat{q}_k(x^{(i)}, a) = \frac{1}{M} \sum_{j=1}^M R_k^j(x^{(i)}, a)$
fin pour
- ***calcul de la nouvelle valeur :***

$v_k \in \operatorname{argmin}_{v \in \mathcal{F}} \widehat{\mathcal{L}}_k^{\mathcal{F}}(\widehat{\mu}; v)$ **(régression)** (voir l'équation (4.11))
- ***calcul de la nouvelle politique :***

$\pi_{k+1} \in \operatorname{argmin}_{\pi \in \Pi} \widehat{\mathcal{L}}_k^{\Pi}(\widehat{\mu}; \pi)$ **(classification)** (voir l'équation (4.16))

fin pour

FIGURE 4.3 – Pseudo-code de l'algorithme CBMPI.

minimise l'erreur empirique

$$\widehat{\mathcal{L}}_k^{\mathcal{F}}(\widehat{\mu}; v) = \frac{1}{N} \sum_{i=1}^N \big(\widehat{v}_k(x^{(i)}) - v(x^{(i)})\big)^2, \tag{4.11}$$

dans le but de minimiser l'erreur

$$\mathcal{L}_k^{\mathcal{F}}(\mu; v) = \Big\| \big[(T_{\pi_k})^m v_{k-1}\big] - v \Big\|_{2,\mu}^2.$$

L'étape gloutonne à l'itération k vise à calculer π_{k+1} comme la meilleure approximation dans l'espace Π de $\mathcal{G}\big[(T_{\pi_k})^m v_{k-1}\big]$. Comme nous allons le voir, ceci peut se faire en résolvant un problème de classification. A partir de la définition d'une politique gloutonne, si $\pi \in \mathcal{G}\big[(T_{\pi_k})^m v_{k-1}\big]$, alors pour tout état $x \in X$,

$$\big[T_\pi (T_{\pi_k})^m v_{k-1}\big](x) = \max_{a \in A} \big[T_a (T_{\pi_k})^m v_{k-1}\big](x). \tag{4.12}$$

En définissant $q_k(x, a) = \big[T_a (T_{\pi_k})^m v_{k-1}\big](x)$, on peut récrire l'équation (4.12) de la manière suivante :

$$q_k\big(s, \pi(x)\big) = \max_{a \in A} q_k(x, a).$$

Il est alors naturel de considérer l'erreur suivante :

$$\mathcal{L}_{\pi_k, v_{k-1}}^{\Pi}(\mu; \pi) = \sum_x \mu(x) \Big(\max_{a \in A} q_k(x, a) - q_k\big(x, \pi(x)\big) \Big) = \Big\| \max_{a \in A} q_k(\cdot, a) - q_k\big(\cdot, \pi(\cdot)\big) \Big\|_{1,\mu}. \tag{4.13}$$

Pour alléger, on utilisera dans la suite la notation $\mathcal{L}_k^{\Pi}$ au lieu $\mathcal{L}_{\pi_k, v_{k-1}}^{\Pi}$. Pour approcher la solution du problème de minimisation de cette erreur, on construit l'ensemble d'états $\mathcal{D}'_k =$

$\{x^{(i)}\}_{i=1}^{N'},\ x^{(i)} \overset{\text{iid}}{\sim} \mu$. Pour chaque état $x^{(i)} \in \mathcal{D}'_k$ et chaque action $a \in A$, on génére M trajectoires indépendantes de longueur $m+1$, [3]

$$\left(x^{(i)}, a, r_0^{(i,j)}, x_1^{(i,j)}, a_1^{(i,j)}, \ldots, a_m^{(i,j)}, r_m^{(i,j)}, x_{m+1}^{(i,j)}\right)_{j=1}^{M},$$

où pour tout $t \geq 1$, $a_t^{(i,j)} = \pi_k(x_t^{(i,j)})$, et $r_t^{(i,j)}$ et $x_{t+1}^{(i,j)}$ sont des réalisations de récompense et état suivant induits par ce choix d'action. A partir de ces trajectoires, on calcule la quantité :

$$\widehat{q}_k(x^{(i)}, a) = \frac{1}{M} \sum_{j=1}^{M} R_k^j(x^{(i)}, a) \tag{4.14}$$

avec

$$R_k^j(x^{(i)}, a) = \sum_{t=0}^{m} \gamma^t r_t^{(i,j)} + \gamma^{m+1} v_{k-1}(x_{m+1}^{(i,j)}). \tag{4.15}$$

Ainsi, il est ici encore facile de voir que $\widehat{q}_k(x^{(i)}, a)$ est un estimateur non-biaisé de $q_k(x^{(i)}, a)$. Finalement, la nouvelle politique π_{k+1} est le classifieur qui minimise l'erreur empirique

$$\widehat{\mathcal{L}}_k^{\Pi}(\widehat{\mu}; \pi) = \frac{1}{N'} \sum_{i=1}^{N'} \Big[\max_{a \in A} \widehat{q}_k(x^{(i)}, a) - \widehat{q}_k\big(x^{(i)}, \pi(x^{(i)})\big) \Big], \tag{4.16}$$

dans le but de minimiser l'erreur $\mathcal{L}_k^{\Pi}(\mu; \pi)$ définie à l'équation (4.13). Il s'agit ici d'un problème de *classification multi-classes à coût sensitif* : multi-classes car en général on peut avoir plus de 2 actions, et coût sensitif car chaque terme apparaissant dans la somme à minimiser ci-dessus peut se récrire comme un produit de deux termes

$$\max_{a \in A} \widehat{q}_k(x^{(i)}, a) - \widehat{q}_k\big(x^{(i)}, \pi(x^{(i)})\big) \ = \ \mathbb{1}_{\pi(x^{(i)}) \notin \operatorname{argmax}_a \widehat{q}_k(x^{(i)}, a)} \left[\max_{a \in A} \widehat{q}_k(x^{(i)}, a) - \widehat{q}_k\big(x^{(i)}, \pi(x^{(i)})\big) \right],$$

le premier valant 1 si π ne choisit pas une des meilleures actions en $x^{(i)}$ (et 0 sinon), et le second pondérant ces erreurs.

Comme pour les algorithmes précédents, si on utilisait une distribution μ qui charge tous les états, si $N = N' = M = \infty$ et si les espaces $\mathcal{F}$ et Π contenaient respectivement toutes les fonctions valeur et toutes les politiques possibles, on génèrerait la même séquence que le schéma décrit par les équations (4.8) et (4.9), qui est équivalent à MPI sans approximation. Chaque itération de CBMPI requiert $Nm + M|A|N'(m+1)$ échantillons de transitions. Lorsqu'on prend $m = \infty$, on retombe sur l'algorithme de la littérature DPI (Direct Policy Iteration) (Lazaric *et al.*, 2010).

4.2 Propagation des erreurs

Pour proposer une analyse qui s'applique aux trois implémentations que nous venons d'introduire, nous allons considérer un algorithme abstrait, AMPI, qui rend compte des erreurs à chaque étape.

3. En pratique, on peut implémenter CBMPI d'une manière plus économe en réutilisant les trajectoires générées lors de l'étape gloutonne dans l'étape d'estimation. Nous ne considérons pas ce cas pour garder l'analyse aussi simple que possible.

Analyse générale

AMPI commence avec une valeur v_0 arbitraire, et à chaque itération $k \geq 1$, il calcule une politique gloutonne par rapport à v_{k-1} avec une possible erreur ϵ'_k :

$$\forall \pi', \quad T_{\pi'} v_{k-1} \leq T_{\pi_k} v_{k-1} + \epsilon'_k,$$

ce que nous notons

$$\pi_k = \widehat{\mathcal{G}}_{\epsilon'_k} v_{k-1}. \tag{4.17}$$

AMPI génère ensuite une nouvelle fonction valeur v_k avec une erreur potentielle ϵ_k :

$$v_k = (T_{\pi_k})^m v_{k-1} + \epsilon_k. \tag{4.18}$$

Avant de montrer comment ces deux erreurs sont propagées au cours des itérations, montrons que ce schéma algorithmique abstrait permet effectivement d'étudier chacune des implémentations que nous avons introduites à la section 4.1.

— **AMPI-V :** Le terme ϵ_k est l'erreur lorsqu'on régresse la fonction valeur v_k. Comme c'est habituellement le cas dans la littérature sur la régression, cette erreur peut être décomposée en deux parties : l'erreur d'approximation liée au choix de l'espace de fonctions $\mathcal{F}$ et l'erreur d'estimation dûe au fait qu'on utilise un nombre fini N de trajectoires. Le terme ϵ'_k est l'erreur liée au fait d'utiliser un nombre fini d'échantillons M pour estimer les actions gloutonnes.

— **AMPI-Q :** Comme pour AMPI-V, le terme ϵ_k correspond à l'erreur de régression. Dans ce cas cependant, l'utilisation de fonctions valeur états-actions fait qu'il n'y a pas d'erreur sur l'étape gloutonne, c'est-à-dire que $\epsilon'_k = 0$.

— **CBMPI :** La situation est légèrement plus compliquée dans ce dernier cas car le schéma correspondant à CBMPI :

$$\begin{aligned} v_k &= (T_{\pi_k})^m v_{k-1} + \epsilon_k \\ \pi_{k+1} &= \widehat{\mathcal{G}}_{\epsilon'_{k+1}} \left[(T_{\pi_k})^m v_{k-1} \right] \end{aligned}$$

ne s'identifie pas directement aux équations (4.17) et (4.18). Cependant, en introduisant la variable auxiliaire $w_k = (T_{\pi_k})^m v_{k-1}$, on voit que $v_k = w_k + \epsilon_k$, et on peut alors voir que

$$\pi_{k+1} = \widehat{\mathcal{G}}_{\epsilon'_{k+1}} \left[w_k \right]. \tag{4.19}$$

Par ailleurs, en utilisant le fait que $v_{k-1} = w_{k-1} + \epsilon_{k-1}$, on a

$$w_k = (T_{\pi_k})^m v_{k-1} = (T_{\pi_k})^m (w_{k-1} + \epsilon_{k-1}) = (T_{\pi_k})^m w_{k-1} + (\gamma P_{\pi_k})^m \epsilon_{k-1}. \tag{4.20}$$

A présent, les équations (4.19) et (4.20) correspondent aux équations (4.17) et (4.18) si l'on remplace v_k par w_k et ϵ_k par $(\gamma P_{\pi_k})^m \epsilon_{k-1}$.

Dans la littérature, les analyses permettant de faire une étude de sensibilité au bruit pour les cas particulier de VI et PI sont assez différentes. L'analyse pour VI est généralement basée sur les propriétés de contraction de l'opérateur d'optimalité de Bellman T. Or, on peut montrer que l'opérateur par lequel PI met à jour la valeur d'une itération à la suivante n'est pas une contraction. Plus généralement, on peut même montrer que ce constat est vrai aussi pour MPI dès lors qu'il diffère de VI (dès que $m > 1$).

Théorème 4.1. *Si $m > 1$, il n'exsite aucune norme pour laquelle l'opérateur par lequel MPI met à jour la valeur est contractant.*

Démonstration. On considère le MDP avec deux états, $\{x_1, x_2\}$, deux actions $\{change, reste\}$, les récompenses $r(x_1) = 0,\ r(x_2) = 1$, et les transitions déterministes

$$P_{change}(x_2|x_1) = P_{change}(x_1|x_2) = P_{reste}(x_1|x_1) = P_{reste}(x_2|x_2) = 1.$$

Considérons deux fonctions valeur $v = (\epsilon, 0)$ et $v' = (0, \epsilon)$ avec $\epsilon > 0$. Leurs politiques gloutonnes sont respectivement $\pi = (reste, change)$ et $\pi' = (change, reste)$, et les itérés suivant v et v' égalent respectivement $(T_\pi)^m v = \begin{pmatrix} \gamma^m \epsilon \\ 1 + \gamma^m \epsilon \end{pmatrix}$ et $(T_{\pi'})^m v' = \begin{pmatrix} \frac{\gamma - \gamma^m}{1-\gamma} + \gamma^m \epsilon \\ \frac{1-\gamma^m}{1-\gamma} + \gamma^m \epsilon \end{pmatrix}$. Ainsi, $(T_{\pi'})^m v' - (T_\pi)^m v = \begin{pmatrix} \frac{\gamma - \gamma^m}{1-\gamma} \\ \frac{\gamma - \gamma^m}{1-\gamma} \end{pmatrix}$, tandis que $v' - v = \begin{pmatrix} -\epsilon \\ \epsilon \end{pmatrix}$. Comme ϵ peut être arbitrairement petit, la norme de $(T_{\pi'})^m v' - (T_\pi)^m v$ peut être arbitrairement plus grande que la norme de $v - v'$ dès que $m > 1$. □

En général, l'analyse de PI repose sur le fait que la séquence de fonctions valeur qu'il génère est croissante. Ici encore, on peut voir que dès que m est fini, les fonctions valeurs générées par MPI peuvent décroître (il suffit de considérer une grande valeur initiale). Autrement dit, on déduit de cette observation et du théorème 4.1 que pour les valeurs non-triviales de m ($1 < m < \infty$), MPI n'est ni contractant ni croissant, et l'on a besoin d'introduire une technique de preuve originale pour étudier la propagation des erreurs [4]. J'ai initialement développé cette analyse dans le cas similaire de λ-Itérations sur les politiques (Scherrer, 2007, 2013b)—qui est très proche de MPI. Dans ce qui suit, je ne décris que les grandes étapes de l'analyse ; les détails des preuves sont dans (Scherrer *et al.*, 2015). L'idée fondamentale est de décomposer l'analyse en considérant les trois quantités suivantes :

1. la distance entre la fonction valeur optimale et la valeur avant approximation à l'itération k :

$$d_k = v_* - (T_{\pi_k})^m v_{k-1} = v_* - (v_k - \epsilon_k);$$

2. le décalage (shift en anglais) entre la valeur avant approximation et la (vraie) valeur de la politique à l'itération k :

$$s_k = (T_{\pi_k})^m v_{k-1} - v_{\pi_k} = (v_k - \epsilon_k) - v_{\pi_k};$$

3. le résidu de Bellman de la valeur à l'itération k :

$$b_k = v_k - T_{\pi_{k+1}} v_k.$$

Nous cherchons un majorant de la *perte* (loss en anglais)

$$l_k = v_{\pi_*} - v_{\pi_k} = v_* - v_{\pi_k} = d_k + s_k$$

liée au fait d'utiliser la politique π_k produite par l'algorithme au lieu de la politique optimale π_*. Pour ce faire, nous allons majorer d_k et s_k, ce qui requiert de majorer également b_k. Précisément, le cœur de l'analyse consiste à prouver les inéquations vectorielles suivantes pour nos trois quantités d'intérêt.

Lemme 4.2. *Soient $x_k = (I - \gamma P_{\pi_k})\epsilon_k + \epsilon'_{k+1}$ et $y_k = -\gamma P_{\pi_*}\epsilon_k + \epsilon'_{k+1}$. Alors, pour tout $k \geq 1$, on a les inégalités vectorielles suivantes :*

$$b_k \leq (\gamma P_{\pi_k})^m b_{k-1} + x_k,$$

4. La difficulté ici n'est pas tant de dériver une borne d'erreur, mais d'en dériver une qui soit suffisamment fine pour qu'on retrouve les résultats connus dans les cas standards AVI et API.

$$d_k \leq \gamma P_{\pi_*} d_{k-1} + y_{k-1} + \sum_{j=1}^{m-1} (\gamma P_{\pi_k})^j b_{k-1},$$
$$s_k = (\gamma P_{\pi_k})^m (I - \gamma P_{\pi_k})^{-1} b_{k-1}.$$

Comme les matrices stochastiques apparaissant ci-dessus sont constituées de termes positifs, les bornes du lemme 4.2 indiquent que la perte l_k peut être contrôlée via ϵ_k et ϵ'_k. En effet, si ϵ est un vecteur qui majore uniformément en k les valeurs absolues composante par composante $|\epsilon_k|$ et $|\epsilon'_k|$, on a $x_k = O(\epsilon)$, $y_k = O(\epsilon)$, la première inégalité implique que $b_k \leq O(\epsilon)$, et par conséquent les deux autres inégalités nous donnent $d_k \leq O(\epsilon)$ et $s_k \leq O(\epsilon)$. Autrement dit, la perte $l_k = d_k + s_k$ satisfait $l_k \leq O(\epsilon)$.

La borne que nous allons décrire pour l_k est le résultat d'expansions et de combinaisons précises des trois inégalités du lemme 4.2. Nous introduisons une notation qui simplifie grandement l'énoncé ainsi que les preuves par rapport aux travaux de Munos (2003, 2007) sur API et AVI, et à mes premiers résultats obtenus sur λ-Itérations sur les Politiques (Scherrer, 2007, 2013b).

Définition 4.3. *Pour tout entier n positif, on définit $\mathbb{P}_n$ comme le plus petit ensemble de matrices tel que*

1. *pour tout ensemble de n politiques, $\{\pi_1, \ldots, \pi_n\}$, $(\gamma P_{\pi_1})(\gamma P_{\pi_2}) \ldots (\gamma P_{\pi_n}) \in \mathbb{P}_n$,*
2. *pour tout $\alpha \in [0,1]$ et $(P_1, P_2) \in \mathbb{P}_n \times \mathbb{P}_n$, $\alpha P_1 + (1-\alpha) P_2 \in \mathbb{P}_n$.*

De plus, on utilisera la notation (légèrement abusive) Γ^n pour désigner n'importe quel élément de $\mathbb{P}_n$. Ainsi, lorsque nous écrivons pour une certaine matrice P que

$$P = \alpha_1 \Gamma^i + \alpha_2 \Gamma^j \Gamma^k = \alpha_1 \Gamma^i + \alpha_2 \Gamma^{j+k},$$

cela signifie : "il existe $P_1 \in \mathbb{P}_i$, $P_2 \in \mathbb{P}_j$, $P_3 \in \mathbb{P}_k$, et $P_4 \in \mathbb{P}_{k+j}$ tels que

$$P = \alpha_1 P_1 + \alpha_2 P_2 P_3 = \alpha_1 P_1 + \alpha_2 P_4."$$

Avec cette nouvelle notation, on peut montrer que le lemme suivant est une conséquence du lemme 4.2.

Lemme 4.4. *Après k itérations, les pertes d'AMPI-V et AMPI-Q satisfont l'équation vectorielle*

$$l_k \leq 2 \sum_{i=1}^{k-1} \sum_{j=i}^{\infty} \Gamma^j |\epsilon_{k-i}| + \sum_{i=0}^{k-1} \sum_{j=i}^{\infty} \Gamma^j |\epsilon'_{k-i}| + h(k),$$

tandis que la perte de CBMPI satisfait l'équation vectorielle

$$l_k \leq 2 \sum_{i=1}^{k-2} \sum_{j=i+m}^{\infty} \Gamma^j |\epsilon_{k-i-1}| + \sum_{i=0}^{k-1} \sum_{j=i}^{\infty} \Gamma^j |\epsilon'_{k-i}| + h(k),$$

où $h(k) = 2 \sum_{j=k}^{\infty} \Gamma^j |d_0|$ ou $h(k) = 2 \sum_{j=k}^{\infty} \Gamma^j |b_0|$.

Remarque 4.5. *Les bornes analogues dérivées pour AVI (Munos, 2007, Lemme 4.1) et API (Munos, 2003, Corollaire 10) ne prennent pas en compte l'erreur dans l'étape gloutonne (autrement dit on a $\epsilon'_k = 0$) et ont la forme suivante :*

$$\limsup_{k \to \infty} l_k \leq 2 \limsup_{k \to \infty} \sum_{i=1}^{k-1} \sum_{j=i}^{\infty} \Gamma^j |\epsilon_{k-i}|.$$

Ceci indique que non seulement la première borne du lemme 4.4 unifie les analyses d'AVI et API, mais en plus elle les généralise au cas où il y a potentiellement une erreur dans l'étape gloutonne et pour un nombre fini d'itérations k. De plus, il est remarquable que nos bornes ne dépendent pas du paramètre m. [5]

Une conséquence immédiate et importante des bornes du lemme 4.4 est qu'on peut déduire une garantie sur la performance des algorithmes. Nous considérons AMPI-Q et AMPI-V (l'argument étant similaire pour CBMPI). Définissons $\epsilon = \sup_{j\geq 1} \|\epsilon_j\|_\infty$ et $\epsilon' = \sup_{j\geq 1} \|\epsilon'_j\|_\infty$, c'est-à-dire des bornes uniformes sur les erreurs. En prenant la norme infinie (utilisant notamment le fait que pour tout i, $\|\Gamma^i\|_\infty = \gamma^i$) et la limite supérieure quand k tend vers l'infini, on obtient :

$$\limsup_{k\to\infty} \|l_k\|_\infty \leq \frac{2\gamma\epsilon + \epsilon'}{(1-\gamma)^2}. \tag{4.21}$$

Ce résultat est une généralisation des bornes connues pour AVI ($m = 1$ et $\epsilon' = 0$) et API ($m = \infty$) décrites par Bertsekas et Tsitsiklis (1996). On a l'interprétation suivante : si on contrôle la norme infinie des erreurs pour toutes les itérations, alors on contrôle la perte de la politique retournée asymptotiquement par l'algorithme par rapport à la politique optimale. Symétriquement, on peut voir que les amplitudes des erreurs ne doivent pas être trop grandes pour que cette garantie soit informative : comme la perte est nécessairement majorée par $2V_{\max} = \frac{2R_{\max}}{1-\gamma}$ (c'est la difference de 2 valeurs qui sont nécessairement dans $[-V_{\max}, V_{\max}]$), cette garantie n'apporte plus rien dès lors que $2\gamma\epsilon + \epsilon' > 2(1-\gamma)^2 V_{\max} = 2(1-\gamma)R_{\max}$.

Supposons qu'il soit possible de résoudre des problèmes de régression et de classification de sorte à contrôler l'erreur en norme infinie. Alors, le résultat ci-dessus signifie qu'on peut faire une *réduction* du problème de contrôle optimal vers une séquence de tels problèmes de régression et de classification. En particulier, si une percée significative était faite dans la littérature de ces problèmes (plus standards), le domaine du contrôle optimal en bénéficierait automatiquement. Comme la plupart des approches de régression et classification (notamment celles que nous avons décrites dans les trois implémentations spécifiques de la section précédente) contrôlent une norme ℓ_p à poids—qui nous rappelons est définie par $\|u\|_{p,\mu} = \left[\sum_x \mu(x)|u(x)|^p\right]^{1/p}$—, la portée d'un résultat comme celui énoncé à l'équation (4.21) est limitée. Le reste de la section est consacrée à la dérivation de bornes en fonction de normes ℓ_p des erreurs.

Munos (2003, 2007); Munos et Szepesvári (2008) et Farahmand *et al.* (2010) ont montré comment dériver ces bornes plus réalistes. L'approche consiste à introduire des coefficients dits de *concentrabilité* parce qu'ils mesurent comment une distribution sur les états peut se concentrer lorsque la dynamique du MDP évolue. Etant donnés deux lois μ et ρ, des entiers i et α, on s'intéresse à la quantité suivante :

$$c_\alpha(i) = \max_{\pi_1,\ldots,\pi_j} \left\| \frac{(P_{\pi_j} P_{\pi_{j-1}} \cdots P_{\pi_1})'\rho}{\mu} \right\|_{\alpha,\mu},$$

où $\pi_1, \ldots, \pi_j$ est n'importe quel ensemble de politiques du MDP, $(P_{\pi_j} P_{\pi_{j-1}} \cdots P_{\pi_1})'\rho$ est la distribution obtenue si partant de ρ, on effectue j pas en suivant successivement les politiques $\pi_1, \pi_2, \cdots \pi_j$, et $\frac{\nu}{\mu}$ désigne le quotient composante par composante des vecteurs ν et μ. [6] Dans

5. L'indépendance par rapport à m est liée au niveau d'abstraction de la définition 4.3. La dépendance visà-vis de m reapparaîtrait si l'on explicitait ce qui est caché dans la notation Γ^j. Mais cela aurait aussi pour effet de compliquer significativement l'analyse, comme on peut le voir dans les résultats décrits dans (Scherrer, 2007, 2013b) pour λ-Itérations sur les politiques.

6. Lorsqu'on s'intéresse à des espaces d'états plus généraux (non nécessairement dénombrables) et que ν et μ sont des mesures générales, on se base alors sur la dérivée de Radon-Nikodym $\frac{d\nu}{d\mu}$, avec la convention que le coefficient $c_q(j)$ est infini dès lors qu'une des mesures du dénominateur n'est pas absolument continue par rapport à μ. Les résultats que nous présentons ci-après s'étendent sans difficulté.

la mesure où ces coefficients apparaîtront dans les bornes que nous allons bientôt décrire, ces dernières ne seront informatives que si ces coefficients sont finis. Nous renvoyons le lecteur aux articles (Munos, 2007; Munos et Szepesvári, 2008; Farahmand *et al.*, 2010) pour plus de détails concernant ces coefficients ; en particulier, on pourra trouver une illustration simple où ces coefficients sont relativement petits dans (Munos, 2007, sections 5.5 et 7). Munis de ces coefficients, nous pouvons maintenant énoncer un lemme technique qui permet de convertir des inégalités vectorielles du type de celles du lemme 4.4 en bornes en norme ℓ_p, et qui généralise légèrement l'analyse de Farahmand *et al.* (2010).

Lemme 4.6. *Soient $\mathcal{I}$ et $(\mathcal{J}_i)_{i\in\mathcal{I}}$ deux ensembles d'entiers positifs ou nuls, $\{\mathcal{I}_1,\dots,\mathcal{I}_n\}$ une partition de $\mathcal{I}$, et f et $(g_i)_{i\in\mathcal{I}}$ des fonctions satisfaisant*

$$|f| \le \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}_i}\Gamma^j|g_i| = \sum_{l=1}^{n}\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\Gamma^j|g_i|.$$

Alors, pour tout entier p, pour tous entiers α et α' tels $\frac{1}{\alpha}+\frac{1}{\alpha'}=1$,[7] *et toutes distributions ρ et μ, on a*

$$\|f\|_{p,\rho} \le \sum_{l=1}^{n}\left(\mathcal{C}_\alpha(l)\right)^{1/p}\sup_{i\in\mathcal{I}_l}\|g_i\|_{p\alpha',\mu}\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\gamma^j,$$

avec le coefficient de concentratibilité suivant

$$\mathcal{C}_\alpha(l) = \frac{\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\gamma^j c_\alpha(j)}{\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\gamma^j}.$$

Appliquer ce lemme aux inégalités vectorielles du lemme 4.4 nous mène au résultat suivant.

Théorème 4.7. *Pour tous entiers α, l, k et d, définissons le coefficient de concentrabilité suivant :*

$$\mathcal{C}_\alpha^{l,k,d} = \frac{(1-\gamma)^2}{\gamma^l-\gamma^k}\sum_{i=l}^{k-1}\sum_{j=i}^{\infty}\gamma^j c_\alpha(j+d).$$

Soient ρ et μ des distributions sur les états. Soient p, α et α' des entiers tels que $\frac{1}{\alpha}+\frac{1}{\alpha'}=1$. Soient enfin $\epsilon = \sup_{1\le j\le k-1}\|\epsilon_j\|_{p\alpha',\mu}$ et $\epsilon' = \sup_{1\le j\le k}\|\epsilon'_j\|_{p\alpha',\mu}$. Après k itérations, les pertes de AMPI-V et AMPI-Q satisfont

$$\|l_k\|_{p,\rho} \le \frac{2(\gamma-\gamma^k)\left(\mathcal{C}_\alpha^{1,k,0}\right)^{\frac{1}{p}}}{(1-\gamma)^2}\epsilon + \frac{(1-\gamma^k)\left(\mathcal{C}_\alpha^{0,k,0}\right)^{\frac{1}{p}}}{(1-\gamma)^2}\epsilon' + g(k),$$

tandis que la perte de CBMPI satisfait

$$\|l_k\|_{p,\rho} \le \frac{2\gamma^m(\gamma-\gamma^{k-1})\left(\mathcal{C}_\alpha^{2,k,m}\right)^{\frac{1}{p}}}{(1-\gamma)^2}\epsilon + \frac{(1-\gamma^k)\left(\mathcal{C}_\alpha^{0,k,0}\right)^{\frac{1}{p}}}{(1-\gamma)^2}\epsilon' + g(k), \tag{4.22}$$

avec

$$g(k) = \frac{2\gamma^k}{1-\gamma}\left(\mathcal{C}_\alpha^{k,k+1,0}\right)^{\frac{1}{p}}\min\left(\|d_0\|_{p\alpha',\mu}, \|b_0\|_{p\alpha',\mu}\right).$$

7. Dans la mesure où nous supposons α et α' entiers, on a $(\alpha,\alpha') \in \{(1,\infty);(2,2);(\infty,1)\}$.

Démonstration. Le résultat est obtenu pour AMPI-V et AMPI-Q en appliquant au lemme 4.4 le lemme 4.6 avec $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3\}$ avec $\mathcal{I}_1 = \{1, \dots, k-1\}$, $\mathcal{I}_2 = \{k, \dots, 2k-1\}$, et $\mathcal{I}_3 = \{2k\}$, et où pour tout $i \in \mathcal{I}$,

$$\mathcal{J}_i = \begin{cases} \{i, i+1, \cdots\} & \text{if} \quad 1 \leq i \leq k-1, \\ \{i-k, i-k+1, \cdots\} & \text{if} \quad k \leq i \leq 2k-1, \\ \{k\} & \text{if} \quad i = 2k. \end{cases}$$

$$\text{et} \quad g_i = \begin{cases} 2\epsilon_{k-i} & \text{if} \quad 1 \leq i \leq k-1, \\ \epsilon'_{k-(i-k)} & \text{if} \quad k \leq i \leq 2k-1, \\ 2d_0 \text{ (or } 2b_0) & \text{if} \quad i = 2k, \end{cases}$$

On utilise en particulier le fait que $\sum_{i=l}^{k-1} \sum_{j=i}^{\infty} \gamma^j = \frac{\gamma^l - \gamma^k}{(1-\gamma)^2}$. La preuve est similaire pour CBMPI. □

Si on fait tendre le nombre d'itérations k vers l'infini, on obtient la borne suivante pour AMPI-V et AMPI-Q :

$$\limsup_{k \to \infty} \|l_k\|_{p,\rho} \leq \frac{2\gamma \left(\mathcal{C}_{\alpha}^{1,\infty,0}\right)^{\frac{1}{p}} \epsilon + \left(\mathcal{C}_{\alpha}^{0,\infty,0}\right)^{\frac{1}{p}} \epsilon'}{(1-\gamma)^2}.$$

Comparée à la borne en norme infinie de l'équation (4.21), on voit ici que le prix à payer pour avoir une borne en norme ℓ_p est le couple de coefficients $\mathcal{C}_{\alpha}^{1,\infty,0}$ et $\mathcal{C}_{\alpha}^{0,\infty,0}$. De plus, il est facile de voir que cette dernière borne est plus générale : quand on fait tendre p vers l'infini, on retrouve l'équation (4.21).

Remarque 4.8. *Via le choix du couple de paramètres α et α' tels que $\frac{1}{\alpha} + \frac{1}{\alpha'} = 1$, il y a un compromis entre l'amplitude des coefficients de concentrabilité (plus α est grand et plus ces coefficients sont grands) et la difficulté de contrôler les erreurs (plus α' est grand et plus le contrôle de l'erreur, en norme $\ell_{p\alpha'}$, est un problème difficile). Sur ce point, notre analyse unifie celles de Munos (2007); Munos et Szepesvári (2008) ($\alpha = \infty$ et $\alpha' = 1$) et de Farahmand* et al. *(2010) ($\alpha = \alpha' = 2$).*

Remarque 4.9. *Il est intéressant de remarquer que notre analyse ne dépend pas* directement *du paramètre m pour AMPI-V et AMPI-Q (bien qu'elle en dépende indirectement à travers ϵ_k). Pour CBMPI, m permet de directement contrôler l'influence de l'erreur d'approximation de la fonction valeur, l'annulant en particulier lorsque m tend vers l'infini (cf. le premier terme de l'équation* (4.22)*). Si on suppose un budget fixe en nombre d'échantillons, augmenter m réduit le nombre de trajectoires utilisées par le classifieur et affecte en général sa qualité. Ainsi, m permet de faire un compromis entre l'erreur d'estimation de l'étape de classification et l'erreur globale de régression de la fonction valeur.*

On notera finalement ici que Canbolat et Rothblum (2012) ont indépendemment développé une analyse de sensibilité au bruit du schéma algorithmique MPI. Cependant, Canbolat et Rothblum (2012) considèrent uniquement le cas de l'erreur contrôlée en norme infinie, et l'analyse proposée concerne seulement la distance $v_* - v_k$, tandis que nous bornons ici la perte $v_* - v_{\pi_k}$. Si on peut déduire une borne sur la perte à partir de celle sur la distance (par exemple en utilisant le théorème 1 de Canbolat et Rothblum 2012), la borne alors obtenue est néanmoins moins fine que la nôtre. En particulier, cela ne permet pas de retrouver *exactement* les résultats standards pour AVI et API, comme c'était le cas avec notre analyse (équation (4.21)).

Application à l'analyse d'une implémentation particulière de CBMPI

Muni du résultat général donné par le théorème 4.7, on peut considérer des instances particulières des trois algorithmes (basés sur des régresseurs ou classifieurs particuliers) et dériver une analyse statistique plus précise en dérivant des bornes sur les erreurs ϵ et ϵ'. En guise d'illustration, nous allons dériver cette analyse plus précise dans le cas d'une instance particulière de CBMPI, qui contient les deux forme d'approximation. On supposera ici qu'on implémente CBMPI en représentant la fonction valeur à chaque étape k dans un espace linéaire. Dans ce cadre, on peut dériver la borne suivante sur l'erreur de mise à jour de la valeur.

Lemme 4.10 (**Erreur de l'étape d'évaluation de CBMPI).** *Soient $\{\phi_j\}_{j=1}^d$ des vecteurs linéairement indépendants sur X tels que $\|\phi_j\|_\infty \leq L$ et soit $\mathcal{F}$ l'espace linéaire engendré par ces vecteurs, i.e. $\mathcal{F} = \{f_\alpha(\cdot) = \phi(\cdot)'\alpha : \alpha \in \mathbb{R}^d\}$. Soit α_* tel que f_{α_*} est la meilleure approximation de $(T_{\pi_k})^m v_{k-1}$ dans $\mathcal{F}$, i.e. $f_{\alpha^*} = \operatorname{argmin}_{f\in\mathcal{F}} \|(T_{\pi_k})^m v_{k-1} - f\|_{2,\mu}$. Etant donnés l'échantilllon $\big\{\big(x^{(i)}, \widehat{v}_k(x^{(i)})\big)\big\}_{i=1}^N$, où pour tout i, $\widehat{v}_k(x^{(i)})$ est un estimateur non-biaisé de $[(T_{\pi_k})^m v_{k-1}](x^{(i)})$ calculé via l'équation* (4.10)*. Alors, la troncature v_k dans $[-V_{\max}, V_{\max}]$ du minimum empirique de $v \mapsto \widehat{\mathcal{L}}_k^{\mathcal{F}}(\widehat{\mu}; v)$ (équation* (4.11)*) satisfait avec probabilité au moins $1-\delta$,*

$$\|\epsilon_k\|_{2,\mu} = \|v_k - (T_{\pi_k})^m v_{k-1}\|_{2,\mu} \leq 4 \min_{f\in\mathcal{F}} \|(T_{\pi_k})^m v_{k-1} - f\|_{2,\mu} + e_1(N,\delta) + e_2(N,\delta),$$

$$\begin{aligned} \text{où} \qquad e_1(N,\delta) &= 32 V_{\max} \sqrt{\frac{2}{N} \log\Big(\frac{27(12e^2N)^{2(d+1)}}{\delta}\Big)}\,, \\ e_2(N,\delta) &= 24\Big(V_{\max} + \|\alpha_*\|_2 L\sqrt{d}\Big)\sqrt{\frac{2}{N}\log\frac{9}{\delta}}\,. \end{aligned}$$

Ce type de résultats est typique de ce qu'on peut trouver dans la littérature concernant la régression linéaire, dans lequel on voit que l'erreur dans le calcul v_k dépend à la fois d'une mesure d'erreur liée au choix de l'espace $\mathcal{F}$ et du nombre d'échantillons N avec une décroissance en $O(\frac{1}{\sqrt{N}})$.

Nous considérons à présent l'analyse de l'erreur lors de l'*étape gloutonne*. Pour simplifier ici, nous supposerons que l'espace d'actions ne contient que deux actions, ce qui permet d'analyser la performance du classifieur via la dimension de Vapnik–Chervonenkis [8].

Lemme 4.11 (**Erreur de l'étape gloutonne de CBMPI).** *On suppose que l'espace des politiques utilisées par le classifieur a une VC-dimension (Vapnik, 1995) finie h. Etant donné l'échantillon $\big\{\big(x^{(i)}, \widehat{q}_k(x^{(i)}, a)\big)\big\}_{i=1}^{N'}$, où pour tout état $x^{(i)}$ et chacune des deux actions a, $\widehat{q}_{k-1}(x^{(i)}, a)$ est calculé à partir de M trajectoires selon l'équation* (4.14)*), la politique π_k obtenue en minimisant le risque empirique de l'équation* (4.16) *satisfait avec probablité au moins $1-\delta$,*

$$\|\epsilon'_k\|_{1,\mu} = \mathcal{L}_{k-1}^{\Pi}(\mu;\pi_k) \leq \inf_{\pi\in\Pi} \mathcal{L}_{k-1}^{\Pi}(\mu;\pi) + 2\big(e'_1(N',\delta) + e'_2(N',M,\delta)\big),$$

$$\begin{aligned} \text{où} \qquad e'_1(N',\delta) &= 16 V_{\max}\sqrt{\frac{2}{N'}\big(h\log\frac{eN'}{h} + \log\frac{32}{\delta}\big)}\,, \\ e'_2(N',M,\delta) &= 8 V_{\max}\sqrt{\frac{2}{MN'}\big(h\log\frac{eMN'}{h} + \log\frac{32}{\delta}\big)}\,. \end{aligned}$$

8. L'extension au cas de plus de deux actions peut se faire en suivant la démarche décrite par Lazaric *et al.* (2012). La différence principale par rapport au cas deux action est que la VC-dimension de l'espace des politiques est remplacée par la dimension de Natarajan.

Ici encore, la forme de ce résultat est conforme aux résultats standards d'analyse de la littérature *apprentissage statistique*. En utilisant le fait que $\|\epsilon_k\|_{1,\mu} \leq \|\epsilon_k\|_{2,\mu}$, les deux lemmes que nous venons de décrire montrent que l'on contrôle les erreurs ϵ et ϵ' du théorème 4.7 en norme ℓ_1 ; en prenant $p = 1$, $\alpha' = 1$ et $\alpha = \infty$, on en déduit la garantie suivante.

Théorème 4.12. *Soit*

$$d' = \sup_{g \in \mathcal{F}, \pi' \in \Pi} \inf_{\pi \in \Pi} \mathcal{L}^{\Pi}_{\pi', g}(\mu; \pi) \qquad \text{et} \qquad d_m = \sup_{g \in \mathcal{F}, \pi \in \Pi} \inf_{f \in \mathcal{F}} \|(T_\pi)^m g - f\|_{2,\mu}$$

où $\mathcal{F}$ et Π sont respectivement l'espace fonctionnel (linéaire) et l'espace de classifieurs utilisés par CBMPI. Avec les hypothèses et notations du théorème 4.7 et des lemmes 4.10 et 4.11, après k itérations, avec probabilité $1 - \delta$, la perte espérée $\mathbb{E}_\rho[l_k] = \|l_k\|_{1,\rho}$ de CBMPI satisfait :

$$\begin{aligned}\|l_k\|_{1,\rho} \leq & \frac{2\gamma^m(\gamma - \gamma^{k-1})\mathcal{C}^{2,k,m}_\infty}{(1-\gamma)^2}\left(d_m + e_1(N, \frac{\delta}{2k}) + e_2(N, \frac{\delta}{2k})\right) \\ & + \frac{(1-\gamma^k)\mathcal{C}^{1,k,0}_\infty}{(1-\gamma)^2}\left(d' + e'_1(N', \frac{\delta}{2k}) + e'_2(N', M, \frac{\delta}{2k})\right) + g(k).\end{aligned}$$

Remarque 4.13. *La borne ci-dessus permet de donner une version quantitative de la remarque qualitative 4.9. Si on suppose un budget (en nombre d'échantillons de transitions) par itération fixe $B = Nm + N'M|A|(m+1)$ équilibré entre le classifieur et le régresseur, alors la borne ci-dessus a la forme suivante :*

$$\|l_k\|_{1,\mu} \leq \tilde{O}\left(\gamma^m\left(d_m + \sqrt{\frac{m}{B}}\right) + d' + \sqrt{\frac{|A|m}{B}}(\sqrt{n} + \sqrt{M})\right).$$

On voit ici clairement le rôle joué par le paramètre m : une grande valeur de m réduit l'influence des erreurs (d'approximation et d'estimation) dues au régresseur—le premier terme—mais augmente en même temps l'erreur d'estimation du classifieur—le troisième et dernier terme—, l'erreur liée au choix d'espace de politique Π restant incompressible.

4.3 Résultats empiriques

Pour finir, nous présentons quelques résultats empiriques qui montrent l'intérêt du schéma algorithmique considéré dans ce chapitre. Des trois variations algorithmiques AMPI-V, AMPI-Q et CBMPI, la dernière est celle qui s'est avérée la plus intéressante en pratique ; nous nous concentrerons de nouveau sur elle. Nous en profiterons notamment pour illustrer l'influence du paramètre m. Pour des résultats empiriques plus variés et détaillés, et notamment des illustrations des algorithmes AMPI-V et AMPI-Q, nous renvoyons le lecteur à l'article (Scherrer *et al.*, 2015).

Nous considérons le jeu de Tetris décrit en introduction (Exemple 0.1 page 7) et allons comparer trois algorithmes : CBMPI, DPI (Lazaric *et al.*, 2010)—qui est équivalent à CBMPI lorsque m tend vers l'infini, et dont l'implémentation pratique (Lagoudakis et Parr, 2003a; Fern *et al.*, 2006) équivaut à CBMPI avec une fonction valeur nulle—et la méthode d'*entropie croisée* (Rubinstein et Kroese, 2004) (CE, pour Cross Entropy en anglais), méthode d'optimisation "boîte noire" similaire au recuit similé qui est une référence pour le domaine de Tetris (Szita et Lőrincz, 2006; Thiéry et Scherrer, 2009b).

Afin de spécifier les divers algorithmes, nous allons nous appuyer sur des espaces de fonctions linéaires pour définir un espace de fonctions valeurs $\mathcal{F}$ et un espace de politiques Π. De manière similaire à ce que nous avons fait dans les deux chapitres précédents, nous considérons pour les

fonctions valeurs l'espace $\mathcal{F}$ de fonctions de l'ensemble des triplets (mur, pièce, action) $S \times P \times A$ dans $\mathbb{R}$ paramétrées par un vecteur $\theta \in \mathbb{R}^d$:

$$\forall (s,p,a), \quad f_\theta(s,p,a) = \phi(s,p,a)'\theta = \sum_{j=1}^{d} \theta_j \phi_j(s,p,a),$$

où les ϕ_j sont des fonctions de base qui ont pour vocation à *résumer* l'information correspondant à la situation et la décision immédiate à tout instant d'une partie de Tetris. En pratique, nous choisissons un sous-ensemble des fonctions de bases parmi les 14 fonctions suivantes :

(i) Les fonctions $\phi_1, \ldots, \phi_5$ pour coder la hauteur maximale : Ces 5 fonctions, qui dépendent uniquement du mur courant s, proposées dans (Scherrer *et al.*, 2015), permettent de décrire de manière extensive la hauteur moyenne du mur.

(ii) Les fonctions $\phi_6, \ldots, \phi_{14}$ de Dellacherie-Thiery (D-T) : Cet ensemble est constitué de 6 fonctions proposées par Dellacherie (Fahey, 2003) qui a longtemps été la référence des bots jouant à Tetris (Thiéry et Scherrer, 2009a), et de 3 ajoutées par Thiéry et Scherrer (2009b). Contrairement aux précédentes, ces fonctions dépendent de l'action effectuée et du mur obtenu *après cette action* : la *hauteur d'arrivée de la pièce*, le *nombre de briques supprimées*, le *nombre de transitions horizontales "brique ↔ non-brique"*, le *nombre de transitions verticales "brique ↔ non-brique"*, le *nombre de trous*, deux *mesures de la profondeurs des trous*[9] et une *mesure de la capacité de la surface supérieure du mur à absorber les pièces sans créer de trous.*

Ainsi, si on utilise l'ensemble des fonctions sur dessus, la valeur en un couple état-action (s,p,a) est approchée par

$$\hat{q}(s,p,a) = \sum_{j=1}^{5} \theta_j \phi(s) + \sum_{j=6}^{14} \theta_j \phi(s,p,a).$$

L'espace des politiques Π utilisé par CBMPI est défini *implicitement* par le même ensemble de fonctions de bases pour former une fonction "score" pour les actions, l'action choisie par la politique étant celle qui a le plus grand score. Par exemple, si nous utilisons toutes les fonctions de base, nous considérons l'ensemble des politiques qui en l'état (s,p) renvoient une action telle que

$$\pi(s,p) \in \arg\max_{a \in A(p)} \sum_{j=1}^{5} \theta'_j \phi(succ(s,p,a)) + \sum_{j=6}^{14} \theta'_j \phi(s,p,a)$$

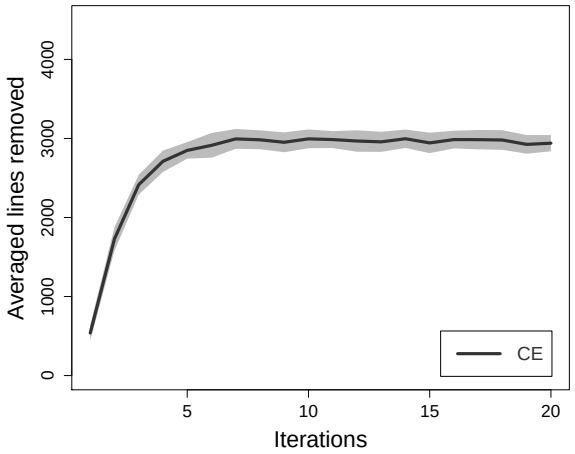
où les paramètres (θ'_j) sont potentiellement différents des (θ_j) utilisé pour la fonction valeur. Une différence subtile pour la définition de la fonction de score pour les politiques est que les premières fonctions de bases $(\phi_1, \ldots, \phi_5)$, qui ne sont originellement définies que sur l'état courant du mur, sont évaluées non en l'état courant mais en l'état successeur $succ(s,p,s)$, ce qui permet d'avoir une dépendance en fonction de l'action a.

Dans les expériences sur le domaine de Tetris, nous considérons deux variantes du jeu : le jeu normal qui se joue sur une zone de jeu de taille 20×10, et une variante réduite de taille 10×10 (de hauteur deux fois moindre). Les politiques générées par les différents algorithmes sont évaluées par leur score (le nombre de lignes supprimées dans une partie commençant sur un mur vide) moyenné sur 200 parties pour la petite variante du jeu (10×10) et sur 20 parties pour le jeu normal (20×10). Tous les algorithmes considérés étant itératifs, on présente des courbes "d'apprentissage" qui donnent l'évolution des performances au cours des itérations : on y présente la moyenne et un intervalle de confiance correspondant à 3 fois la déviation standard. On peut directement contrôler le nombre B d'échantillons utilisés par CBMPI et DPI

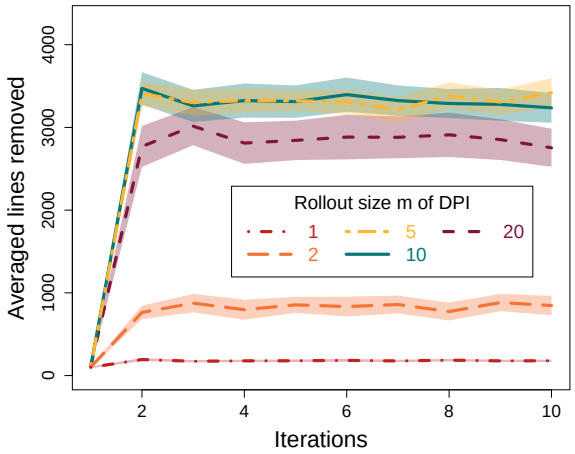
9. Voir (Thiéry et Scherrer, 2009a) pour plus de détails.

en choisissant le nombre de trajectoires NM et leur longueur m ; on a en effet $B = (m+1)NM|A|$ échantillons [10]. Par contre, avec une approche d'optimisation "boîte noire", où l'évaluation d'un contrôleur se fait en jouant une ou plusieurs parties, le nombre total d'échantillons n'est pas contrôlé mais mesuré *a posteriori*.

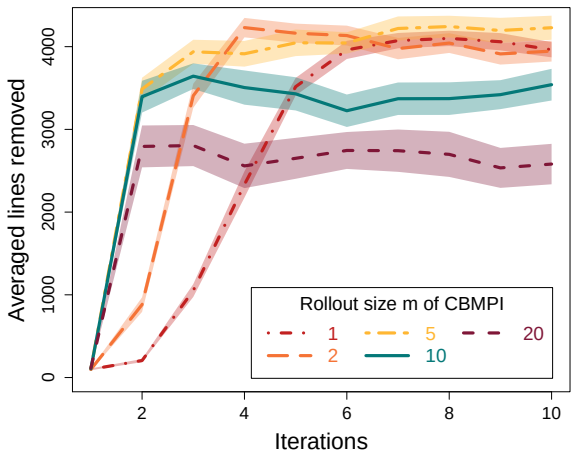
Nous avons effectué des expériences sur la petite variante du jeu, ce qui nous permet de comprendre le rôle des paramètres principaux et des fonctions pour la représentation paramétrée. Nous avons ensuite utilisé les meilleurs paramètres dans la variante plus grande (normale) du jeu.



(a) Entropie croisée (CE).



(b) DPI avec un budget de $B = 8,000,000$ échantillons par itération, pour les valeurs $m = \{1, 2, 5, 10, 20\}$.



(c) CBMPI avec un budget de $B = 8,000,000$ échantillons par itération, pour les valeurs $m = \{1, 2, 5, 10, 20\}$.

FIGURE 4.4 – Courbes d'apprentissage pour CE, DPI et CBMPI pour le jeu de taille 10×10.

Tetris, taille réduite (10×10) La figure 4.4 montre les courbes d'apprentissage pour les différents algorithmes sur la version réduite du jeu. Nous avons paramétré les politiques des trois algorithmes à l'aide des fonctions D-T. Pour CBMPI, quelques expérimentations nous

10. En pratique ici, nous avons réutilisé les échantillons de l'étape gloutonne pour l'étape d'évaluation.

ont amenés à choisir l'intégralité des fonctions $\phi_1, \ldots, \phi_{14}$ pour représenter la fonction valeur. L'algorithme CE atteint le score moyen de $3,000$ après 10 itérations utilisant un budget moyen de $B = 65,000,000$. Dans ces expériences, nous avons fixé à $B = 8,000,000$ le nombre d'échantillons par itération utilisés par CBMPI et DPI. Nous avons fait varier la longueur m des trajectoires dans $\{1, 2, 5, 10, 20\}$, fixé le nombre de répétitions par action $M = 1$ (qui s'est avéré la meilleure valeur), et déduit le nombre total N de trajectoires en conséquence. DPI atteint 3400 lignes en moyenne pour les valeurs $m = 5$ et $m = 10$. Pour DPI, des trajectoires trop courtes ($m = 1$) donnent de mauvaises estimations de $\widehat{q}$, tandis que de trop longues trajectoires ($m = 20$) plombent l'erreur d'estimation. L'algorithme CBMPI est celui qui donne les meilleures résultats, en atteignant un score moyen de $4,200$ pour $m = 5$ (soit $N = \frac{8000000}{(5+1)\times 32} \approx 42,000$). Contrairement à DPI, CBMPI obtient une performance raisonnable avec de courtes trajectoires ; ceci suggère que la fonction valeur est relativement bien représentée, et donc construit de meilleurs estimations de $\widehat{q}$.

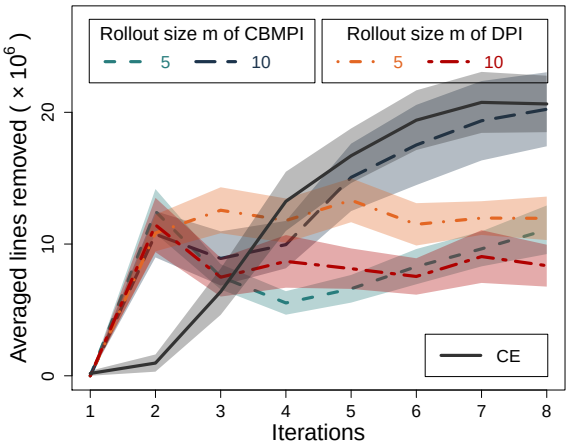


FIGURE 4.5 – Courbes d'apprentissage pour CBMPI, DPI et CE pour le jeu 10×20.

Tetris, taille normale (10×20) La figure 4.5 montre la performance des algorithmes décrits plus haut dans le jeu de taille normale. Ici, nous avons fixé pour DPI et CBMPI un budget par itération de $B = 32,000,000$. DPI atteint son meilleur score, $12,000,000$, après 3 itérations pour la valeur $m = 5$. CBMPI atteint le score $20,000,000$ après 8 itérations pour la valeur $m = 10$. CE atteint une performance égale à celle de CBMPI, un score moyen de $20,000,000$ après 8 itérations ; cependant, le nombre total d'échantillons utilisés par cette dernière méthode est comparativement 6 fois plus élevé : après 8 itérations (nombre d'itérations pour que les deux méthodes atteignent leur meilleur score) CBMPI utilise *seulement* $256,000,000$ échantillons tandis que CE en utilise $1,700,000,000$.

4.4 Bilan et perspectives

Nous avons considéré une famille de schémas algorithmiques, MPI, pour approcher la solution de MDP de grande taille. Nous avons décrits trois implémentations originales, AMPI-V, AMPI-Q et CBMPI, qui généralisent un certain nombre d'algorithmes de la littérature : fitted-value iteration, fitted-Q iteration, et classification-based policy iteration. Nous avons décrit nos principaux résultats d'analyse de propagation d'erreur, résultats qui ont le bon goût d'unifier ceux connus pour API et AVI. Nous avons détaillé cette analyse d'erreur pour l'algorithme

CBMPI et obtenu une garantie de performance qui dépend de la qualité des espaces choisis pour représenter les politiques et les valeurs, ainsi que le nombre d'échantillons utilisés. Cette analyse éclaire notamment le rôle du paramètre principal m du schéma algorithmique : à nombre d'échantillons constant, ce dernier permet de faire un compromis entre la qualité de la représentation de la valeur et l'erreur d'estimation de la politique gloutonne. Des simulations numériques sur le jeu de Tetris ont permis de confirmer cette interprétation qualitative. En particulier, l'algorithme CBMPI s'est montré plus efficace (en terme du nombre d'échantillons) pour construire des contrôleurs de qualité pour ce jeu qu'une approche d'optimisation "boîte noire" qui était alors la référence.

A l'exception d'AMPI-Q (qui n'est pas l'algorithme qui a montré les meilleures performances empiriques), les schémas numériques que nous avons présentés concernent le cas où le nombre d'actions est fini et petit. Développer et analyser des schémas pour des grands espace d'actions (par exemple continus sous une hypothèse de régularité) constitue une perspective intéressante. Nous avons fait la remarque que nos résultats unifiés ne dépendent pas du paramètre principal m. Néanmoins, les étapes qui amènent à ce résultat dépendent de m, et nous pensons qu'il doit être possible de mener une preuve plus simple où m n'apparaît pas. Nous n'avons pas été en mesure de le faire et cela constitue une piste naturelle de recherche. Le sous-problème de classification utilisé par CBMPI est une instance de classification multi-classe à coût sensitif. Si le cas de deux classes est bien compris dans la littérature, le cas "k classes" n'admet à notre connaissance pas de solution standard. Approfondir ce problème est une direction envisageable. Enfin, on trouve dans la littérature d'autres schémas itératifs pour résoudre les problèmes de contrôle optimal, notamment ceux décrits par Yu et Bertsekas (2013) et Rogers (2007). Appliquer le même genre d'analyse de sensibilité fait partie des pistes pour des travaux futurs.

Chapitre 5

Sur l'utilisation de politiques non-stationnaires

Nous avons décrit dans le précédent chapitre une famille de schémas algorithmiques approchés qui tentent, à chaque itération, de suivre ce que feraient des algorithmes exacts de programmation dynamique. Dans le cas des deux algorithmes les plus standards, rappelons que cela revient à considérer des schémas qui construisent des séquences de fonctions valeurs v_k ou politiques π_k comme suit

$$\text{Itérations sur les valeurs approché (AVI) :} \qquad v_{k+1} \leftarrow Tv_k + \epsilon_{k+1} \tag{5.1}$$

$$\text{Itérations sur les politiques approché (API) :} \qquad \left\{ \begin{array}{rcl} v_k & \leftarrow & v_{\pi_k} + \epsilon_k \\ \pi_{k+1} & \leftarrow & \text{un élément de } \mathcal{G}v_k \end{array} \right. \tag{5.2}$$

où v_0 et π_0 sont arbitraires, et où l'on rappelle que T est l'opérateur d'optimalité de Bellman, v_{π_k} est la valeur de la politique π_k, et $\mathcal{G}v_k$ est l'ensemble des politiques gloutonnes par rapport à v_k. A chaque itération k, le terme ϵ_k rend compte d'une approximation de l'opérateur de Bellman (pour AVI) ou de l'évaluation de v_{π_k} (pour API). L'étude précise de ces erreurs pour des implémentations particulières de ces approximations a été (notamment) abordé dans les deux chapitres précédents. Dans ce chapitre, nous ferons pour simplifier l'hypothèse que pour tout k, les termes d'erreurs ϵ_k sont contrôlés en norme infinie : $\|\epsilon_k\|_\infty \leq \epsilon$ avec $\epsilon \geq 0$. Des résultats d'analyse de sensibilité aux erreurs que nous avons généralisés dans le précédent chapitre, sont historiquement dûs à Singh et Yee (1994); Gordon (1995); Bertsekas et Tsitsiklis (1996) pour AVI et Bertsekas et Tsitsiklis (1996) pour API :

Théorème 5.1. *Pour API (respectivement AVI), la perte liée au fait de suivre la politique π_k (respectivement une politique π_k de $\mathcal{G}v_{k-1}$) au lieu de la politique optimale π_* satisfait :*

$$\limsup_{k\to\infty} \|v_* - v_{\pi_k}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2}\epsilon.$$

Dans ce chapitre, nous allons nous intéresser particulièrement à la dépendance de ces garanties en fonction du facteur d'actualisation γ. Les résultats présentés ici ont été obtenus avec Boris Lesner et ont donné lieu aux deux publications suivantes : (Scherrer et Lesner, 2012) et (Lesner et Scherrer, 2015). Le coefficient $\frac{2\gamma}{(1-\gamma)^2}$ peut être très grand, et ce d'autant plus que γ est en pratique souvent proche de 1. Ainsi, la garantie ci-dessous est généralement perçue comme étant pessimiste en pratique. De manière intéressante, cette constante $\frac{2\gamma}{(1-\gamma)^2}$ apparaît dans de nombreux travaux de la littérature sur AVI (Singh et Yee, 1994; Gordon, 1995; Tsitsiklis et Roy, 1996; Guestrin *et al.*, 2001, 2003; Pineau *et al.*, 2003; Even-dar, 2005; Ernst *et al.*, 2005;

Munos, 2007; Munos et Szepesvári, 2008; Petrik et Scherrer, 2008; Farahmand *et al.*, 2010) et API (Kakade et Langford, 2002; Munos, 2003; Lagoudakis et Parr, 2003b; Antos *et al.*, 2008; Farahmand *et al.*, 2008; Maillard *et al.*, 2010; Busoniu *et al.*, 2011; Lazaric *et al.*, 2012; Gabillon *et al.*, 2011; Bertsekas, 2011; Farahmand *et al.*, 2010; Azar *et al.*, 2011), ainsi que dans des travaux auxquels j'ai contribué sur les généralisations λ-PI et AMPI mentionnées au chapitre précédent (Scherrer, 2013b; Thiéry et Scherrer, 2010; Scherrer et Thiéry, 2010; Scherrer *et al.*, 2015), suggérant qu'elle a peu de chance d'être améliorée. En effet, la borne ci-dessus (et la constante $\frac{2\gamma}{(1-\gamma)^2}$) est optimale pour API (Bertsekas et Tsitsiklis, 1996, Exemple 6.4), et nous avons montré avec Boris Lesner qu'elle l'est également pour AVI (section 5.1) et pour AMPI (section 5.3), résultats qui n'étaient à notre connaissance pas connus dans la littérature.

Bien que la théorie du contrôle optimal à horizon infini assure qu'il existe une politique optimale qui est stationnaire, l'idée centrale des contributions présentées ici est de montrer qu'il peut être préférable de chercher à construire une politique *non-stationnaire* dans la mesure où elles jouissent d'une meilleure garantie "pire cas". De plus, nous allons montrer que ces politiques non-stationnaires peuvent être obtenues via des modifications mineures des algorithmes standards. Nous montrons comment déduire une politique non-stationnaire à partir de l'exécution d'une instance d'AVI. Nous décrivons ensuite deux variations d'API qui permettent de calculer des politiques non-stationnaires. Nous finirons par présenter un schéma qui permet d'interpoler entre ces variations non-stationnaires d'AVI et d'API d'une manière similaire à AMPI. Pour l'ensemble de ces algorithmes approchés avec une erreur par itération bornée par ϵ, nous fournissons des garanties de performance qui peuvent être réduites jusqu'à $\frac{2\gamma}{1-\gamma}\epsilon$, c'est-à-dire d'un facteur multiplicatif $\frac{1}{1-\gamma}$ meilleur que la garantie du théorème 5.1, ce qui est significatif dans les situations typiques ou γ est proche de 1. Ainsi, une conséquence quelque peu inattendue de ces résultats est que le problème de "calculer des politiques approximativement optimales *non-stationnaires*" est sensiblement plus simple que celui de "calculer des politiques approximativement optimales *stationnaires*." Enfin, nous montrons que les garanties améliorées obtenues pour ces algorithmes non-stationnaires sont optimales dans le sens où elles ne peuvent pas en général être améliorées. Avant de décrire ces contributions, nous allons commencer par introduire quelques notations utiles pour les politiques non-stationnaires.

Etant donnée une séquence $\pi_1, \pi_2, \ldots, \pi_k$ de k politiques stationnaires (cette séquence sera claire dans les contextes que nous aborderons plus loin), et pour tout entier $1 \leq \ell \leq k$, nous noterons $\pi_{k,\ell}$ la politique *périodique non-stationnaire* qui choisit la première action selon π_k, la deuxième selon π_{k-1}, ..., la $\ell^{\text{ième}}$ selon $\pi_{k-\ell+1}$ et qui boucle indéfiniment sur cette séquence de taille ℓ. Formellement, on écrira :

$$\pi_{k,\ell} = \pi_k \ \pi_{k-1} \ \cdots \ \pi_{k-\ell+1} \ \pi_k \ \pi_{k-1} \ \cdots \pi_{k-\ell+1} \cdots$$

Il est immédiat de généraliser la notion de fonction valeur à ce type de politiques, notée $v_{\pi_{k,\ell}}$, et de voir qu'elle est l'unique point fixe de l'opérateur composé suivant :

$$\forall v, \quad T_{k,\ell} v = T_{\pi_k} \ T_{\pi_{k-1}} \ \cdots \ T_{\pi_{k-\ell+1}} v.$$

Enfin, il sera utile d'introduire le noyau actualisé suivant :

$$\Gamma_{k,\ell} = (\gamma P_{\pi_k})(\gamma P_{\pi_{k-1}}) \cdots (\gamma P_{\pi_{k-\ell+1}}).$$

Généralisant le cas *stationnaire*, on pourra facilement observer le fait que l'opérateur $T_{k,\ell}$ est affine et que pour toute paire de fonctions v et v', on a les identités suivantes :

$$T_{k,\ell} v - T_{k,\ell} v' = \Gamma_{k,\ell}(v - v') \tag{5.3}$$

$$\text{et} \quad T_{k,\ell}(v + v') = T_{k,\ell} v + \Gamma_{k,\ell} v'. \tag{5.4}$$

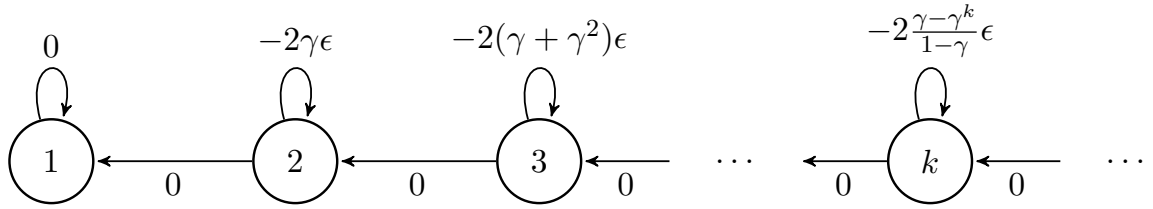


FIGURE 5.1 – Le MDP déterministe pour lequel la borne du théorème 5.1 est optimale pour AVI et API.

5.1 Optimalité de la constante $\frac{2\gamma}{(1-\gamma)^2}$ pour les politiques stationnaires

La garantie du théorème 5.1 est optimale pour API dans le sens où il existe une instance de MDP (Bertsekas et Tsitsiklis, 1996, Exemple 6.4) et une séquence d'erreurs $(\epsilon_k)_{k\geq 1}$ satisfaisant $\|\epsilon_k\|_\infty \leq \epsilon$ tels que la borne est satisfaite avec égalité. A notre connaissance, une observation similaire n'était pas répertoriée dans la littérature pour AVI. Comme nous allons le voir, l'exemple utilisé pour montrer l'optimalité de la borne pour API permet de montrer que la borne est également optimale pour AVI. Ce MDP déterministe, tiré de (Bertsekas et Tsitsiklis, 1996, Exemple 6.4), est représenté à la figure 5.1 et décrit ci-après.

Exemple 5.2. *On considère un MDP avec un nombre infini d'états* $1, 2, \ldots$. *L'état* 1 *est absorbant et a une récompense nulle. Pour tout état* $i > 1$, *il y a deux actions possibles : soit on* bouge *vers l'état* $i-1$ *avec une récompense nulle, soit on* reste *dans l'état* i *avec une récompense*

$$r_i = -2\frac{\gamma - \gamma^i}{1-\gamma}\epsilon.$$

Pour simplifier légèrement ce qui suit, on considèrera qu'en l'état 1 *absorbant, seule l'action "*reste*" est disponible (on a bien* $r_1 = 0$*). Comme les récompenses non nulles sont strictement négatives, il est clair que l'unique politique optimale consiste à toujours choisir l'action "*bouge*" qui permet de passer de l'état* i *à l'état* $i-1$ *jusqu'à l'état absorbant* 1, *et ainsi que la valeur optimale* v_* *vaut* 0 *en tout état.*

On considère ici qu'on initialise AVI avec $v_0 = v_* = 0$. Enfin, nous allons supposer que la séquence d'erreurs (ϵ_k) est définie, pour tout $k > 0$, par :

$$\epsilon_k(i) = \begin{cases} -\epsilon & \text{si } i = k, \\ \epsilon & \text{si } i = k+1, \\ 0 & \text{sinon.} \end{cases}$$

Avec l'instance que nous venons de décrire, nous allons montrer que pour toute itération k, parmi les politiques gloutonnes $\mathcal{G}v_k$ par rapport à la valeur v_k, il en existe une, π_{k+1}, qui choisit l'action "*bouge*" dans tous les états sauf l'état $k+1$, où elle choisit "*rester*". Ainsi, en cet état $k+1$, on a une boucle impliquant une récompense négative, et la valeur de cette politique en cet état satisfait :

$$v_{\pi_{k+1}}(k+1) = \sum_{t=0}^{\infty} \gamma^t r_{k+1} = \frac{r_{k+1}}{1-\gamma} = -2\frac{\gamma - \gamma^{k+1}}{(1-\gamma)^2}\epsilon.$$

Autrement dit, comme $v_* = 0$, on voit qu'on atteint la borne du théorème 5.1 à la limite où k tend vers l'infini.

Pour montrer qu'on peut effectivement obtenir cette séquence de politiques, nous allons prouver par récurrence sur k que, pour tout $k \geq 1$,

$$v_k(i) = \begin{cases} -\gamma^{k-1}\epsilon & \text{si } i < k, \\ r_k/2 - \epsilon & \text{si } i = k, \\ -(r_k/2 - \epsilon) & \text{si } i = k+1, \\ 0 & \text{sinon.} \end{cases}$$

Comme $v_0 = v_* = 0$, une politique gloutonne est clairement de "*bouger*" en chaque état $i \geq 2$, et l'on obtient $v_1 = v_0 + \epsilon_1 = \epsilon_1$, ce qui prouve que la relation ci-dessus est vraie pour $k = 1$. Supposons à présent qu'elle est vraie à l'étape k et montrons qu'elle est également vraie à l'étape $k+1$.

Notons q_k^{b} la valeur si la première action effectuée est "*bouge*" et si on obtient ensuite la valeur donnée par v_k. Pour tout $i > 1$, on a $q_k^{\mathrm{b}}(i) = 0 + \gamma v_k(i-1)$, et ainsi

$$q_k^{\mathrm{b}}(i) = \begin{cases} \gamma(-\gamma^{k-1}\epsilon) & = -\gamma^k\epsilon & \text{si } i = 2, \ldots, k \\ \gamma(r_k/2 - \epsilon) & = r_{k+1}/2 & \text{si } i = k+1 \\ -\gamma(r_k/2 - \epsilon) & = -r_{k+1}/2 & \text{si } i = k+2 \\ 0 & & \text{sinon.} \end{cases}$$

Notons maintenant q_k^{r} la valeur si la première action effectuée est "*reste*" et si on obtient ensuite la valeur donnée par v_k. Pour tout $i > 0$, on a $q_k^{\mathrm{r}}(i) = r_i + \gamma v_k(i)$, et ainsi

$$q_k^{\mathrm{r}}(i) = \begin{cases} r_i + \gamma(-\gamma^{k-1}\epsilon) & = r_i - \gamma^k\epsilon & \text{si } i = 1, \ldots, k-1 \\ r_k + \gamma(r_k/2 - \epsilon) & = r_k + r_{k+1}/2 & \text{si } i = k \\ r_{k+1} - r_{k+1}/2 & = r_{k+1}/2 & \text{si } i = k+1 \\ r_{k+2} + \gamma 0 & = r_{k+2} & \text{si } i = k+2 \\ 0 & & \text{sinon.} \end{cases}$$

Tout d'abord, comme $r_1 = 0$ et $\epsilon_{k+1}(1) = 0$, la valeur en l'état 1 satisfait, comme désiré, pour tout k,

$$v_{k+1}(1) = q_k^{\mathrm{r}}(1) + \epsilon_{k+1}(1) = -\gamma^k\epsilon.$$

Ensuite, come $r_i < 0$ pour tout $i > 1$, on a $q_k^{\mathrm{b}}(i) > q_k^{\mathrm{r}}(i)$ pour tous ces états sauf $i = k+1$ pour lequel on a

$$q_k^{\mathrm{b}}(k+1) = q_k^{\mathrm{r}}(k+1) = r_{k+1}/2.$$

En utilisant finalement le fait que

$$v_{k+1} = \max(q_k^{\mathrm{b}}, q_k^{\mathrm{r}}) + \epsilon_{k+1},$$

on déduit que la valeur v_{k+1} a la forme voulue, ce qui clôt la démonstration par récurrence.

Finalement, comme pour tout $i > 1$, on a $q_k^{\mathrm{b}}(i) \geq q_k^{\mathrm{r}}(i)$ avec égalité si et seulement si $i = k+1$, on déduit qu'il existe une politique π_{k+1} gloutonne par rapport à v_k qui choisit l'action optimale ("*bouge*") en chaque état excepté $k+1$ où elle choisit l'action "*reste*". Ceci conclut la preuve.

5.2 Algorithmes pour calculer des politiques non-stationnaires

Comme l'exemple 5.2 de la section précédente montre que la borne du Théorème 5.1 est optimale, le seul espoir pour obtenir une meilleure garantie implique de changer les algorithmes. Dans cette section, nous décrivons des modifications mineures d'AVI et d'API qui permettent de construire une politique non-stationnaire pour lesquelles on a une garantie effectivement améliorée.

NSVI : déduction d'une politique non-stationnaire à partir de l'exécution d'AVI. Bien que le schéma algorithmique AVI—décrit à l'équation (5.1)—est généralement vu comme générant une séquence de fonctions $v_0, v_1, \ldots, v_{k-1}$, on peut aussi le considérer comme générant implicitement une séquence de politiques [1] $\pi_1, \pi_2, \ldots, \pi_k$ telles que pour $i = 0, \ldots, k-1$,

$$\pi_{i+1} \in \mathcal{G}v_i.$$

Tandis que l'algorithme standard AVI renvoie uniquement la politique π_k, la variation NSVI proposée ici consiste simplement à changer ce qui est retourné par AVI : on renvoie la politique périodique non-stationnaire $\pi_{k,\ell}$ qui boucle sur les ℓ dernières politiques générées par AVI. Le résultat qui suit montre que c'est effectivement une bonne idée.

Théorème 5.3. *Après k itérations de NSVI, pour tout entier $1 \leq \ell \leq k$, la perte liée au fait de suivre la politique non-stationnaire $\pi_{k,\ell}$ au lieu de la politique optimale π_* satisfait :*

$$\|v_* - v_{\pi_{k,\ell}}\|_\infty \leq \frac{2}{1-\gamma^\ell}\left(\frac{\gamma - \gamma^k}{1-\gamma}\epsilon + \gamma^k \|v_* - v_0\|_\infty\right).$$

Lorsque $\ell = 1$ et k tend vers l'infini, on retrouve exactement le résultat du théorème 5.1. Quand $\ell > 1$, cette garantie est un facteur $\frac{1-\gamma^\ell}{1-\gamma}$ multiplicatif meilleur que la borne du théorème 5.1. Le choix de ℓ qui optimise la borne, $\ell = k$, consiste à boucler sur l'intégralité des politiques induites depuis le début de l'algorithme, et mène à la garantie suivante :

$$\|v_* - v_{\pi_{k,k}}\|_\infty \leq 2\left(\frac{\gamma}{1-\gamma} - \frac{\gamma^k}{1-\gamma^k}\right)\epsilon + \frac{2\gamma^k}{1-\gamma^k}\|v_* - v_0\|_\infty,$$

qui tend vers $\frac{2\gamma}{1-\gamma}\epsilon$ lorsque k tend vers l'infini.

Une étape importante pour la preuve de ce résultat réside dans le lemme suivant, qui implique notamment que pour une valeur de ℓ suffisamment grande, $v_k = Tv_{k-1} + \epsilon_k$ est une assez bonne approximation (à un facteur additif $\frac{\epsilon}{1-\gamma}$ près) de la valeur $v_{\pi_{k,\ell}}$ de la politique non-stationnaire $\pi_{k,\ell}$, tandis qu'en général c'est une approximation plus grossière de la valeur v_{π_k} de la dernière politique stationnaire π_k.

Lemme 5.4. *Après k itérations de NSVI, pour tout entier $1 \leq \ell \leq k$,*

$$\|Tv_{k-1} - v_{\pi_{k,\ell}}\|_\infty \leq \gamma^\ell \|v_{k-\ell} - v_{\pi_{k,\ell}}\|_\infty + \frac{\gamma - \gamma^\ell}{1-\gamma}\epsilon.$$

Preuve du lemme 5.4. La valeur de $\pi_{k,\ell}$ satisfait :

$$v_{\pi_{k,\ell}} = T_{\pi_k} T_{\pi_{k-1}} \cdots T_{\pi_{k-\ell+1}} v_{\pi_{k,\ell}}. \tag{5.5}$$

En utilisant la relation de récurrence $v_{k+1} = T_{\pi_{k+1}} v_k + \epsilon_{k+1}$ et l'identité (5.4), on déduit que la séquence de valeur générées par AVI satisfait :

$$T_{\pi_k} v_{k-1} = T_{\pi_k} T_{\pi_{k-1}} \cdots T_{\pi_{k-\ell+1}} v_{k-\ell} + \sum_{i=1}^{m-1} \Gamma_{k,i}\epsilon_{k-i}. \tag{5.6}$$

En soustrayant les équations (5.6) et (5.5), l'identité (5.3) nous permet de voir que

$$Tv_{k-1} - v_{\pi_{k,\ell}} = T_{\pi_k} v_{k-1} - v_{\pi_{k,\ell}} = \Gamma_{k,\ell}(v_{k-\ell} - v_{\pi_{k,\ell}}) + \sum_{i=1}^{m-1} \Gamma_{k,i}\epsilon_{k-i}$$

et le résultat suit en prenant la norme infinie et en exploitant le fait que $\|\Gamma_{k,i}\|_\infty = \gamma^i$. □

1. Une séquence de fonctions valeur peut induire plusieurs séquences de politiques gloutonnes dans la mesure où chacune de ces valeurs peut avoir plusieurs politique gloutonnes. Les résultats que nous décrivons ci-après sont valables pour n'importe quel choix parmi ces possibilités.

Nous pouvons maintenant prouver la garantie de performance.

Preuve du théorème 5.3. En utilisant le fait que T est une contraction de coefficient γ en norme infinie, on obtient :

$$\begin{aligned}\|v_* - v_k\|_\infty &= \|v_* - Tv_{k-1} + \epsilon_k\|_\infty \\ &\leq \|Tv_* - Tv_{k-1}\|_\infty + \epsilon \\ &\leq \gamma\|v_* - v_{k-1}\|_\infty + \epsilon.\end{aligned}$$

On en déduit, par récurrence sur k, que pour tout $k \geq 1$,

$$\|v_* - v_k\|_\infty \leq \gamma^k\|v_* - v_0\|_\infty + \frac{1-\gamma^k}{1-\gamma}\epsilon. \tag{5.7}$$

On conclut en utilisant le lemme 5.4 et (deux fois) l'équation (5.7) :

$$\begin{aligned}&\|v_* - v_{\pi_{k,\ell}}\|_\infty \\ &\leq \|Tv_* - Tv_{k-1}\|_\infty + \|Tv_{k-1} - v_{\pi_{k,\ell}}\|_\infty \\ &\leq \gamma\|v_* - v_{k-1}\|_\infty + \gamma^\ell\|v_{k-\ell} - v_{\pi_{k,\ell}}\|_\infty + \frac{\gamma-\gamma^\ell}{1-\gamma}\epsilon \\ &\leq \gamma\left(\gamma^{k-1}\|v_* - v_0\|_\infty + \frac{1-\gamma^{k-1}}{1-\gamma}\epsilon\right) + \gamma^\ell\left(\|v_{k-\ell} - v_*\|_\infty + \|v_* - v_{\pi_{k,\ell}}\|_\infty\right) + \frac{\gamma-\gamma^\ell}{1-\gamma}\epsilon \\ &\leq \gamma^k\|v_* - v_0\|_\infty + \frac{\gamma-\gamma^k}{1-\gamma}\epsilon + \gamma^\ell\left(\gamma^{k-\ell}\|v_* - v_0\|_\infty + \frac{1-\gamma^{k-\ell}}{1-\gamma}\epsilon + \|v_* - v_{\pi_{k,\ell}}\|_\infty\right) + \frac{\gamma-\gamma^\ell}{1-\gamma}\epsilon \\ &= \gamma^\ell\|v_* - v_{\pi_{k,\ell}}\|_\infty + 2\gamma^k\|v_* - v_0\|_\infty + \frac{2(\gamma-\gamma^k)}{1-\gamma}\epsilon \\ &\leq \frac{2}{1-\gamma^\ell}\left(\frac{\gamma-\gamma^k}{1-\gamma}\epsilon + \gamma^k\|v_* - v_0\|_\infty\right).\end{aligned}$$

□

Nous allons maintenant présenter des résultats similaires pour deux algorithmes de type API pour calculer des politiques non-stationnaires. Contrairement à AVI où seul ce qui est retourné *in fine* par l'algorithme a été modifié, les changements que nous allons opérer sur le schéma standard API sont légèrement plus compliqués.

NSPI($\mathbb{N}$) : une variation d'API pour calculer une politique non-stationnaire avec période croissante. En s'inspirant des observations faites dans le paragraphe précédent au sujet d'AVI, nous considérons tout d'abord une variation d'API, NSPI($\mathbb{N}$), qui, à chaque itération, approche non pas la valeur de la dernière politique stationnaire π_k mais celle de la politique périodique non-stationnaire $\pi_{k,k}$ qui boucle sur l'ensemble des politiques $\pi_1, \ldots, \pi_k$ générées depuis le début de l'algorithme :

$$\begin{aligned}v_k &\leftarrow v_{\pi_{k,k}} + \epsilon_k \\ \pi_{k+1} &\leftarrow \text{un élément de } \mathcal{G}v_k\end{aligned}$$

où la politique initiale $\pi_{1,1}$ est choisie arbitrairement, et où l'on rappelle que $v_{\pi_{k,k}}$ est le point fixe de l'opérateur $T_{\pi_{k,k}}$. Ainsi, itération après itération, la politique non-stationnaire $\pi_{k,k}$ est constitué d'un nombre de plus en plus grand de politiques stationnaires et sa période est croissante. Cet algorithme jouit de la garantie suivante.

Théorème 5.5. *Après k itérations de NSPI($\mathbb{N}$), la perte liée au fait de suivre la politique non-stationnaire $\pi_{k,k}$ au lieu de la politique optimale π_* satisfait :*

$$\|v_* - v_{\pi_{k,k}}\|_\infty \leq \frac{2(\gamma - \gamma^k)}{1-\gamma}\epsilon + \gamma^{k-1}\|v_* - v_{\pi_{1,1}}\|_\infty + 2(k-1)\gamma^k V_{\max}.$$

Lorsque k tend vers l'infini, cette borne tend vers $\frac{2\gamma}{1-\gamma}\epsilon$, ce qui est un facteur multiplicatif $\frac{1}{1-\gamma}$ meilleure que la borne standard d'API donnée au théorème 5.1.

Preuve du théorème 5.5. En utilisant le fait que $T_{k+1,k+1}v_{\pi_{k,k}} = T_{\pi_{k+1}}T_{k,k}v_{\pi_{k,k}} = T_{\pi_{k+1}}v_{\pi_{k,k}}$ et que $T_{\pi_{k+1}}v_k \geq T_{\pi_*}v_k$ (puisque $\pi_{k+1} \in \mathcal{G}v_k$), on a :

$$\begin{aligned}
&v_* - v_{\pi_{k+1,k+1}} \\
&= T_{\pi_*}v_* - T_{k+1,k+1}v_{\pi_{k+1,k+1}} \\
&= T_{\pi_*}v_* - T_{\pi_*}v_{\pi_{k,k}} + T_{\pi_*}v_{\pi_{k,k}} - T_{k+1,k+1}v_{\pi_{k,k}} + T_{k+1,k+1}v_{\pi_{k,k}} - T_{k+1,k+1}v_{\pi_{k+1,k+1}} \\
&= \gamma P_{\pi_*}(v_* - v_{\pi_{k,k}}) + T_{\pi_*}v_{\pi_{k,k}} - T_{\pi_{k+1}}v_{\pi_{k,k}} + \Gamma_{k+1,k+1}(v_{\pi_{k,k}} - v_{\pi_{k+1,k+1}}) \\
&= \gamma P_{\pi_*}(v_* - v_{\pi_{k,k}}) + T_{\pi_*}v_k - T_{\pi_{k+1}}v_k + \gamma(P_{\pi_{k+1}} - P_{\pi_*})\epsilon_k + \Gamma_{k+1,k+1}(v_{\pi_{k,k}} - v_{\pi_{k+1,k+1}}) \\
&\leq \gamma P_{\pi_*}(v_* - v_{\pi_{k,k}}) + \gamma(P_{\pi_{k+1}} - P_{\pi_*})\epsilon_k + \Gamma_{k+1,k+1}(v_{\pi_{k,k}} - v_{\pi_{k+1,k+1}}).
\end{aligned}$$

En prenant la norme infinie, et en utilisant le fait que $\|v_{\pi_{k,k}}\|_\infty \leq V_{\max}$, $\|v_{\pi_{k+1,k+1}}\|_\infty \leq V_{\max}$ et que $\|\Gamma_{k+1,k+1}\|_\infty = \gamma^{k+1}$, on obtient :

$$\|v_* - v_{\pi_{k+1,k+1}}\|_\infty \leq \gamma\|v_* - v_{\pi_{k,k}}\|_\infty + 2\gamma\epsilon + 2\gamma^{k+1}V_{\max}.$$

Enfin, une récurrence sur k nous permet de conclure que

$$\|v_* - v_{\pi_{k,k}}\|_\infty \leq \frac{2(\gamma - \gamma^k)}{1-\gamma}\epsilon + \gamma^{k-1}\|v_* - v_{\pi_{1,1}}\|_\infty + 2(k-1)\gamma^k V_{\max}. \qquad \square$$

Bien qu'il ait une garantie asymptotique améliorée, l'algorithme que nous venons de décrire a plusieurs inconvénients :

1. la borne de garantie après un nombre fini d'itérations contient un terme inélégant de la forme $2(k-1)\gamma^k V_{\max}$;
2. même lorsqu'il n'y a pas d'erreur (quand $\epsilon = 0$) on ne peut pas garantir, comme pour Itérations sur les politiques, qu'il génère une séquence de politiques de valeurs croissantes et qu'il converge en un temps fini (on peut d'ailleurs facilement voir que ce n'est pas le cas !) ;
3. en pratique, la mémoire requise croît avec le nombre d'itérations

L'algorithme que nous allons présenter ci-après permet de pallier ces problèmes.

NSPI(ℓ) : une variation d'API pour calculer une politique non-stationnaire avec période fixe. Nous considérons à présent une variation d'API, NSPI(ℓ), paramétrée par un entier $\ell \geq 1$, qui itère de la manière suivante :

$$\begin{aligned}
v_k &\leftarrow v_{\pi_{k,\ell}} + \epsilon_k \\
\pi_{k+1} &\leftarrow \text{un élément de } \mathcal{G}v_k
\end{aligned}$$

où la politique non-stationnaire initiale $\pi_{0,\ell}$ est construite à partir d'un ensemble arbitraire de ℓ politiques stationnaires $\pi_{-\ell+1}, \pi_{-\ell+2}, \ldots, \pi_{-1}, \pi_0$. Contrairement à l'algorithme présenté

dans le paragraphe précédent, la politique non-stationnaire implique seulement les ℓ dernières politiques stationnaires construites par l'algorithme. On pourra remarquer de plus que nous venons de décrire une stricte généralisation du schéma standard API, sur lequel on retombe en prenant $\ell = 1$. Cet algorithme jouit de la garantie suivante.

Théorème 5.6. *Pour tout ℓ, après k itérations de NSPI(ℓ), la perte liée au fait de suivre la politique non-stationnaire $\pi_{k,\ell}$ au lieu de la politique optimale π_* satisfait :*

$$\|v_* - v_{\pi_{k,\ell}}\|_\infty \leq \gamma^k \|v_* - v_{\pi_{0,\ell}}\|_\infty + \frac{2(\gamma - \gamma^{k+1})}{(1-\gamma)(1-\gamma^\ell)}\epsilon.$$

Lorsque $\ell = 1$ et k tend vers l'infini, on retrouve exactement la borne du théorème 5.1. Quand $\ell > 1$ et k tend vers l'infini, cette borne coïncide avec celle que nous avons décrite plus haut pour AVI (Théorème 5.3) : elle est un facteur multiplicatif $\frac{1-\gamma^\ell}{1-\gamma}$ meilleur que la borne standard d'API donnée au théorème 5.1.

Un argument central pour prouver ce résultat, décrit dans le lemme suivant, montre que de manière à similaire à l'algorithme standard API, l'algorithme que nous venons de décire exhibe une propriété de monotonie (approchée).

Lemme 5.7. *A chaque itération, la valeur $v_{\pi_{k+1,\ell}}$ de la politique non-stationnaire*

$$\pi_{k+1,\ell} \;=\; \pi_{k+1}\ \pi_k\ \ldots\ \pi_{k+2-\ell}\ \pi_{k+1}\ \pi_k\ \ldots\ \pi_{k-\ell+2}\ldots$$

ne peut pas être beaucoup moindre que la valeur $v_{\pi'_{k,\ell}}$ de la politique non-stationnaire

$$\pi'_{k,\ell} \;=\; \pi_{k-\ell+1}\ \pi_k\ \ldots\ \pi_{k+2-\ell}\ \pi_{k-\ell+1}\ \pi_k\ \ldots\ \pi_{k-\ell+2}\ldots$$

dans le sens précis suivant :

$$v_{\pi_{k+1,\ell}} \geq v_{\pi'_{k,\ell}} - \frac{2\gamma}{1-\gamma^\ell}\epsilon.$$

La politique $\pi'_{k,\ell}$ diffère de $\pi_{k+1,\ell}$ par le fait que tous les ℓ pas, elle choisit la politique "vieille de ℓ pas" $\pi_{k-\ell+1}$ au lieu de "la plus récente" π_{k+1}. De plus, $\pi'_{k,\ell}$ est liée à $\pi_{k,\ell}$ de la manière suivante : $\pi'_{k,\ell}$ choisit sa première action selon $\pi_{k-\ell+1}$ puis exécute $\pi_{k,\ell}$; dit autrement, comme $\pi_{k,\ell}$ boucle indéfiniment sur $\pi_k\pi_{k-1}\ldots\pi_{k-\ell+1}$, $\pi'_{k,\ell} = \pi_{k-\ell+1}\pi_{k,\ell}$ peut être vue comme une "rotation" de 1 pas sur la droite de $\pi_{k,\ell}$. Quand il n'y a pas d'erreur ($\epsilon = 0$), le lemme ci-dessus implique que pour tout k et ℓ, $\pi_{k+\ell} \geq \pi_k$, autrement dit qu'on peut extraire une séquence de politiques non-stationnaires dont la valeur est croissante ; comme dans le cas standard, ceci implique la convergence en temps fini de l'algorithme dès lors que le nombre d'états et d'actions est fini. Enfin, on notera que lorsque $\ell = 1$, le lemme se réduit à un résultat connu sur API (voir par exemple (Bertsekas et Tsitsiklis, 1996, Lemme 6.1)).

Preuve du lemme 5.7. Comme $\pi'_{k,\ell}$ choisit la première action selon $\pi_{k-\ell+1}$ puis exécute $\pi_{k,\ell}$, on a $v_{\pi'_{k,\ell}} = T_{\pi_{k-\ell+1}} v_{\pi_{k,\ell}}$. Maintenant, puisque $\pi_{k+1} \in \mathcal{G}v_k$, on a $T_{\pi_{k+1}} v_k \geq T_{\pi_{k-\ell+1}} v_k$ et

$$\begin{aligned}
v_{\pi'_{k,\ell}} - v_{\pi_{k+1,\ell}} &= T_{\pi_{k-\ell+1}} v_{\pi_{k,\ell}} - v_{\pi_{k+1,\ell}} \\
&= T_{\pi_{k-\ell+1}} v_k - \gamma P_{\pi_{k-\ell+1}} \epsilon_k - v_{\pi_{k+1,\ell}} \\
&\leq T_{\pi_{k+1}} v_k - \gamma P_{\pi_{k-\ell+1}} \epsilon_k - v_{\pi_{k+1,\ell}} \\
&= T_{\pi_{k+1}} v_{\pi_{k,\ell}} + \gamma(P_{\pi_{k+1}} - P_{\pi_{k-\ell+1}})\epsilon_k - v_{\pi_{k+1,\ell}} \\
&= T_{\pi_{k+1}} T_{k,\ell} v_{\pi_{k,\ell}} - T_{k+1,\ell} v_{\pi_{k+1,\ell}} + \gamma(P_{\pi_{k+1}} - P_{\pi_{k-\ell+1}})\epsilon_k \\
&= T_{k+1,\ell} T_{\pi_{k-\ell+1}} v_{\pi_{k,\ell}} - T_{k+1,\ell} v_{\pi_{k+1,\ell}} + \gamma(P_{\pi_{k+1}} - P_{\pi_{k-\ell+1}})\epsilon_k
\end{aligned}$$

$$
\begin{aligned}
&= \Gamma_{k+1,\ell}(T_{\pi_{k-\ell+1}} v_{\pi_{k,\ell}} - v_{\pi_{k+1,\ell}}) + \gamma(P_{\pi_{k+1}} - P_{\pi_{k-\ell+1}})\epsilon_k \\
&= \Gamma_{k+1,\ell}(v_{\pi'_{k,\ell}} - v_{\pi_{k+1,\ell}}) + \gamma(P_{\pi_{k+1}} - P_{\pi_{k-\ell+1}})\epsilon_k,
\end{aligned}
$$

dont on déduit

$$
v_{\pi'_{k,\ell}} - v_{\pi_{k+1,\ell}} \leq (I - \Gamma_{k+1,\ell})^{-1}\gamma(P_{\pi_{k+1}} - P_{\pi_{k-\ell+1}})\epsilon_k.
$$

Le lemme résulte alors du fait que $\|\epsilon_k\|_\infty \leq \epsilon$ et que $\|(I - \Gamma_{k+1,\ell})^{-1}\|_\infty = \frac{1}{1-\gamma^\ell}$. □

Muni du lemme, nous pouvons passer à la preuve de la garantie.

Preuve du théorème 5.6. En utilisant le fait que 1) $T_{k+1,\ell+1} v_{\pi_{k,\ell}} = T_{\pi_{k+1}} T_{k,\ell} v_{\pi_{k,\ell}} = T_{\pi_{k+1}} v_{\pi_{k,\ell}}$ et que 2) $T_{\pi_{k+1}} v_k \geq T_{\pi_*} v_k$ (puisque $\pi_{k+1} \in \mathcal{G}v_k$), on a pour tout $k \geq 1$,

$$
\begin{aligned}
& v_* - v_{\pi_{k+1,\ell}} \\
&= T_{\pi_*} v_* - T_{k+1,\ell} v_{\pi_{k+1,\ell}} \\
&= T_{\pi_*} v_* - T_{\pi_*} v_{\pi_{k,\ell}} + T_{\pi_*} v_{\pi_{k,\ell}} - T_{k+1,\ell+1} v_{\pi_{k,\ell}} + T_{k+1,\ell+1} v_{\pi_{k,\ell}} - T_{k+1,\ell} v_{\pi_{k+1,\ell}} \\
&= \gamma P_{\pi_*}(v_* - v_{\pi_{k,\ell}}) + T_{\pi_*} v_{\pi_{k,\ell}} - T_{\pi_{k+1}} v_{\pi_{k,\ell}} + \Gamma_{k+1,\ell}(T_{\pi_{k-\ell+1}} v_{\pi_{k,\ell}} - v_{\pi_{k+1,\ell}}) \\
&\leq \gamma P_{\pi_*}(v_* - v_{\pi_{k,\ell}}) + T_{\pi_*} v_k - T_{\pi_{k+1}} v_k + \gamma(P_{\pi_{k+1}} - P_{\pi_*})\epsilon_k + \Gamma_{k+1,\ell}(T_{\pi_{k-\ell+1}} v_{\pi_{k,\ell}} - v_{\pi_{k+1,\ell}}) \\
&\leq \gamma P_{\pi_*}(v_* - v_{\pi_{k,\ell}}) + \gamma(P_{\pi_{k+1}} - P_{\pi_*})\epsilon_k + \Gamma_{k+1,\ell}(T_{\pi_{k-\ell+1}} v_{\pi_{k,\ell}} - v_{\pi_{k+1,\ell}}). \qquad (5.8)
\end{aligned}
$$

Considérons la politique $\pi'_{k,\ell}$ définie au lemme 5.7. Remarquant, comme au début de la preuve du lemme 5.7 que $T_{\pi_{k-\ell+1}} v_{\pi_{k,\ell}} = v_{\pi'_{k,\ell}}$, l'équation (6.7) peut être récrite comme suit :

$$
v_* - v_{\pi_{k+1,\ell}} \leq \gamma P_{\pi_*}(v_* - v_{\pi_{k,\ell}}) + \gamma(P_{\pi_{k+1}} - P_{\pi_*})\epsilon_k + \Gamma_{k+1,\ell}(v_{\pi'_{k,\ell}} - v_{\pi_{k+1,\ell}}).
$$

En utilisant le fait que $v_* \geq v_{\pi_{k,\ell}}$, $v_* \geq v_{\pi_{k+1,\ell}}$ et le lemme 5.7, on obtient

$$
\begin{aligned}
\|v_* - v_{\pi_{k+1,\ell}}\|_\infty &\leq \gamma\|v_* - v_{\pi_{k,\ell}}\|_\infty + 2\gamma\epsilon + \frac{\gamma^\ell(2\gamma\epsilon)}{1-\gamma^\ell} \\
&= \gamma\|v_* - v_{\pi_{k,\ell}}\|_\infty + \frac{2\gamma}{1-\gamma^\ell}\epsilon.
\end{aligned}
$$

Finalement, on en déduit par une récurrence sur k que pour tout $k \geq 1$,

$$
\|v_* - v_{\pi_{k,\ell}}\|_\infty \leq \gamma^k\|v_* - v_{\pi_{0,\ell}}\|_\infty + \frac{2(\gamma - \gamma^{k+1})}{(1-\gamma)(1-\gamma^\ell)}\epsilon. \quad \square
$$

NSMPI(m, ℓ) : une variation d'AMPI pour construire des politiques non-stationnaires avec période fixe. Dans le chapitre précédent, nous avons décrit en détails une famille d'algorithmes, AMPI, qui permet d'interpoler entre les deux approches standard AVI et API. Nous venons de décrire des versions non-stationnaires d'algorithmes de type AVI et API, et une question naturelle est de savoir s'il est possible de les interpoler de façon analogue. C'est ce que nous décrivons maintenant. Etant donnés deux paramètres entiers $m \geq 0$ et $\ell \geq 1$, une fonction valeur v_0 et un ensemble de $\ell-1$ politiques stationnaires $\pi_0, \pi_{-1}, \pi_{-\ell+2}$, on considère l'algorithme NSMPI(m, ℓ) qui génère une séquence de couples de fonction valeurs et de politiques comme suit :

$$
\begin{aligned}
&\pi_{k+1} \leftarrow \text{un élément de } \mathcal{G}v_k \\
&v_{k+1} \leftarrow (T_{\pi_{k+1,\ell}})^m T_{\pi_{k+1}} v_k + \epsilon_k.
\end{aligned}
$$

L'étape gloutonne (de mise à jour de la politique) est identique à celle que nous avons décrite dans l'ensemble des algorithmes de ce manuscrit. L'étape d'évaluation (de mise à jour de la fonction valeur) implique l'opérateur de Bellman non-stationnaire $T_{\pi_{k+1,\ell}}$ (composé avec lui-même m fois) introduit au début de ce chapitre et l'opérateur de Bellman standard $T_{\pi_{k+1}}$. De manière similaire aux algorithmes que nous avons décrits dans les paragraphes précédents, après k itérations, l'algorithme retourne la politique non-stationnaire $\pi_{k,\ell}$ qui boucle sur les ℓ dernières politiques générées.

Pour les valeurs $m = 0$ and $m = \infty$, il est facile de voir qu'on retombe respectivement sur NSVI et NSPI(ℓ). Lorsque $\ell = 1$, on retombe sur AMPI que nous avons décrit au chapitre précédent (et on retrouve respectivement les versions standards d'AVI et d'API pour $m = 0$ et $m = \infty$). Autrement dit, ce dernier schéma algorithmique a le bon goût de généraliser la plupart des algorithmes de programmation dynamique que nous avons décrits jusqu'ici.

A ce stade, la question naturelle suivante est de savoir si cette généralisation hérite des garanties de performances énoncées plus haut. Comme le montre le théorème suivant, la réponse est affirmative.

Théorème 5.8. *Pour tout couple de paramètres $m \geq 0$ et $\ell \geq 1$, après k itérations de NSMPI(m, ℓ), la perte liée au fait de suivre la politique non-stationnaire $\pi_{k,\ell}$ au lieu de la politique optimale π_* satisfait :*

$$\|v_* - v_{\pi_{k,\ell}}\|_\infty \leq \frac{2(\gamma - \gamma^k)}{(1-\gamma)(1-\gamma^\ell)}\epsilon + \frac{2\gamma^k}{1-\gamma}\|v_* - v_0\|_\infty.$$

Ce théorème généralise (asymptotiquement) les résultats des théorèmes 5.3 et 5.6 (les garanties sont très proches dans les cas limites $m = 0$ et $m = \infty$), ainsi que le résultat en norme infinie d'AMPI présenté dans le chapitre précédent, à l'équation (4.21) page 57. Comme nous l'avions déjà observé pour AMPI ($\ell = 1$), il est remarquable que le résultat ci-dessus (ℓ quelconque) ne dépende pas du paramètre d'interpolation m.

La preuve est une généralisation de celle esquissée au chapitre précédent. De manière analogue à ce que nous avions fait, on définit pour tout k les quantité suivantes :

1. la distance entre la fonction valeur optimale et la valeur avant approximation à l'itération k :

$$d_k = v_* - (T_{\pi_k})^m v_{k-1} = v_* - (v_k - \epsilon_k);$$

2. le décalage (shift en anglais) entre la valeur avant approximation et la (vraie) valeur de la politique à l'itération k :

$$s_k = (v_k - \epsilon_k) - v_{\pi_{k,\ell}};$$

3. le résidu de Bellman de la valeur à l'itération k :

$$b_k = T_{\pi_{k+1}} v_k - T_{k+1,\ell} T_{\pi_{k+1}} v_k;$$

Notre objectif est d'obtenir un majorant de la *perte* (loss en anglais) :

$$l_k = v_{\pi_*} - v_{\pi_k} = v_* - v_{\pi_k} = d_k + s_k.$$

Le cœur de la preuve, détaillée dans (Lesner et Scherrer, 2015), consiste à dériver les relations de récurrence suivantes, qui généralisent celles du lemme (4.2) page 55 au cas $\ell > 1$.

Lemme 5.9. *Pour tout $k \geq 1$, on a les inégalités vectorielles suivantes :*

$$b_k \leq \Gamma_{k+1}\left((\Gamma_{k,\ell})^m b_{k-1} + (I - \Gamma_{k,\ell})\epsilon_k\right),$$
$$d_k = \gamma P_{\pi_*} d_{k-1} - \gamma P_{\pi_*}\epsilon_{k-1} + \sum_{i=0}^{m-1}(\Gamma_{k,\ell})^i b_{k-1},$$
$$s_k = (\Gamma_{k,\ell})^m \sum_{j=0}^{\infty}(\Gamma_{k,\ell})^j b_{k-1}.$$

L'exploitation de ces inégalités pour obtenir le résultat du théorème est alors analogue (bien que légèrement plus compliquée) à ce qui est fait pour AMPI.

5.3 Optimalité de la constante $\frac{2\gamma}{(1-\gamma)(1-\gamma^\ell)}$ pour les politiques non-stationnaires ℓ-périodiques

Le théorème 5.8 est le résultat le plus général concernant les garanties de performance si on utilise un algorithme de programmation dynamique pour calculer une politique stationnaire ($\ell = 1$) ou non ($\ell > 1$), avec un degré d'interpolation quelconque entre AVI ($m = 0$) et API ($m = \infty$). La (dernière) question naturelle, à laquelle nous avons répondu affirmativement avec Boris Lesner, est celle de l'optimalité de cette borne.

Théorème 5.10. *Il existe une instance de MDP, une fonction valeur initiale v_0, un ensemble initial de politiques $\pi_0, \pi_{-1}, \ldots, \pi_{-\ell+2}$ et une séquence de termes d'erreur $(\epsilon_k)_{k\geq 1}$ satisfaisant $\|\epsilon_k\|_\infty \leq \epsilon$, tels que pour tout nombre d'itérations k, la borne du théorème 5.8 est satisfaite avec égalité.*

Ce théorème étend les résultats décrits dans la première section de ce chapitre, qui rappelons-le portaient sur API ($m = \infty$, $\ell = 1$) et AVI ($m = 0$, $\ell = 1$). A notre connaissance, ce résultat était nouveau même dans le cas d'AMPI ($\ell = 1$ et m arbitraire).

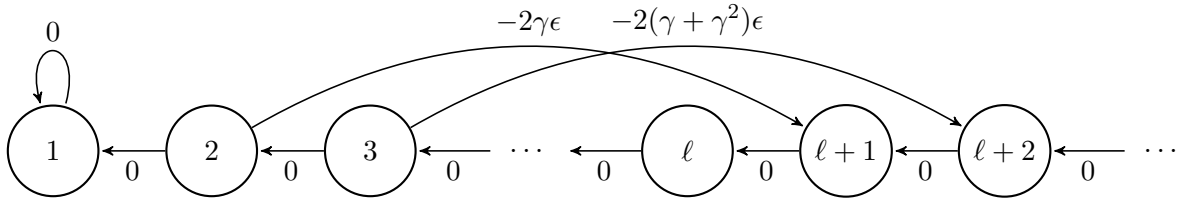


FIGURE 5.2 – Le MDP déterministe pour lequel la borne du théorème 5.8 est optimale pour tout m et ℓ.

La preuve de ce résultat repose sur la famille (indexée par ℓ) de MDPs déterministes illustrés à la figure 5.2 et décrits ci-après, et qui généralise au cas $\ell > 1$ l'instance décrite à la figure 5.1 page 69.

Exemple 5.11. *On considère un MDP avec un nombre infini d'états $1, 2, \ldots$. L'état 1 est absorbant et a une récompense nulle. Pour tout état $i > 1$, il y a deux actions possibles : gauche ($\leftarrow$) et droite ($\rightarrow$) ; la fonction récompense et les transitions sont caractérisées comme suit :*

$$r(i, \leftarrow) = 0, \qquad r(i, \rightarrow) = -2\frac{\gamma - \gamma^i}{1-\gamma}\epsilon,$$

$$P(i|i+1,\leftarrow)=1, \qquad\qquad P(i+\ell-1|i,\rightarrow)=1,$$

ainsi que $r(1)=0$ *et* $P(1|1)=1$ *pour l'état* 1. *Par souci de clarté, on utilisera la notation* r_i *pour la récompense non nulle* $r(i,\rightarrow)$ *obtenue quand on choisit l'action* $\rightarrow$ *en l'état* i.

Comme les récompenses non nulles sont strictement négatives, il est clair que l'unique politique optimale consiste à toujours choisir l'action $\leftarrow$ *qui permet de passer de l'état* i *à l'état* $i-1$ *jusqu'à l'état absorbant* 1, *et ainsi que la valeur optimale* v_* *vaut* 0 *en tout état.*

On considère ici qu'on initialise NSMPI(m,ℓ) avec $v_0 = v_* = 0$. Nous supposons que la séquence d'erreurs (ϵ_k) est la même que celle décrite dans le précédent exemple. Nous la reprécisons par souci de clarté : pour tout $k>0$,

$$\epsilon_k(i)=\begin{cases}-\epsilon & \text{si } i=k,\\ \epsilon & \text{si } i=k+1,\\ 0 & \text{sinon.}\end{cases}$$

Nous ne donnerons cette fois-ci pas de preuve détaillée car elle est fastidieuse. Nous renvoyons le lecteur à (Lesner et Scherrer, 2015) pour plus de détails. En bref, pour toute valeur des paramètres m et ℓ, pour tout itération k et tout état i, nous avons dérivé une expression exacte de la valeur $v_k(i)$. Ceci permet de prouver que la séquence des politiques $\pi_1,\pi_2,\ldots,\pi_k$ générées par l'algorithme jusqu'à l'itération k est telle que pour tout $i\leq k$, la politique π_i choisit l'action optimale $\leftarrow$ dans tous les états sauf l'état i, dans lequel elle choisit $\rightarrow$. Ainsi, l'exécution de la politique non-stationnaire $\pi_{k,\ell}$ à partir de l'état k induit la dynamique suivante : à cause de π_k on choisit l'action $\rightarrow$, ce qui amène le système en l'état $k+\ell-1$ avec une récompense r_k ; ensuite, les politiques $\pi_{k-1},\pi_{k-2},\ldots,\pi_{k-\ell+1}$ font choisir l'action $\leftarrow$ $\ell-1$ fois consécutives avec une récompense 0. Ainsi, après ℓ pas, le système se retrouve de nouveau dans l'état k, et par la périodicité de la politique, on rechoisit l'action $\rightarrow$ et $\ell-1$ fois l'action $\leftarrow$ indéfiniment. Le système est donc bloqué dans une boucle de période ℓ, dans laquelle une récompense r_k est reçue toutes les ℓ pas. Par conséquent, la valeur de cette politique non-stationnaire en l'état k est :

$$v_{\pi_{k,\ell}}(k)=\sum_{t=0}^{\infty}\gamma^{\ell t}r_k=\frac{r_k}{1-\gamma^\ell}=-\frac{2(\gamma-\gamma^k)}{(1-\gamma)(1-\gamma^\ell)}\epsilon.$$

On a alors la relation suivante

$$\|v_*-v_{\pi_{k,\ell}}\|_\infty \geq |v_{\pi_{k,\ell}}(k)|=\frac{2(\gamma-\gamma^k)}{(1-\gamma)(1-\gamma^\ell)}\epsilon$$

qui correspond *exactement* à la borne du théorème 5.8 (dans la mesure où $v_0=v_*=0$).

5.4 Bilan et perspectives

Nous avons rappelé au début de ce chapitre les garanties de performance standard lorsqu'on calcule une politique stationnaire à l'aide d'une version approchée d'AVI et d'API (théorème 5.1). Après avoir argumenté que cette borne était optimale, nous avons décrit quatre algorithmes—un de type AVI, deux de type API, et un dernier de type AMPI (qui interpole entre AVI et API)—qui construisent des politiques non-stationnaires aves des garanties de performance améliorées (l'amélioration pouvant atteindre un facteur multiplicatif $\frac{1}{1-\gamma}$). Les meilleures performances asymptotiques sont obtenues lorsque ces algorithmes considèrent des politiques non-stationnaires dont la période s'approche de l'infini, ce qui peut constituer une limitation car la

mémoire requise est alors infinie. Cependant, pour trois des quatre algorithmes, un paramètre (ℓ) permet de faire un compromis—usuel en informatique—entre la qualité d'approximation $\frac{2\gamma}{(1-\gamma^\ell)(1-\gamma)}\epsilon$ et la quantité de mémoire $O(\ell)$ requise. En pratique, il est facile de voir qu'il suffit de choisir $\ell = \left\lceil \frac{1}{1-\gamma} \right\rceil$, c'est-à-dire une mémoire linéaire en l'*horizon effectif du problème* (et donc aussi en sa difficulté) pour obtenir une garantie de performance de [2] $\frac{2\gamma}{(1-e^{-1})(1-\gamma)}\epsilon \leq \frac{3.164\gamma}{1-\gamma}\epsilon$.

Comme il est connu que parmi les politiques optimales, il en existe au moins une qui est stationnaire, les résultats que nous avons présentés dans ce chapitre consistant à considérer des politiques non-stationnaires peuvent à première vue paraître surprenants. Il existe, cependant, un schéma d'approximation relativement simple pour les problèmes à horizon infini actualisés—qui n'a à notre connaissance pas été documenté dans la littérature—où l'apparition de politiques non-stationnaires peut sembler beaucoup plus naturelle, et donner un éclairage sur nos contributions. Etant donné un problème à horizon infini, considérons l'approximation qui consiste à tout d'abord le transformer en un problème à horizon fini en "stoppant" le système après un instant T (de manière équivalente, cela revient à supposer que les récompenses sont nulles après T). Contrairement au problème originel (à horizon infini), cette approximation à horizon fini n'est plus un problème stationnaire, et sa solution doit être cherchée dans l'espace des politiques non-stationnaires à horizon T, et est naturellement calculée de manière rétrograde par programmation dynamique (approchée). On peut alors montrer (Kakade, 2002, en adaptant l'analyse de sensibilité pour les problèmes à horizon fini du théorème 2.5.1) qu'un schéma AVI avec une erreur ϵ à chaque étape jouira d'une performance de garantie de $2\sum_{i=0}^{T-1}\gamma^t\epsilon = \frac{2(1-\gamma^T)}{1-\gamma}\epsilon$. Si on ajoute l'erreur due au fait de tronquer l'horizon après l'instant T ($\gamma^T\frac{R_{\max}}{1-\gamma}$), on obtient une erreur globale en $O\left(\frac{1}{1-\gamma}\epsilon\right)$ dès que T est de l'ordre $\tilde{O}\left(\frac{1}{1-\gamma}\right)$. Bien que ce schéma d'approximation puisse requérir en général une mémoire importante (notamment quand γ est proche de 1), il jouit de garanties aussi bonnes que les meilleures que nous avons décrites dans ce chapitre. Vis-à-vis de ce schéma, les algorithmes que nous avons détaillés ont pour intérêt principal de permettre un contrôle explicite de la mémoire.

Une autre interprétation, un tout petit peu plus fine, est la suivante. On peut se rendre compte que NSVI, NSPI(ℓ) et leur généralisation NSMPI(m, ℓ) sont des algorithmes naturels pour résoudre des MDPs ℓ-périodiques. Nos résultats montrent que la sensibilité au bruit est d'autant plus faible que la période ℓ est grande. Ainsi, quand nous proposons d'appliquer ces algorithmes à un problème stationnaire, il s'agit de considérer ce problème comme un problème ℓ-périodique (ceci se fait sans aucune approximation), et de lui appliquer une méthode adaptée.

Parmi toutes les contributions exposées dans ce manuscrit, celles dans ce chapitre sont celles qui sont probablement les plus abouties : non seulement avons-nous généralisé les garanties de la sensibilité aux erreurs au schéma algorithmique très général NSMPI(ℓ, m), mais des instances de MDPs nous ont permis de montrer que ces garanties ne peuvent pas être améliorées dans le cas général. Autrement dit, nous avons *in fine* une caractérisation fine et complète de la sensibilité au bruit de cette grande famille d'algorithmes de programmation dynamique.

Plusieurs perspectives naturelles sont néanmoins à considérer. Tout d'abord, l'analyse décrite dans ce chapitre s'est limitée à l'hypothèse simplificatrice d'un contrôle de l'erreur en norme infinie. Adapter notre analyse à l'hypothèse plus raisonnable d'une erreur contrôlée par exemple en norme ℓ_2 à poids est immédiat via le lemme technique 4.6 page 4.6. Dans le prochain chapitre, nous développerons un peu plus précisément l'analyse de l'algorithme NSMPI(ℓ, m) sous l'hypothèse d'un contrôle de l'erreur en norme l_1 à poids.

D'un point de vue un peu plus concret, une piste intéressante serait d'instancier des implémentations précises de NSMPI(ℓ, m) pour étudier l'intérêt pratique de l'utilisation de politiques

2. Avec ce choix de ℓ, on a $\ell \geq \frac{1}{\log 1/\gamma}$ et ainsi $\frac{2}{1-\gamma^\ell} \leq \frac{2}{1-e^{-1}} \leq 3.164$.

non-stationnaires. Si un certain nombre d'expériences "jouet" dans (Lesner et Scherrer, 2015; Tagorti, 2015) et dans le chapitre suivant montrent une amélioration empirique conséquente, une étude un peu plus approfondie sur des problèmes de plus grande taille reste à faire.

Finalement, comme perspective un peu plus ouverte (et donc aussi intéressante), le présent chapitre aura montré comment la formulation du *problème à horizon fini* pouvait venir au secours de celle *à horizon infini*. Dans l'optique d'approfondir les propriétés particulières et respectives de ces deux formulations, une question qui me semble intéressante est de voir dans quelle mesure un certain nombre de résultats obtenus à horizon infini pourraient suggérer de nouvelles idées pour les problèmes à horizon fini. Par exemple, lorsque l'horizon est grand (mais fini), vouloir garder le contrôle de la taille mémoire du contrôleur peut être utile en pratique ; comprendre les enjeux précis d'approcher une politique de long horizon par une politique stationnaire (ou périodique) est une direction qui me semble intéressante.

Chapitre 6

Sur quelques schémas d'approximation de type Itérations sur les Politiques

Dans le chapitre 4 nous avons dérivé des garanties d'erreur détaillées pour des algorithmes de programmation dynamique approchée. D'une manière générale, pour un algorithme commettant une erreur ϵ contrôlée en norme ℓ_p à poids, nous avons montré que nous pouvons obtenir des garanties de la forme :

$$\|v_* - v_{\pi_k}\| = \frac{2\gamma C}{(1-\gamma)^2} + O(\gamma^k),$$

où C est une constante dite de concentrabilité qui dépend d'un certain nombre de propriétés de mélange du MDP. Dans le chapitre 5, nous avons montré comment améliorer la dépendance de cette borne en fonction du facteur d'actualisation γ en utlisant des politiques non-stationnaires. Dans ce chapitre, nous poursuivons ce "travail sur la constante $\frac{2\gamma C}{(1-\gamma)^2}$" dans des garanties comme celles ci-dessus, en décrivant une étude comparative d'un certain nombre de schémas algorithmiques, et en portant cette fois-ci une attention particulière à ce qui est caché dans la constante de concentrabilité C. Plus précisément, nous considérons plusieurs schémas approximatifs de type PI. Ces schémas peuvent être vus comme implémentant une approximation de l'opérateur gourmand, $\mathcal{G}_\epsilon$, qui prend en paramètres une distribution ν et une fonction $v : S \to \mathcal{R}$, et renvoie une politique π qui est (ϵ, ν)-approximativement gourmande par rapport à v dans le sens suivant :

$$\nu(Tv - T_\pi v) = \nu(\max_{\pi'} T_{\pi'} v - T_\pi v) \leq \epsilon. \tag{6.1}$$

où pour tout x, νx est égal à $\mathbb{E}_{s\sim\nu}[x(s)]$. En pratique, cette approximation peut être effectuée via une régression ℓ_2 de la fonction de valeur états/actions Q—une régression directe est suggérée pour *Conservative Policy Iteration* (CPI) dans (Kakade et Langford, 2002; Kakade, 2003), une approche LSTD(0) de type point fixe projeté est utilisée pour *Least Squares Policy Iteration* (LSPI) (Lagoudakis et Parr, 2003b)—où via un problème de classification (à coût sensitif) (Lagoudakis et Parr, 2003a; Lazaric *et al.*, 2010)—c'est notamment le cas pour l'algorithme DPI (CBMPI avec $m = \infty$) décrit au chapitre 4. Avec cet opérateur, nous allons décrire plusieurs schémas de type PI dans la section 6.1. Ensuite, la section 6.2 présente une analyse détaillée et comparée de leur garanties de performance, de leur complexité en temps et en mémoire. La section 6.3 présente ensuite des simulations numériques qui illustrent leur comportement et confirme notre analyse. Ce travail a donné lieu à un article (Scherrer, 2014), dans lequel le lecteur intéressé pourra trouver des compléments (preuves et simulations numériques non reproduites ici).

6.1 Algorithmes

Itérations sur les politiques approché (API). Nous commençons par décrire le schéma le plus standard, API (Bertsekas et Tsitsiklis, 1996). A chaque itération k, l'algorithme passe à la politique qui est approximativement gourmande par rapport à la valeur v_{π_k} de la politique courante π_k pour une distribution ν donnée :

$$\pi_{k+1} \leftarrow \mathcal{G}_{\epsilon_{k+1}}(\nu, v_{\pi_k}). \tag{6.2}$$

S'il n'y a pas d'erreur ($\epsilon_k = 0$) et si ν met un poids positif en chaque état, il est facile de voir que cet algorithme génère la même séquence de politiques que la version exacte de PI, car d'après l'équation (6.1), les politiques sont exactement gourmandes.

Itérations sur les politiques conservatif (CPI/CPI(α)/API(α)). Nous passons maintenant à la description d'itérations sur les politiques conservatif (CPI) proposé par Kakade et Langford (2002). A l'itération k, CPI, décrit par l'équation (6.3), utilise la distribution $d_{\pi_k,\nu} = (1-\gamma)\nu(I - \gamma P_{\pi_k})^{-1}$—la mesure d'occupation cumulée actualisée induite par π_k en partant de ν—pour appeler l'opérateur gourmand approché, et un pas d'apprentissage α_k pour générer un mélange stochastique de toutes les politiques qui ont été retournées par l'opérateur gourmand approché, ce qui explique l'adjectif "conservatif" :

$$\pi_{k+1} \leftarrow (1-\alpha_{k+1})\pi_k + \alpha_{k+1}\mathcal{G}_{\epsilon_{k+1}}(d_{\pi_k,\nu}, v_{\pi_k}). \tag{6.3}$$

Le pas d'apprentissage α_{k+1} peut être choisi de sorte que chaque itération induise une amélioration de l'espérance de la valeur de la politique sachant que le processus est initialisé selon ν (Kakade et Langford, 2002). L'article original décrit également un critère d'arrêt. Si l'utilisation d'un pas d'apprentissage adaptatif et de cette condition d'arrêt est intéressante pour dériver une analyse élégante de performance (voir (Kakade et Langford, 2002) et la prochaine section), elle est en général très conservatrice. En pratique, le pas d'apprentissage α_k doit être choisi par un mécanisme de recherche linéaire, ou fixé à une petite valeur α. Nous ferons référence à ce dernier cas de CPI en écrivant CPI(α).

Il est naturel de considérer également l'algorithme API(α) (évoqué par Lagoudakis et Parr (2003a)), qui est une variation d'API qui est conservative comme CPI(α) dans le sens où elle mélange la nouvelle politique avec les précédentes avec les poids α et $1-\alpha$, mais qui utilise directement la distribution ν pour l'appel à l'opérateur approximativement gourmand :

$$\pi_{k+1} \leftarrow (1-\alpha)\pi_k + \alpha\mathcal{G}_{\epsilon_{k+1}}(\nu, v_{\pi_k}) \tag{6.4}$$

Parce qu'il utilise ν au lieu de $d_{\pi_k,\nu}$, API(α) est plus simple à implémenter que CPI(α) [1].

"Policy Search by Dynamic Programming" pour des problèmes à horizon infini (PSDP$_\infty$). Nous allons à présent décrire un algorithme d'un type similaire à API—dans le sens où à chaque étape, il passe à une nouvelle politique déterministe—mais qui est conservatif comme CPI—dans le sens où les politiques considérées évoluent de plus en plus doucement. Cet algorithme est une variation naturelle de l'algorithme "Policy Search by Dynamic Programming"

1. En pratique, contrôler l'étape approximativement gourmande par rapport à $d_{\pi_k,\nu}$ requiert de générer des échantillons selon cette distribution. Comme expliqué par Kakade et Langford (2002), un échantillon pour cette distribution peut être obtenu en simulant une trajectoire partant d'une distribution ν et suivant la politique π_k, et en s'arrêtant à chaque étape avec une probabilité $1-\gamma$. En particulier, la génération d'un échantillon pour $d_{\pi_k,\nu}$ requiert en moyenne $\frac{1}{1-\gamma}$ échantillons du MDP sous-jacent. De ce point de vue, API(α) est beaucoup plus léger.

(PSDP) de Bagnell *et al.* (2003)—originellement proposé dans le cadre des problèmes à horizion fini—au cas de l'horion infini ; nous y ferons donc référence via l'appellation PSDP_∞. A notre connaissance cependant, cette variation (pour un problème à horizon infini) n'avait jamais été décrite dans la littérature.

L'algorithme est fondé sur l'utilisation de politiques non-stationnaires à horizon fini. Etant donnée une séquence de politiques déterministes stationnaires (π_k) que l'algorithme va progressivement générer, nous noterons $\sigma_k = \pi_k \pi_{k-1} \dots \pi_1$ la politique non-stationnaire à horizon k qui effectue la première action selon π_k, la deuxième selon π_{k-1}, etc. Sa valeur est $v_{\sigma_k} = T_{\pi_k} T_{\pi_{k-1}} \dots T_{\pi_1} r$. Nous noterons $\varnothing$ la politique non-stationnaire "vide". Remarquons que $v_\varnothing = r$ et que n'importe quelle politique à horizon infini qui commence par $\sigma_k = \pi_k \pi_{k-1} \dots \pi_1$, ce que nous noterons (abusivement) "$\sigma_k \dots$" a une valeur $v_{\sigma_k \dots} \geq v_\sigma - \gamma^k V_{\max}$. En partant de $\sigma_0 = \varnothing$, l'algorithme construit implicitement une séquence de politiques non-stationnaires (σ_k) en concaténant de manière itérative les politiques retournées par l'opérateur approximativement gourmand :

$$\pi_{k+1} \leftarrow \mathcal{G}_{\epsilon_{k+1}}(\nu, v_{\sigma_k}). \tag{6.5}$$

Tandis que l'algorithme originel PSDP de Bagnell *et al.* (2003) considère un horizon fini T et effectue T itérations, l'algorithme que nous considérons ici a un nombre *indéfini* d'itérations. L'algorithme peut être arrêté à n'importe quelle itération k. L'analyse que nous allons bientôt décrire suggère de retourner n'importe quelle politique qui commence par la politique non-stationnaire σ_k. Vu que σ_k est une politique à horizon fini qui est approximativement optimale (pour l'horizon fini), et que nous considérons le cas d'un problème à horizon infini, il est naturel de considérer en pratique comme résultat la politique qui boucle indéfiniment sur σ_k, ce que nous noterons $(\sigma_k)^\infty$ (dans le chapitre précédent, nous notions cet objet $\pi_{k,k}$).

D'un point de vue pratique, PSDP_∞ et CPI requièrent de stocker toutes les politiques (déterministes stationnaires) générées depuis le début. La complexité en mémoire est donc proportionnelle au nombre d'itérations, ce qui peut s'avérer problématique. Le but du prochain paragraphe, qui présente le dernier algorithme étudié dans cet article, est de proposer une solution à ce problème potentiel de mémoire.

Itérations sur les politiques non-stationnaires (NSPI(ℓ)). Nous avons originellement conçu le schéma algorithmique de l'équation (6.5) (PSDP_∞) comme une simplification de NSPI($\mathbb{N}$) présenté au précédent chapitre et introduit dans Scherrer et Lesner (2012) [2]. Par rapport à l'équation (6.5), la seule différence de NSPI($\mathbb{N}$) réside dans le fait que l'étape approximativement gourmande est faite par rapport à la valeur $v_{(\sigma_k)^\infty}$ de la politique qui répète indéfiniment σ_k (formellement l'algorithme fait $\pi_{k+1} \leftarrow \mathcal{G}_{\epsilon_{k+1}}(\nu, v_{(\sigma_k)^\infty})$) au lieu de la valeur v_{σ_k} des k premiers pas. Suivant l'intuition que lorsque k est grand, ces deux valeurs doivent être proches, nous avons considéré l'algorithme PSDP_∞ car il est plus simple.

NSPI($\mathbb{N}$) souffre du même problème potentiel de mémoire que CPI et PSDP_∞. De manière intéressante, le chapitre qui précède (initialement, Scherrer et Lesner (2012)) décrit un autre algorithme, NSPI(ℓ), qui prend en entrée un paramètre qui contrôle directement le nombre de politiques stockées en mémoire.

De manière similaire à PSDP_∞, NSPI(ℓ) est fondé sur l'utilisation de politiques non-stationnaires. Il prend en entrée un paramètre ℓ. Il requiert un ensemble de ℓ politiques déterministes stationnaires $\pi_{\ell-1}, \pi_{\ell-2}, \dots, \pi_0$ et génère itérativement de nouvelles politiques $\pi_1, \pi_2, \dots$. Pour tout $k \geq 0$, nous noterons σ_k^ℓ la politique non-stationnaire à horizon fini ℓ qui exécute *dans l'ordre inverse* les ℓ dernières politiques, ce que nous écrirons formellement : $\sigma_k^\ell = \pi_k \ \pi_{k-1} \ \dots \ \pi_{k-\ell+1}$.

2. Nous avons pris conscience ultérieurement qu'il s'agissait d'une variation naturelle de PSDP. Pour "rendre à César ce qui est à César", nous avons gardé PSDP comme référence principale et donné le nom PSDP_∞.

De plus, nous noterons $(\sigma_k^\ell)^\infty$ la politique à horizon fini, non-stationnaire avec période ℓ, qui boucle indéfiniment sur σ_k^ℓ (dans le chapitre précédent, nous avons utilisé la notation $pi_{k,\ell}$). Partant de $\sigma_0^\ell = \pi_0\pi_1 \dots \pi_{\ell-1}$, l'algorithme itère de la façon suivante :

$$\pi_{k+1} \leftarrow \mathcal{G}_{\epsilon_{k+1}}(\nu, v_{(\sigma_k^\ell)^\infty}). \tag{6.6}$$

Chaque itération requiert de calculer une politique π_{k+1} qui est approximativement gourmande par rapport à la valeur $v_{(\sigma_k^\ell)^\infty}$ de $(\sigma_k^\ell)^\infty$, qui est le point fixe de l'opérateur composé [3] :

$$\forall v,\ T_{k,\ell}v = T_{\pi_k}T_{\pi_{k-1}} \dots T_{\pi_{k-\ell+1}}v.$$

Lorsqu'on passe de l'itération k à l'itération $k+1$, le processus consiste à ajouter la politique π_{k+1} au début de la politique $\pi_k\pi_{k-1} \dots \pi_{k-\ell+2}$ à horizon $\ell - 1$, formant ainsi une nouvelle politique σ_{k+1}^ℓ à horizon ℓ. Ce faisant, l'algorithme oublie la politique la plus ancienne $\pi_{k-\ell+1}$ de σ_k^ℓ et garde une mémoire constante de taille ℓ. A n'importe quelle itération, l'algorithme peut être stoppé, et on retourne la politique $\pi_{k,\ell} = (\sigma_k^\ell)^\infty$ à horizon infini qui répète indéfiniment σ_k^ℓ. Comme nous l'avons déja mentionné au chapitre précédent, il est facile de voir que NSPI(ℓ) est identique à API lorsque $\ell = 1$. De manière plus intéressante, si on ajoute des actions spéciales "stop" en chaque état qui mène dans un état terminal avec une récompense strictement plus petite que $-R_{\max}$, et si on initialise avec une séquence infinie de politique qui ne prennent que cette action "stop", alors NSPI(ℓ) avec $\ell = \infty$ est équivalent à PSDP_∞.

6.2 Analyse

Pour tous les algorithmes considérés, nous allons décrire des bornes sur la perte

$$E_{s\sim\mu}[v_{\pi_*}(s) - v_\pi(s)] = \mu(v_{\pi_*} - v_\pi)$$

liée au fait d'exécuter la politique π (potentiellement stochastique ou non-stationnaire) renvoyée par les algorithmes au lieu de la politique optimale π_* à partir d'une distribution μ, en fonction d'une borne uniforme ϵ sur les erreurs (ϵ_k). Afin de dériver ces garanties, nous aurons besoin d'introduire plusieurs *constantes de concentrabilité*, qui mesurent l'adéquation de la distribution μ avec laquelle on mesure la perte, et la distribution ν utilisée par les algorithmes.

Définition 6.1. *Soient $c(1), c(2), \dots$ les plus petits coefficients dans $[1,\infty) \cup \{\infty\}$ tels que pour tout i et tout ensemble de politiques déterministes stationnaires $\pi_1, \pi_2, \dots, \pi_i$, $\mu P_{\pi_1} P_{\pi_2} \dots P_{\pi_i} \leq c(i)\nu$. Pour tout ℓ, k, Nous définissons les constantes suivantes de $[1,\infty) \cup \{\infty\}$:*

$$C^{(1,k)} = (1-\gamma)\sum_{i=0}^{\infty} \gamma^i c(i+k),$$

$$C^{(2,\ell,k)} = (1-\gamma)(1-\gamma^\ell)\sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \gamma^{i+j\ell} c(i+j\ell+k).$$

De manière similaire, soient $c_{\pi_*}(1), c_{\pi_*}(2), \dots$ les plus petits coefficients dans $[1,\infty) \cup \{\infty\}$ tels que pour tout i, $\mu(P_{\pi_*})^i \leq c_{\pi_*}(i)\nu$. Nous définissons la constante de $[1,\infty) \cup \{\infty\}$:

$$C_{\pi_*}^{(1)} = (1-\gamma)\sum_{i=0}^{\infty} \gamma^i c_{\pi_*}(i).$$

3. En pratique, implémenter cet algorithme peut trivialement être fait via la classification à coût sensitif d'une manière similaire à celle suggérée dans Lazaric *et al.* (2010). Cela peut également être fait via une extension immédiate de LSTD au cas non-stationnaire.

Algorithme	Borne de l'erreur			# Iter.	Mémoire	Référence
API (équation (6.2)) (= NSPI(1))	$C^{(2,1,0)}$	$\frac{1}{(1-\gamma)^2}$	ϵ	$\frac{1}{1-\gamma}\log\frac{1}{\epsilon}$	1	(Lazaric *et al.*, 2010)
	$C^{(1,0)}$	$\frac{1}{(1-\gamma)^2}$	$\epsilon\log\frac{1}{\epsilon}$			
API(α) (équation (6.4)	$C^{(1,0)}$	$\frac{1}{(1-\gamma)^2}$	ϵ	$\frac{1}{\alpha(1-\gamma)}\log\frac{1}{\epsilon}$		
CPI(α)	$C^{(1,0)}$	$\frac{1}{(1-\gamma)^3}$	ϵ	$\frac{1}{\alpha(1-\gamma)}\log\frac{1}{\epsilon}$		
CPI (équation (6.3))	$C^{(1,0)}$	$\frac{1}{(1-\gamma)^3}$	$\epsilon\log\frac{1}{\epsilon}$	$\frac{1}{1-\gamma}\frac{1}{\epsilon}\log\frac{1}{\epsilon}$		
	C_{π_*}	$\frac{1}{(1-\gamma)^2}$	ϵ	$\frac{\gamma}{\epsilon^2}$		(Kakade et Langford, 2002)
PSDP$_\infty$ (équation (6.5)) ($\simeq$ NSPI(∞))	C_{π_*}	$\frac{1}{(1-\gamma)^2}$	$\epsilon\log\frac{1}{\epsilon}$	$\frac{1}{1-\gamma}\log\frac{1}{\epsilon}$		
	$C^{(1)}_{\pi_*}$	$\frac{1}{1-\gamma}$	ϵ	$\frac{1}{1-\gamma}\log\frac{1}{\epsilon}$		
NSPI(ℓ) (équation (6.6))	$C^{(2,\ell,0)}$	$\frac{1}{(1-\gamma)(1-\gamma^\ell)}$	ϵ	$\frac{1}{1-\gamma}\log\frac{1}{\epsilon}$	ℓ	
	$\frac{C^{(1,0)}}{\ell}$	$\frac{1}{(1-\gamma)^2(1-\gamma^\ell)}$	$\epsilon\log\frac{1}{\epsilon}$	$\frac{1}{1-\gamma}\log\frac{1}{\epsilon}$		
	$C^{(1)}_{\pi_*}+\gamma^\ell\frac{C^{(2,\ell,\ell)}}{1-\gamma^\ell}$	$\frac{1}{1-\gamma}$	ϵ	$\frac{1}{1-\gamma}\log\frac{1}{\epsilon}$		
	$C_{\pi_*}+\gamma^\ell\frac{C^{(2,\ell,0)}}{m(1-\gamma^\ell)}$	$\frac{1}{(1-\gamma)^2}$	$\epsilon\log\frac{1}{\epsilon}$	$\frac{1}{1-\gamma}\log\frac{1}{\epsilon}$		

TABLE 6.1 – Comparaison des algorithmes.

Finalement, soit C_{π_*} le plus petite constante de $[1,\infty)\cup\{\infty\}$ telle que

$$d_{\pi_*,\mu} = (1-\gamma)\mu(I-\gamma P_{\pi_*})^{-1} \leq C_{\pi_*}\nu.$$

Avec ces notations, notre première contribution consiste à fournir une comparaison approfondie de tous les algorithmes. Ceci est fait Table 6.1. Pour chaque algorithme, nous décrivons une ou plusieurs bornes de performance, le nombre d'itérations et la mémoire requis correspondants. Pour garder une présentation claire, nous montrons seulement la dépendance de ces quantités en fonction des coefficients de concentration, du facteur d'actualisation γ, de la qualité ϵ de l'opérateur gourmand approché, et—le cas échéant—du paramètre principal (α, ℓ) des algorithmes. Pour API(α), CPI(α), CPI et PSDP$_\infty$, la mémoire requise égale le nombre d'itérations. La dernière colonne contient des références pour les bornes existantes dans la littérature. Toutes les bornes (sauf deux) sont des contributions originales.

Preuves des bornes de performance de la Table 6.1.
PSDP$_\infty$:
Pour tout k, nous avons

$$\begin{aligned} v_{\pi_*} - v_{\sigma_k} &= T_{\pi_*}v_{\pi_*} - T_{\pi_*}v_{\sigma_{k-1}} + T_{\pi_*}v_{\sigma_{k-1}} - T_{\pi_k}v_{\sigma_{k-1}} \\ &\leq \gamma P_{\pi_*}(v_\pi - v_{\sigma_{k-1}}) + e_k \end{aligned}$$

où nous avons défini $e_k = \max_{\pi'} T_{\pi'}v_{\sigma_{k-1}} - T_{\pi_k}v_{\sigma_{k-1}}$. Comme P_π est positive, nous déduisons par induction :

$$v_{\pi_*} - v_{\sigma_k} \leq \sum_{i=0}^{k-1}(\gamma P_{\pi_*})^i e_{k-i} + \gamma^k V_{\max}.$$

En multipliant des deux côtés par μ, utilisant la définition des coefficients c_{π_*} et le fait que $\nu_j e_j \leq \epsilon_j \leq \epsilon$, nous obtenons :

$$\begin{aligned} \mu(v_{\pi_*} - v_{\sigma_k}) &\leq \sum_{i=0}^{k-1}\mu(\gamma P_{\pi_*})^i e_{k-i} + \gamma^k V_{\max} \\ &\leq \sum_{i=0}^{k-1}\gamma^i c_{\pi_*}(i)\epsilon_{k-i} + \gamma^k V_{\max} \\ &\leq \left(\sum_{i=0}^{k-1}\gamma^i c_{\pi_*}(i)\right)\epsilon + \gamma^k V_{\max}. \end{aligned} \tag{6.7}$$

La borne en fonction de $C^{(1)}_{\pi_*}$ est obtenue en utilisant le fait que $v_{\sigma_k\infty} \geq v_{\sigma_k} - \gamma^k V_{\max}$ et en prenant $k \geq \left\lceil \frac{\log \frac{2V_{\max}}{\epsilon}}{1-\gamma} \right\rceil$. Repartant de l'équation (6.7) et utilisant la définition de C_{π_*} (en particulier le fait que pour tout i, $\mu(\gamma P_{\pi_*})^i \leq \frac{1}{1-\gamma}d_{\pi^*,\mu} \leq \frac{C_{\pi_*}}{1-\gamma}\nu$) et le fait que $\nu e_j \leq \epsilon_j$, nous obtenons :

$$\begin{aligned} \mu(v_{\pi_*} - v_{\sigma_k}) &\leq \sum_{i=0}^{k-1}\mu(\gamma P_{\pi_*})^i e_{k-i} + \gamma^k V_{\max} \\ &\leq \frac{C_{\pi_*}}{1-\gamma}\sum_{i=1}^{k}\epsilon_i + \gamma^k V_{\max} \end{aligned}$$

et la deuxième borne est obtenue en utilisant le fait que $v_{\sigma_{k\cdots}} \geq v_{\sigma_k} - \gamma^k V_{\max}$, $\sum_{i=1}^{k} \epsilon_i \leq k\epsilon$, et en considérant le nombre d'itérations $k = \left\lceil \frac{\log \frac{2V_{\max}}{\epsilon}}{1-\gamma} \right\rceil$.

API/NSPI(ℓ) :

API est identique à NSPI(1), et ses bornes sont des cas particuliers des deux premières bornes de NSPI(ℓ), donc nous considérons seulement NSPI(ℓ). En suivant la technique de preuve du théorème 5.6 page 74, notant $\Gamma_{k,\ell} = (\gamma P_{\pi_k})(\gamma P_{\pi_{k-1}}) \cdots (\gamma P_{\pi_{k-\ell+1}})$ et $e_{k+1} = \max_{\pi'} T_{\pi'} v_{\pi_{k,\ell}} - T_{\pi_{k+1}} v_{\pi_{k,\ell}}$, on peut montrer que :

$$v_{\pi_*} - v_{\pi_{k,\ell}} \leq \sum_{i=0}^{k-1} (\gamma P_{\pi_*})^i (I - \Gamma_{k-i,\ell})^{-1} e_{k-i} + \gamma^k V_{\max}.$$

En multipliant les deux côtés par μ (et observant que $e_k \geq 0$) et le fait que $\nu e_j \leq \epsilon_j \leq \epsilon$, nous obtenons

$$\begin{aligned}
\mu(v_{\pi_*} - v_{\pi_k}) &\leq \sum_{i=0}^{k-1} \mu(\gamma P_{\pi_*})^i (I - \Gamma_{k-i,\ell})^{-1} e_{k-i} + \gamma^k V_{\max} && (6.8)\\
&\leq \sum_{i=0}^{k-1} \left(\sum_{j=0}^{\infty} \gamma^{i+j\ell} c(i + j\ell) \epsilon_{k-i} \right) + \gamma^k V_{\max} && (6.9)\\
&\leq \sum_{i=0}^{k-1} \sum_{j=0}^{\infty} \gamma^{i+j\ell} c(i + j\ell) \epsilon + \gamma^k V_{\max}, && (6.10)
\end{aligned}$$

ce qui prouve la première borne en prenant $k \geq \left\lceil \frac{\log \frac{2V_{\max}}{\epsilon}}{1-\gamma} \right\rceil$. Repartant de l'équation (6.9) et faisant l'hypothèse (pour simplifier l'écriture) que $\epsilon_{-k} = 0$ pour tout $k \geq 0$, nous obtenons

$$\begin{aligned}
\mu(v_{\pi_*} - v_{\pi_k}) - \gamma^k V_{\max} &\leq \sum_{l=0}^{\lceil \frac{k-1}{\ell} \rceil} \sum_{h=0}^{\ell-1} \sum_{j=0}^{\infty} \gamma^{h+(l+j)\ell} c(h + (l+j)\ell) \epsilon_{k-h-l\ell}\\
&\leq \sum_{l=0}^{\lceil \frac{k-1}{\ell} \rceil} \sum_{h=0}^{\ell-1} \sum_{j=l}^{\infty} \gamma^{h+j\ell} c(h + j\ell) \max_{k-(l+1)\ell+1 \leq p \leq k-l\ell} \epsilon_p\\
&\leq \sum_{l=0}^{\lceil \frac{k-1}{\ell} \rceil} \sum_{h=0}^{\ell-1} \sum_{j=0}^{\infty} \gamma^{h+j\ell} c(h + j\ell) \max_{k-(l+1)\ell+1 \leq p \leq k-l\ell} \epsilon_p\\
&= \left(\sum_{h=0}^{\ell-1} \sum_{j=0}^{\infty} \gamma^{h+j\ell} c(h + j\ell) \right) \sum_{l=0}^{\lceil \frac{k-1}{\ell} \rceil} \max_{l-(l+1)\ell+1 \leq p \leq k-l\ell} \epsilon_p\\
&\leq \left(\sum_{i=0}^{\infty} \gamma^i c(i) \right) \left\lceil \frac{k-1}{\ell} \right\rceil \epsilon, && (6.11)
\end{aligned}$$

ce qui prouve la deuxième borne en prenant $k = \left\lceil \frac{\log \frac{2V_{\max}}{\epsilon}}{1-\gamma} \right\rceil$. Finalement, repartant de l'équation (6.8), et utilisant le fait que $(I - \Gamma_{k-i,\ell})^{-1} = I + \Gamma_{k-i,\ell}(I - \Gamma_{k-i,\ell})^{-1}$, on peut voir que

$$\mu(v_{\pi_*} - v_{\pi_k}) - \gamma^k V_{\max} \leq \sum_{i=0}^{k-1} \mu(\gamma P_{\pi_*})^i e_{k-i} + \sum_{i=0}^{k-1} \mu(\gamma P_{\pi_*})^i \Gamma_{k-i,\ell} (I - \Gamma_{k-i,\ell})^{-1} e_{k-i}.$$

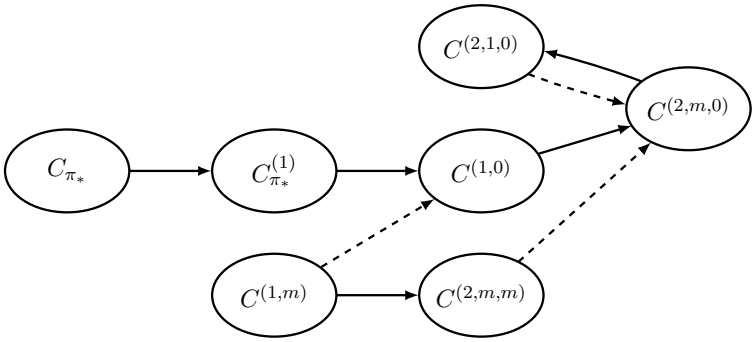


FIGURE 6.1 – Hiérarchie des constantes de concentrabilité

Le premier terme de la partie droite de l'inégalité peut être borné exactement comme dans le cas de PSDP_∞. Pour le deuxième terme, nous avons

$$\sum_{i=0}^{k-1} \mu(\gamma P_{\pi_*})^i \Gamma_{k-i,\ell}(I - \Gamma_{k-i,\ell})^{-1} e_{k-i} \leq \sum_{i=0}^{k-1} \sum_{j=1}^{\infty} \gamma^{i+j\ell} c(i+j\ell)\epsilon_{k-i}$$
$$= \gamma^\ell \sum_{i=0}^{k-1} \sum_{j=0}^{\infty} \gamma^{i+j\ell} c(i+(j+1)\ell)\epsilon_{k-i},$$

et on suit la même démarche que précédemment (de l'équation (6.9) aux équations (6.10) et (6.11)) pour conclure.

CPI, CPI(α), API(α) :
Les pas conservatifs peuvent être analysés via une généralisation fastidieuse de l'analyse d'API par Munos (2003). Nous renvoyons le lecteur à (Scherrer, 2014, Appendice C) pour les détails. □

Notre deuxième contribution, qui complète notre liste comparative de bornes, est que nous pouvons montrer qu'il existe une hiérarchie parmi les constantes qui apparaissent dans les bornes de la Table 6.1. Dans le graphe dirigé de la figure 6.1, une constante B est une descendante de A *si et seulement si* l'implication $\{B < \infty \Rightarrow A < \infty\}$ est vraie [4]. L'équivalence "si et seulement si" est ici importante : elle signifie que si A est un parent de B, et B n'est pas un parent de A, alors il existe un MDP pour lequel A est finie tandis que B est infinie ; en d'autres termes, une garantie exprimée en fonction de la constante A peut être arbitrairement meilleure qu'une garantie exprimée en fonction de B. Ainsi, la meilleure constante de concentrabilité est C_{π_*}, tandis que les pires sont $C^{(2,1,0)}$ et $C^{(2,\ell,0)}$. Pour finir de compléter notre étude, ajoutons que pour tout MDP et toute distribution μ, il est possible de trouver une distribution ν pour l'algorithme (on rappelle que les coefficients de concentration dépendent de ν et μ) tel que C_{π_*} est fini, alors que ce n'est pas le cas pour $C^{(1)}_{\pi_*}$ (et par voie de conséquence pour toutes les autres constantes). Autrement dit, seuls les algorithmes possédant une borne d'erreur exprimée en fonction de la constante C_{π_*} peuvent être assurés (si le paramètre ν est bien choisi !) d'avoir une garantie de performance informative.

Preuves des relations d'ordre entre les constantes de concentrabilité (Figure 6.1).
$C_{\pi_*} \to C^{(1)}_{\pi_*}$ **:**

4. Les flèches en ligne pointillée sont utilisées pour souligner le fait que la comparaison des coefficients est restreinte au cas où le paramètre ℓ est fini.

(i) Nous avons $C_{\pi_*} \leq C^{(1)}_{\pi_*}$ puisque :

$$d_{\pi_*,\mu} = (1-\gamma)\mu(I-\gamma P_{\pi_*})^{-1} = (1-\gamma)\sum_{i=0}^{\infty}\gamma^i\mu(P_{\pi_*})^i \leq (1-\gamma)\sum_{i=0}^{\infty}\gamma^i c_{\pi_*}(i)\nu = C^{(1)}_{\pi_*}\nu$$

et le fait que C_{π_*} est la plus petite constante vérifiant cette inégalité. (ii) Nous pouvons avoir $C_{\pi_*} < \infty$ et $C^{(1)}_{\pi_*} = \infty$ en considérant un MDP sur $\mathbb{N}$ où π_* induit une transition déterministe de l'état i vers l'état $i+1$.

$\underline{C^{(1)}_{\pi_*} \to C^{(1,0)}}$ **:**
(i) Nous avons $C^{(1)}_{\pi_*} \leq C^{(1,0)}$ parce que pour tout i, $c_{\pi_*}(i) \leq c(i)$. (ii) Il est facile d'obtenir $C^{(1)}_{\pi_*} < \infty$ et $C^{(1,0)} = \infty$ en utilisant deux politiques distinctes.

$\underline{C^{(1,0)} \to C^{(2,\ell,0)}}$ **et** $\underline{C^{(1,\ell)} \to C^{(2,\ell,\ell)}}$ **:**
(i) $C^{(1,\ell)} \leq \frac{1}{1-\gamma^\ell}C^{(2,\ell,\ell)}$ est vrai car

$$\frac{C^{(1,\ell)}}{1-\gamma} = \sum_{i=0}^{\infty}\gamma^i c(i+\ell) \leq \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\gamma^{i+j\ell}c(i+(j+1)\ell) = \frac{1}{(1-\gamma)(1-\gamma^\ell)}C^{(2,\ell,\ell)}.$$

(ii) On peut avoir $C^{(1,\ell)} < \infty$ et $C^{(2,\ell,\ell)} = \infty$ lorsque $c(i) = \Theta(\frac{1}{i^2\gamma^i})$, car le terme générique de $C^{(1,\ell)}$ est $\Theta(\frac{1}{i^2})$ (la série converge) tandis que celui de $C^{(2,\ell,\ell)}$ est $\Theta(\frac{1}{i})$ (la série diverge). Le raisonnement est similaire pour l'autre relation.

$\underline{C^{(1,\ell)} \to C^{(1,0)}}$ **et** $\underline{C^{(2,\ell,\ell)} \to C^{(2,\ell,0)}}$ **:**
Nous faisons ici l'hypothèse que $\ell < \infty$. (i) Nous avons $C^{(1,\ell)} \leq \frac{1}{\gamma^\ell}C^{(1,0)}$ et $C^{(2,\ell,\ell)} \leq \frac{1}{\gamma^\ell}C^{(2,\ell,0)}$. (ii) Il suffit d'avoir $c(j) = \infty$ pour un $j < \ell$ pour avoir $C^{(2,\ell,0)} = \infty$ tandis que $C^{(2,\ell,\ell)} < \infty$, ou pour avoir $C^{(1,0)} = \infty$ tandis que $C^{(1,\ell)} < \infty$.

$\underline{C^{(2,1,0)} \leftrightarrow C^{(2,\ell,0)}}$ **:**
(i) Nous avons clairement $C^{(2,\ell,0)} \leq \frac{1-\gamma^\ell}{1-\gamma}C^{(2,1,0)}$. (ii) $C^{(2,\ell,0)}$ peut être réécrit comme suit :

$$C^{(2,\ell,0)} = (1-\gamma)(1-\gamma^\ell)\sum_{i=0}^{\infty}\left(1+\left\lfloor\frac{i}{\ell}\right\rfloor\right)\gamma^i c(i).$$

Alors, en utilisant le fait que $1+\lfloor\frac{i}{\ell}\rfloor \geq \max\left(1,\frac{i}{\ell}\right)$, nous avons

$$\begin{aligned}
\frac{1-\gamma}{1-\gamma^\ell}C^{(2,\ell,0)} &\geq \sum_{i=0}^{\infty}\max\left(1,\frac{i}{\ell}\right)\gamma^i c(i) \\
&\geq \sum_{i=0}^{\ell-1}\gamma^i c(i) + \sum_{i=m}^{\infty}\frac{i}{\ell}\gamma^i c(i) \\
&\geq \sum_{i=0}^{\ell-1}\gamma^i c(i) + \frac{\ell}{\ell+1}\sum_{i=m}^{\infty}\frac{i+1}{\ell}\gamma^i c(i) \\
&= \sum_{i=0}^{\ell-1}\gamma^i c(i) + \frac{\ell}{\ell+1}\left(C^{(2,1,0)} - \sum_{i=0}^{\ell-1}\gamma^i c(i)\right) \\
&= \frac{\ell}{\ell+1}C^{(2,1,0)} + \frac{1}{\ell+1}\sum_{i=0}^{\ell-1}\gamma^i c(i).
\end{aligned}$$

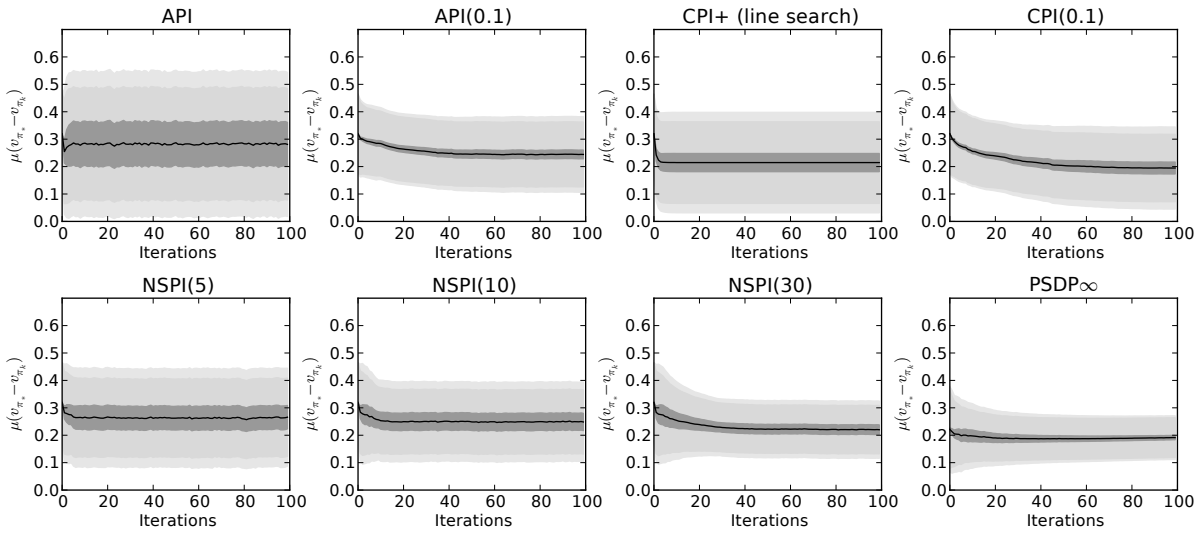


FIGURE 6.2 – **Statistiques pour toutes les instances.** Les MDPs $(M_i)_{1 \leq i \leq 30}$ sont i.i.d. et ont la même distribution que M_1. Conditionnellement à un MDP M_i et un algorithme, les mesures d'erreur pour toutes les itérations k sont i.i.d. et ont la même distribution que $L_{1,k}$. La ligne centrale des courbes d'apprentissage donne un estimé empirique de la moyenne globale$(\mathbb{E}[L_{1,k}])_k$. Les trois régions grisées (du plus foncé au plus clair) sont respectivement des estimés de la variabilité (sur les MDPs) $(\mathbb{E}[\mathrm{Std}[L_{1,k}|M_1]])_k$ de l'erreur moyenne, la moyenne (sur les MDPs) $(\mathbb{E}[\mathrm{Std}[L_{1,k}|M_1]])_k$ de la déviation standard de l'erreur, et la variabilité (sur les MDPs) $(\mathrm{Std}[\mathrm{Std}[L_{1,k}|M_1]])_k$ de la déviation standard de l'erreur. Pour facilier les comparaisons, toutes les courbes sont affichés avec les même axes x et y

Ainsi, quand ℓ est fini, $C^{(2,\ell,0)} < \infty \Rightarrow C^{(2,1,0)} < \infty$.

□

L'algorithme standard API jouit de garanties exprimées en fonction de $C^{(2,1,0)}$ et $C^{(1,0)}$ seulement. Comme CPI a une borne de performance en fonction de C_{π_*}, il a une garantie de performance qui peut être arbitrairement meilleure que celle d'API, tandis que le contraire n'est pas vrai. Ceci, cependant, se fait au prix d'une augmentation exponentielle de la complexité en temps, car CPI requiert un nombre d'itération de l'ordre de $O(\frac{1}{\epsilon^2})$, tandis que la garantie d'API est obtenue après seulement $O\left(\log \frac{1}{\epsilon}\right)$ itérations. Quand l'analyse de CPI est relaxée de sorte à exprimer la borne de performance en fonction de la (moins bonne) constante $C^{(1,0)}$ (utilisée aussi pour API), nous pouvons légèrement améliorer la vitesse—$\tilde{O}\left(\frac{1}{\epsilon}\right)$—, bien qu'elle soit toujours exponentiellement moins bonne que celle d'API. Ce résultat est prouvé à l'aide d'une technique qui a aussi été utilisée pour CPI(α) et API(α). Nous conjecturons que la borne pour CPI(α) peut être améliorée, et notamment qu'elle doit être aussi bonne que celle de CPI dès lors que le coefficient d'apprentissage α est suffisamment petit.

PSDP$_\infty$ jouit de deux garanties, et a une vitesse de convergence rapide comme celle d'API. Une borne a une meilleure dépendance par rapport à $\frac{1}{1-\gamma}$, mais est exprimée en fonction de la constante moins bonne $C^{(1)}_{\pi_*}$. La seconde garantie est presque aussi bonne que celle de CPI dans la mesure où elle contient seulement un terme $\log \frac{1}{\epsilon}$ en plus, mais a la propriété sympathique d'être garantie rapidement par rapport à ϵ : en un temps $O(\log \frac{1}{\epsilon})$ au lieu de $O(\frac{1}{\epsilon^2})$, c'est-à-dire exponentiellement plus vite. Ainsi, PSDP$_\infty$ est un algorithme qui est en théorie meilleur que CPI (aussi bon mais plus rapide) et API (aussi rapide mais meilleur).

Maintenant, d'un point de vue pratique, PSDP et CPI doivent stocker toutes les politiques générées depuis le début. La mémoire requise par ces algorithmes est ainsi proportionnelle au nombre d'itérations. Même si PSDP_∞ a besoin d'un nombre d'itérations bien moindre que CPI, le besoin de mémoire correspondant peut être problématique lorsque ϵ est petit ou γ est proche de 1. Nous avons expliqué que NSPI(ℓ) peut être vu comme généralisant API et PSDP_∞. Comme (i) les deux ont une bonne complexité en temps, (ii) API a la meilleure complexité en mémoire, et (iii) PSDP_∞ a la meilleure garantie de performance, NSPI(ℓ) est un bon candidat pour faire un compromis entre la garantie et la mémoire. Si, dans la Table 6.1, les deux premières bornes pour cet algorithme étendent celle d'API, les deux suivantes sont composées de deux termes : les termes de gauche sont identiques à ceux obtenus pour PSDP_∞, tandis que les termes de droite, nouveaux, sont contrôlés par γ^ℓ, qui peut être rendu arbitrairement petit en augmentant le paramètre de mémoire ℓ. Notre analyse confirme ainsi l'intuition que NSPI(ℓ) permet de faire un compromis performance/mémoire entre API (plus petite mémoire) et PSDP_∞ (meilleure performance). En d'autres termes, dès lors que la mémoire devient une contrainte, NSPI(ℓ) constitue l'alternative naturelle à PSDP_∞.

6.3 Simulations numériques

Dans cette section, nous présentons des simulations numériques pour illustrer le comportement empirique des différents algorithmes considérés dans le chapitre. Nous considérons le schéma standard API comme référence. CPI, tel qu'il est décrit dans Kakade et Langford (2002), est très lent (sur une expérience impliquant un problème avec 100 états, il a progressé très lentement et a mis plusieurs millions d'itérations pour s'arrêter) et nous ne l'avons pas évalué dans sa forme originelle. A la place, nous avons considéré deux variations : CPI+ qui est identique à CPI sauf qu'il choisit un pas d'apprentissage α_k à chaque itération en faisant une recherche linéaire dans la direction de la politique fournie par l'opérateur gourmand approché [5], et CPI(α) avec $\alpha = 0.1$, qui effectue des pas "relativement petits mais pas trop" à chaque itération. Afin d'évaluer l'utilité pour CPI d'utiliser la distribution $d_{\nu,\pi}$ lors du choix de la politique approximativement gourmande, nous avons considéré également API(α) avec $\alpha = 0.1$, la variation conservative d'API décrite à l'équation (6.4) qui fait des petits pas, et qui diffère seulement de CPI(α) par le fait que l'opérateur approximativement gourmand utilise la distribution ν au lieu de $d_{\pi_k,\nu}$. En plus de ces algorithmes, nous considérons PSDP_∞et NSPI(ℓ) avec les valeurs $\ell \in \{5, 10, 30\}$.

Afin de mesurer précisément leur qualité, nous considérons des MDPs de taille finie pour lesquels la fonction de la valeur optimale peut être calculée exactement. Plus précisément, nous considérons des problèmes "Garnet" introduits par Archibald *et al.* (1995), qui sont une classe de MDPs finis construits aléatoirement. Ils ne correspondent à aucune application particulière, mais sont représentatifs de MDPs que l'on peut rencontrer en pratique. En bref, nous considérons des problèmes Garnet avec $|\mathcal{S}| \in \{50, 100, 200\}$, $|\mathcal{A}| \in \{2, 5, 10\}$, et un facteur de branchement dans $\{1, 2, 10\}$. L'opérateur gourmand approché utilisé par tous les algorithmes est implémenté comme l'opérateur gourmand exact appliqué à une projection bruitée de la vraie fonction de valeur sur un espace linéaire de dimension $\frac{|\mathcal{S}|}{10}$. Cette projection est une projection orthogonale par rapport à la norme quadratique pondérée par ν ou $d_{\nu,\pi}$ (pour CPI+ et CPI(α)) où ν est uniforme.

Pour chacune des $3^3 = 27$ instances des paramètres de problèmes Garnet, nous générons 30 MDPs Garnet i.i.d. $(M_i)_{1\leq i\leq 30}$. Pour chaque MDP M_i, nous exécutons API, API(0.1), CPI+,

5. Nous avons implémenté un mécanisme simpliste de recherche linéaire, qui considère l'ensemble des pas d'apprentissage $2^i\alpha$ où α est le pas minimal déterminé par CPI pour garantir une amélioration (cf. Kakade et Langford (2002)).

CPI(0.1), NSPI(ℓ) pour $\ell \in \{5, 10, 30\}$ et PSDP$_\infty$ 30 fois. Pour chaque exécution, nous calculons, à chaque itération $k \in (1, 100)$ la performance, c'est-à-dire la perte $L_{j,k} = \mu(v_{\pi_*} - v_{\pi_k})$ par rapport à la politique optimale. La figure 6.2 montre des statistiques sur ces variables aléatoires. Pour chaque algorithme, nous traçons des courbes d'apprentissage avec des intervalles de confiance qui rendent compte de la variabilité de la performance à travers les différentes exécutions et les différents problèmes tirés aléatoirement. Dans la partie "Compléments" de l'article (Scherrer, 2014), nous présentons des statistiques qui sont conditionnées aux valeurs de n_S, n_A et b, ce qui éclaire sur l'influence de ces paramètres.

A partir de ces expériences et des statistiques, nous pouvons faire une série d'observations. L'algorithme standard API est bien plus variable que les autres et tend à fournir les pires performances en moyenne. CPI+ et CPI(α) ont une performance asymptotique moyenne similaire. Si CPI(α) montre un peu moins de variabilité, il est beaucoup plus lent que CPI+ qui converge toujours en un petit nombre d'itérations (la plupart du temps en moins de 10 itérations, et toujours en moins de 20). API(α)—la variation conservative naïve d'API qui est aussi plus simple que CPI(α)—est empiriquement proche de CPI(α), bien qu'étant en moyenne légèrement moins bon. CPI+, CPI(α) et PSDP$_\infty$ ont une performance moyenne similaire, mais la variabilité de PSDP$_\infty$ est significativement moindre. PSDP$_\infty$ apparaît ainsi comme donnant les meilleurs résultats empiriques. NSPI(ℓ) constitue effectivement un pont entre API et PSDP$_\infty$. En augmentant ℓ, le comportement de NSPI(ℓ) se rapproche progressivement de celui de PSDP$_\infty$. Avec $\ell = 30$, NSPI(ℓ) est globalement meilleur que API(α), CPI+, et CPI(α), et est proche de PSDP$_\infty$. Des expérimentations complémentaires ont montré que l'ensemble de ces observations est stable par rapport aux paramètres n_S et n_A. De manière intéressante, les différences entre les algorithmes tendent à se dissiper lorsque la dynamique du problème est de plus en plus stochastique (lorsque le facteur de branchement augmente). Ceci est conforme à notre analyse basée sur les constantes de concentrabilité : celles-ci sont toutes finies lorsque la dynamique est fortement "mélangeante", et leur différences relatives sont les plus grandes pour des dynamiques déterministes.

6.4 Bilan et perspectives

Dans ce chapitre, nous avons considéré plusieurs variations de PI pour les problèmes à horizon infini : API, CPI, NSPI(ℓ), API(α) et PSDP$_\infty$. Nous avons en particulier expliqué le fait que l'algorithme original que nous avons introduit avec Boris Lesner et décrit au chapitre précédent, NSPI(ℓ), généralise API (obtenu lorsque $\ell = 1$) et PSDP$_\infty$ (lorsque $\ell = \infty$). La table 6.1 synthétise les garanties de performance de ces algorithmes. La plupart des bornes qui y sont décrites sont à notre connaissance nouvelles.

L'un des premiers messages important de notre travail est que ce qui est habituellement caché dans les constantes de concentrabilité compte. Les constantes impliquées dans les bornes d'API, CPI, PSDP$_\infty$ et pour les principaux termes (à gauche) de NSPI(ℓ) peuvent être ordonnés de la pire à la meilleure comme suit : $C^{(2,1,0)}, C^{(1,0)}, C^{(1)}_{\pi_*}, C_{\pi_*}$. Une hiérarchie détaillée de toutes ces constantes est fournie à la figure 6.1. A notre connaissance, c'est la première fois qu'une comparaison détaillée et complète des bornes de performance d'algorithmes de programmation dynamique est effectuée, et notre hiérarchie des constantes a des implications qui vont au-delà des schémas de type itération sur les politiques que nous avons considérés ici. En fait, plusieurs autres algorithms de programmation dynamique, notamment AVI (Munos, 2007), λPI (Scherrer, 2013b), et AMPI (décrit au chapitre 4 et dans Scherrer *et al.* (2015)), ont des garanties qui sont exprimées en fonction de la pire constante $C^{(2,1,0)}$, ce qui suggère qu'ils ne devraient pas être compétitifs avec ceux que nous avons décrits ici.

D'un point de vue purement technique, plusieurs de nos bornes sont dérivées par paires ; ceci

est dû au fait que nous utilisons une nouvelle technique de preuve. Celle-ci a permis de dériver une nouvelle borne de performance pour API, qui est meilleure que l'existant car elle implique la constance $C^{(1,0)}$ au lieu de $C^{(2,1,0)}$. Cela nous a également permis de dériver de nouvelles bornes pour CPI (et sa variation algorithmique CPI(α)) qui est moins bonne en termes de garantie mais meilleure du point de vue de la complexité en temps ($\tilde{O}(\frac{1}{\epsilon})$ au lieu de $O(\frac{1}{\epsilon^2})$). Nous pensons que cette nouvelle technique pourrait être utile dans de futures études sur la résolution approchée de MDPs.

Résumons les principaux enseignement de notre étude. 1) Les garanties pour CPI peuvent être arbitrairement meilleures que celles d'API/API(α), parce qu'elle sont exprimées en fonction de la meilleure constante de concentrabilité C_{π_*}, mais ceci se fait au prix d'une augmentation relative—exponentielle en $\frac{1}{\epsilon}$—du nombre d'itérations. 2) PSDP$_\infty$ combine les avantages d'API et CPI : sa garantie de performance est similaire à celle de CPI, mais elle est obtenue en un nombre d'itérations identique à celui d'API. 3) Contrairement à API qui requiert une mémoire constante, la mémoire requise par CPI et PSDP$_\infty$ est proportionnelle au nombre d'itérations, ce qui peut être problématique lorsque le facteur d'actualisation γ est proche de 1, ou lorsque l'erreur d'approximation ϵ est proche de 0 ; nous avons montré que l'algorithme NSPI(ℓ) permet de faire un compromis entre la qualité de performance de PSDP$_\infty$ et la mémoire d'API.

La limite principale de notre analyse réside en l'hypothèse, considérée tout au long de ce chapitre, que tous les algorithmes disposent d'un operateur approximativement gourmand de qualité ϵ. Ceci n'est en général par réaliste, car on ne peut pas contrôler directement le niveau de qualité ϵ. De plus, la comparaison de tous les algorithmes par rapport à cette quantité est limitée, car les problèmes d'optimisation sous-jacents sont de complexités variées : par exemple, les méthodes comme CPI cherchent dans un espace de politiques stochastiques, tandis qu'API se déplace dans un espace de politiques déterministes. Approfondir notre étude en explicitant ce qui est caché dans ce terme ϵ—comme nous l'avons fait dans le chapitre 4 pour AMPI—constitue une perspective naturelle.

Dernier point, et non le moindre, du côté pratique, nous avons effectué des simulations numériques qui confirment notre analyse que des algorithmes avec de meilleures constantes de concentrabilité doivent être préférés. Sur plus de 800 MDPs Garnet avec diverses caractéristiques, CPI(α), CPI+ (CPI avec une recherche linéaire simpliste), PSDP$_\infty$ et NSPI(ℓ) ont des performances qui surpassent significativement celles du schéma standard API. CPI+, CPI(α) et PSDP$_\infty$ ont des performances moyennes similaires, mais PSDP$_\infty$ est bien moins variable, ce qui fait de lui globalement le meilleur algorithme. Finalement, NSPI(ℓ) permet de faire un pont entre API et PSDP$_\infty$, et permet d'atteindre des performances similaires à celle de PSDP$_\infty$ tout en contrôlant la mémoire. Considérer d'autres instances de ces schémas algorithmiques, analyser leurs performances et les tester empiriquement sur des problèmes de plus grande taille constitue l'une des perspectives les plus naturelles de ce travail.

Bibliographie

M. Akian et S. Gaubert : Policy iteration for perfect information stochastic mean payoff games with bounded first return times is strongly polynomial. Rapport technique arxiv 1310.4953v1, 10 2013.

A. Antos, R. Munos et Cs. Szepesvári : Fitted Q-iteration in continuous action-space MDPs. *Dans Neural Information Processing Systems*, pages 9–16, 2007.

A. Antos, Cs. Szepesvári et R. Munos : Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71 (1):89–129, 2008.

T. Archibald, K. McKinnon et L. Thomas : On the generation of Markov decision processes. *Journal of the Operational Research Society*, 46:354–361, 1995.

M. Gheshlaghi Azar, V. Gómez et H.J. Kappen : Dynamic Policy Programming with Function Approximation. *Dans 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15, Fort Lauderdale, FL, USA, 2011.

J.A. Bagnell, S.M. Kakade, A. Ng et J. Schneider : Policy search by dynamic programming. *Dans Neural Information Processing Systems*, volume 16. MIT Press, December 2003.

L. Baird : Residual algorithms : Reinforcement learning with function approximation. *Dans International Conference on Machine Learning*, pages 30–37. Morgan Kaufmann, 1995.

D.P. Bertsekas : Approximate policy iteration : a survey and some new methods. *Journal of Control Theory and Applications*, 9:310–335, 2011. ISSN 1672-6340.

D.P. Bertsekas et S. Ioffe : Temporal differences-based policy iteration and applications in neuro-dynamic programming. Rapport technique, MIT, 1996.

D.P. Bertsekas et J.N. Tsitsiklis : *Neurodynamic Programming.* Athena Scientific, 1996.

A. Boumaza et B. Scherrer : Optimal control subsumes harmonic control. *Dans IEEE International Conference on Robotics and Automation*, pages 2841–2846, Rome, Italy, avril 2007. IEEE.

A. Boumaza et B. Scherrer : Analyse d'un algorithme d'intelligence en essaim pour le fourragement. *Revue d Intelligence Artificielle*, 22(6):791–816, 2008.

J. A. Boyan : Technical update : Least-squares temporal difference learning. *Machine Learning*, 49:233–246, 2002.

H. Burgiel : How to Lose at Tetris. *Mathematical Gazette*, 81:194–200, 1997.

L. Busoniu, A. Lazaric, M. Ghavamzadeh, R. Munos, R. Babuska et B. De Schutter : Least-squares methods for Policy Iteration. *Dans* M. Wiering et M. van Otterlo, éditeurs : *Reinforcement Learning : State of the Art*. Springer, 2011.

P. Canbolat et U. Rothblum : (Approximate) iterated successive approximations algorithm for sequential decision processes. *Annals of Operations Research*, pages 1–12, 2012. ISSN 0254-5330.

D. Choi et B. Van Roy : A generalized Kalman filter for fixed point approximation and efficient temporal-difference learning. *Discrete Event Dynamic Systems*, 16:207–239, 2006.

Y. Engel : *Algorithms and Representations for Reinforcement Learning*. Thèse de doctorat, Hebrew University, 2005.

D. Ernst, P. Geurts et L. Wehenkel : Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

E. Even-dar : Planning in pomdps using multiplicity automata. *Dans Uncertainty in Artificial Intelligence*, pages 185–192, 2005.

C. Fahey : Tetris AI, Computer plays Tetris, 2003. `http://colinfahey.com/tetris/tetris.html`.

A.M. Farahmand, M. Ghavamzadeh, Cs. Szepesvári et S. Mannor : Regularized policy iteration. *Dans Neural Information Processing Systems*, 2008.

A.M. Farahmand, R. Munos et Cs. Szepesvári : Error propagation for approximate policy and value iteration. *Dans Neural Information Processing Systems*, pages 568–576, 2010.

J. Fearnley : Exponential lower bounds for policy iteration. *Dans 37th international colloquium conference on Automata, languages and programming : Part II*, ICALP'10, pages 551–562, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-14161-7, 978-3-642-14161-4.

E.A. Feinberg, J. Huang et B. Scherrer : Modified policy iteration algorithms are not strongly polynomial for discounted dynamic programming. *Operations Research Letters*, 42: 429 – 431, 2014.

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*, 25:75–118, 2006.

V. Gabillon, A. Lazaric, M. Ghavamzadeh et B. Scherrer : Classification-based policy iteration with a critic. *Dans International Conference on Machine Learning*, pages 1049–1056, 2011.

M. Geist et O. Pietquin : Algorithmic survey of parametric value function approximation. *IEEE Transactions on Neural Networks and Learning Systems*, 24(6):845 – 867, 2013.

G.J. Gordon : Stable function approximation in dynamic programming. *Dans International Conference on Machine Learning*, pages 261–268, 1995.

C. Guestrin, D. Koller et R. Parr : Max-norm projections for factored mdps. *Dans IJCAI*, 2001.

C. Guestrin, D. Koller, R. Parr et S. Venkataraman : Efficient Solution Algorithms for Factored MDPs. *Journal of Artificial Intelligence Research (JAIR)*, 19:399–468, 2003.

L. Györfi, M. Kholer, A. Krzyzak et H. Walk : *A Distribution-Free Theory of Nonparametric Regression*. Springer-Verlag, New York, 2002.

T.D. Hansen, P.B. Miltersen et U. Zwick : Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *J. ACM*, 60(1):1 :1–1 :16, février 2013. ISSN 0004-5411.

T.D. Hansen et U. Zwick : Lower bounds for Howard's algorithm for finding minimum mean-cost cycles. *Dans ISAAC (1)*, pages 415–426, 2010.

R. Hollanders, J.C. Delvenne et R. Jungers : The complexity of policy iteration is exponential for discounted markov decision processes. *Dans IEEE conference on Decision and control*, 2012.

S.M. Kakade : A natural policy gradient. *Dans Neural Information Processing Systems*, pages 1531–1538, 2002.

S.M. Kakade : *On the Sample Complexity of Reinforcement Learning*. Thèse de doctorat, University College London, 2003.

S.M. Kakade et J. Langford : Approximately Optimal Approximate Reinforcement Learning. *Dans International Conference on Machine Learning*, pages 267–274, 2002.

M.J. Kearns et S.P. Singh : Bias-variance error bounds for temporal difference updates. *Dans* Nicolò Cesa-Bianchi et Sally A. Goldman, éditeurs : *COLT*, pages 142–147. Morgan Kaufmann, 2000.

M. Lagoudakis et R. Parr : Reinforcement Learning as Classification : Leveraging Modern Classifiers. *Dans International Conference on Machine Learning*, pages 424–431, 2003a.

M. G. Lagoudakis et R. Parr : Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003b.

A. Lazaric, M. Ghavamzadeh et R. Munos : Analysis of a Classification-based Policy Iteration Algorithm. *Dans International Conference on Machine Learning*, pages 607–614, 2010.

A. Lazaric, M. Ghavamzadeh et R. Munos : Finite-sample analysis of least-squares policy iteration. *Journal of Machine Learning Research*, 13:3041–3074, octobre 2012.

B. Lesner et B. Scherrer : Non-Stationary Approximate Modified Policy Iteration. *Dans ICML 2015*, Lille, France, juillet 2015.

O.A. Maillard, R. Munos, A. Lazaric et M. Ghavamzadeh : Finite Sample Analysis of Bellman Residual Minimization. *Dans* Masashi Sugiyama et Qiang Yang, éditeurs : *Asian Conference on Machine Learpning. JMLR : Workshop and Conference Proceedings*, volume 13, pages 309–324, 2010.

Y. Mansour et S.P. Singh : On the complexity of policy iteration. *Dans Uncertainty in Artificial Intelligence*, pages 401–408, 1999.

M. Melekopoglou et A. Condon : On the complexity of the policy improvement algorithm for Markov decision processes. *INFORMS Journal on Computing*, 6(2):188–192, 1994.

R. Munos : Error bounds for approximate policy iteration. *Dans International Conference on Machine Learning*, 2003.

R. Munos : Performance Bounds in L_p-norm for Approximate Value Iteration. *SIAM Journal on Control and Optimization*, 46(2):541–561, 2007.

R. Munos et Cs. Szepesvári : Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9:815–857, 2008. ISSN 1532-4435.

A. Nedic et D.P. Bertsekas : Least squares policy evaluation algorithms with linear function approximation. *Theory and Applications*, 13:79–110, 2002.

M. Petrik et B. Scherrer : Biasing Approximate Dynamic Programming with a Lower Discount Factor. *Dans Neural Information Processing Systems*, Vancouver, Canada, 2008.

J. Pineau, G.J. Gordon et S. Thrun : Point-based value iteration : An anytime algorithm for POMDPs. *Dans International Joint Conference on Artificial Intelligence*, volume 18, pages 1025–1032, 2003.

B.A. Pires et Cs. Szepesvári : Statistical linear estimation with penalized estimators : an application to reinforcement learning. *Dans ICML*, pages 1535–1542, 2012.

I. Post et Y. Ye : The simplex method is strongly polynomial for deterministic Markov decision processes. Rapport technique, arXiv :1208.5083v2, 2012.

M. Puterman : *Markov Decision Processes*. Wiley, New York, 1994.

M. Puterman et M. Shin : Modified policy iteration algorithms for discounted Markov decision problems. *Management Science*, 24(11), 1978.

L.C.G. Rogers : Pathwise stochastic optimal control. *SIAM J. Control Optim.*, 46(3):1116–1132, 2007. ISSN 0363-0129.

R. Rubinstein et D. Kroese : *The cross-entropy method : A unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*. Springer-Verlag, 2004.

B. Scherrer : Performance bounds for lambda policy iteration. *CoRR*, abs/0711.0694, 2007.

B. Scherrer : Should one compute the temporal difference fix point or minimize the bellman residual ? the unified oblique projection view. *Dans International Conference on Machine Learning*, 2010.

B. Scherrer : Improved and Generalized Upper Bounds on the Complexity of Policy Iteration. *Dans Neural Information Processing Systems*, South Lake Tahoe, United States, décembre 2013a.

B. Scherrer : Performance Bounds for λ-Policy Iteration and Application to the Game of Tetris. *Journal of Machine Learning Research*, 14:1175–1221, 2013b.

B. Scherrer : Approximate Policy Iteration Schemes : A Comparison. *Dans ICML - 31st International Conference on Machine Learning - 2014*, Pékin, China, juin 2014. URL `https://hal.inria.fr/hal-00989982`.

B. Scherrer : Improved and Generalized Upper Bounds on the Complexity of Policy Iteration. *Mathematics of Operations Research*, 2016. A paraître.

B. Scherrer, M. Ghavamzadeh, V. Gabillon et M. Geist : Approximate Modified Policy Iteration. *Dans 29th International Conference on Machine Learning - ICML 2012*, Edinburgh, United Kingdom, juin 2012.

B. Scherrer, M. Ghavamzadeh, V. Gabillon, B. Lesner et M. Geist : Approximate Modified Policy Iteration and its Application to the Game of Tetris. *Journal of Machine Learning Research*, page 47, 2015. A paraître.

B. Scherrer et B. Lesner : On the use of non-stationary policies for stationary infinite-horizon Markov decision processes. *Dans Neural Information Processing Systems*, South Lake Tahoe, United States, décembre 2012.

B. Scherrer et C. Thiéry : Performance bound for approximate optimistic policy iteration. Technical report, INRIA, 2010.

N. Schmitz : How good is Howard's policy improvement algorithm ? *Zeitschrift für Operations Research*, 29(7):315–316, 1985. ISSN 0340-9422.

R. Schoknecht : Optimality of reinforcement learning algorithms with linear function approximation. *Dans Neural Information Processing Systems*, pages 1555–1562, 2002.

P.J. Schweitzer et A. Seidmann : Generalized polynomial approximations in markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110(2):568 – 582, 1985.

S. Singh et R. Yee : An Upper Bound on the Loss from Approximate Optimal-Value Functions. *Machine Learning*, 16-3:227–233, 1994.

D.A. Spielman et S.H. Teng : Smoothed analysis of algorithms : Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, mai 2004.

R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, Cs. Szepesvári et E. Wiewiora : Fast gradient-descent methods for temporal-difference learning with linear function approximation. *Dans International Conference on Machine Learning*, 2009.

R.S. Sutton : Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

R.S. Sutton et A.G. Barto : *Reinforcement Learning : An Introduction*. MIT Press, 1998. ISBN 0262193981.

Cs. Szepesvári : *Algorithms for Reinforcement Learning*. Morgan and Claypool, 2010.

I. Szita et A. Lőrincz : Learning Tetris Using the Noisy Cross-Entropy Method. *Neural Computation*, 18(12):2936–2941, 2006.

D.B. Szyld : The many proofs of an identity on the norm of oblique projections. *Numerical Algorithms*, 42:309–323, 2006.

M. Tagorti : *Sur les abstractions et projections des processus décisionnels de Markov de grande taille*. Thèse de doctorat, Université de Lorraine, 2015.

M. Tagorti et B. Scherrer : On the Rate of Convergence and Error Bounds for LSTD(λ). *Dans ICML 2015*, Lille, France, juillet 2015.

C. Thiéry et B. Scherrer : Building Controllers for Tetris. *International Computer Games Association Journal*, 32:3–11, 2009a.

C. Thiéry et B. Scherrer : Improvements on Learning Tetris with Cross Entropy. *International Computer Games Association Journal*, 32, 2009b.

C. Thiéry et B. Scherrer : Least-squares λ-policy iteration : Bias-variance trade-off in control problems. *Dans International Conference on Machine Learning*, pages 1071–1078, 2010.

A.C. Thompson : *Minkowski Geometry*. Cambridge University Press, 1996.

J.N. Tsitsiklis et B. Van Roy : Feature-Based Methods for Large Scale Dynamic Programming. *Machine Learning*, 22(1-3):59–94, 1996.

J.N. Tsitsiklis et B. Van Roy : An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.

V.N. Vapnik : *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0-387-94559-8.

R.J. Williams et L. C. Baird : Tight performance bounds on greedy policies based on imperfect value functions. Rapport technique, College of Computer Science, Northeastern University, 1993.

Y. Ye : The simplex and policy-iteration methods are strongly polynomial for the markov decision problem with a fixed discount rate. *Math. Oper. Res.*, 36(4):593–603, 2011.

B. Yu : Rates of convergence for empirical processes stationnary mixing consequences. *The Annals of Probability*, 19:3041–3074, 1994.

H. Yu : Convergence of least-squares temporal difference methods under general conditions. *Dans International Conference on Machine Learning*, 2010.

H. Yu et D.P. Bertsekas : Error bounds for approximations from projected linear equations. *Mathematics of Operations Research*, 35(2):306–329, 2010.

H. Yu et D.P. Bertsekas : Q-learning and policy iteration algorithms for stochastic shortest path problems. *Annals OR*, 208(1):95–132, 2013.