Une approche modifiée de Lambda-Policy Iteration

Christophe Thiery, Bruno Scherrer

Une approche modifiée de λ-Policy Iteration

Christophe Thiéry, Bruno Scherrer

LORIA - INRIA Lorraine
Campus Scientifique BP 239
54506 Vandoeuvre-lès-Nancy CEDEX
christophe.thiery@loria.fr
http://www.loria.fr/∼thierych

Résumé : Dans le cadre du contrôle optimal stochastique, nous proposons une manière modifiée de mettre en oeuvre l'algorithme λ-Policy Iteration (Bertsekas & Tsitsiklis, 1996), une méthode qui généralise Value Iteration et Policy Iteration en introduisant un paramètre λ. Nous montrons que cette version modifiée, qui est analogue à Modified Policy Iteration, généralise tous ces algorithmes et converge vers la fonction de valeur optimale. En nous appuyant sur des arguments analytiques et expérimentaux, nous mettons en évidence le fait que lorsque l'algorithme est appliqué de manière exacte, le paramètre λ ne permet pas d'améliorer la vitesse de convergence de manière significative.

Mots-clés : Contrôle optimal stochastique, Apprentissage par renforcement, Programmation dynamique, Processus Décisionnels de Markov, Modified λ-Policy Iteration

Introduction

Bertsekas & Tsitsiklis (1996) ont proposé l'algorithme λ-Policy Iteration, une méthode qui généralise les deux algorithmes classiques de la programmation dynamique, Value Iteration et Policy Iteration, en ajoutant un paramètre $\lambda \in [0, 1]$. Dans cet article, nous étudions la version exacte de cet algorithme, même si c'est surtout dans un contexte d'approximation qu'il révèle son potentiel. En nous inspirant de Modified Policy Iteration, nous proposons une méthode plus générale, intitulée Modified λ-Policy Iteration, qui unifie ces quatre algorithmes, et nous étudions ses propriétés de convergence. Nous démontrons que ce nouvel algorithme converge vers la fonction de valeur optimale et nous établissons des bornes sur sa vitesse de convergence. Ces bornes suggèrent qu'asymptotiquement, il est préférable d'utiliser $\lambda = 1$. Enfin, une étude expérimentale met en évidence, sur une application classique de type navigation sur une grille 2D, qu'avec cette version modifiée de λ-Policy Iteration, la convergence est en effet la plus rapide lorsque $\lambda = 1$.

1 Présentation de λ-Policy Iteration

Dans cette première partie, après avoir introduit les notations et rappelé le contexte du contrôle optimal, nous présentons l'algorithme λ-Policy Iteration tel qu'il a été proposé par Bertsekas & Tsitsiklis (1996) en version exacte.

1.1 Notations utilisées

Le cadre des Processus Décisionnels de Markov permet de formaliser le contrôle optimal stochastique, qui considère un agent informatique devant prendre des décisions afin de maximiser un signal de récompense sur le long terme. Un Processus Décisionnel de Markov (PDM) est un quadruplet (S, A, T, R) où :

- S est l'espace d'états ;
- A est l'espace d'actions ;
- T est la fonction de transition : $T(s, a, s')$ est la probabilité d'arriver dans l'état s' sachant que l'on est dans l'état s et que l'on effectue l'action a.
- R est la fonction de récompense : $R(s, a) \in \mathbb{R}$ est la récompense reçue en effectuant l'action $a \in A$ depuis l'état $s \in S$.

Une *politique* est une fonction $\pi : S \rightarrow A$ qui associe à chaque état l'action correspondante : $\pi(s_t) = a_t$. La *fonction de valeur* d'une politique π est la fonction $V^\pi : S \rightarrow \mathbb{R}$ qui associe à chaque état l'espérance du cumul des récompenses que l'on peut obtenir à partir de cet état en suivant la politique π :

$$V^\pi(s) = E\left[\sum_{k=0}^{\infty} \gamma^k R(s_k, \pi_{s_k}) \middle| s_0 = s\right]$$

où $\gamma \in [0, 1]$ est un *facteur d'actualisation* permettant de diminuer d'importance aux récompenses lointaines. Une propriété fondamentale de la fonction de valeur d'une politique π est le fait qu'elle vérifie une équation récursive, *l'équation de Bellman* (Bellman, 1957) :

$$V^\pi(s) = R(s, \pi(s), s') + \gamma . \sum_{s'} T(s, \pi(s), s').V^\pi(s'). \tag{1}$$

Ainsi, la valeur d'un état dépend de la récompense immédiate et de la valeur des états suivants. Cette équation récursive est le fondement de nombreux alorithmes liés aux Processus Décisionnels de Markov. On peut la réécrire de manière condensée en introduisant l'opérateur de Bellman B_π :

$$[B_\pi V](s) = R(s, \pi(s)) + \gamma . \sum_{s'} T(s, \pi(s), s').V(s'). \tag{2}$$

Cet opérateur est contractant (Puterman, 1994) et son unique point fixe est la fonction de valeur V^π.

Dans le cadre des PDM, l'objectif est de déterminer une politique optimale. On note V^* la fonction de valeur optimale, qui associe à chaque état la meilleure espérance possible des récompenses.

$$V^*(s) = \max_\pi V^\pi(s).$$

Il peut exister plusieurs politiques optimales, qui partagent alors cette fonction de valeur. Si l'on connaît la fonction de valeur optimale, alors on en déduit facilement une politique optimale π^* en sélectionnant la politique *gourmande* sur les valeurs de V^* :

$$\pi^*(s) = \arg\max_a \left(R(s, a) + \gamma . \sum_{s'} T(s, a, s').V^*(s') \right).$$

La fonction de valeur optimale vérifie elle aussi une équation récursive, *l'équation d'optimalité de Bellman* (Bellman, 1957) :

$$V^*(s) = \max_a \left(R(s, a) + \gamma . \sum_{s'} T(s, a, s').V^*(s') \right). \tag{3}$$

Là aussi, on introduit un opérateur, noté B :

$$[BV](s) = \max_a \left(R(s, a) + \gamma . \sum_{s'} T(s, a, s').V(s') \right) \tag{4}$$

pour avoir une notation condensée de l'équation (3) :

$$BV^* = V^*.$$

L'opérateur B est contractant (Puterman, 1994) et son unique point fixe est la fonction de valeur optimale V^*. V^* est donc la seule fonction de valeur vérifiant $BV^* = V^*$. Cet opérateur permet notamment d'exprimer le fait qu'une politique π soit gourmande par rapport à une fonction de valeur V : on écrit alors $BV = B_\pi V$.

Nous présentons maintenant quelques algorithmes qui permettent de calculer la fonction de valeur optimale, en particulier λ-Policy Iteration que nous étudions dans la suite de cet article. Ces algorithmes sont présentés d'une manière qui permettra plus clairement de les unifier dans la suite de l'article.

1.2 Value Iteration

L'algorithme Value Iteration (Bellman, 1957), issu de la programmation dynamique, est l'un des algorithmes de base des Processus Décisionnels de Markov.

Algorithme 1 (Value Iteration)
$k \leftarrow 0, V_0$ *arbitraire*
Répéter :
π_{k+1} *choisie telle que* $B_{\pi_{k+1}} V_k = BV_k$
$V_{k+1} \leftarrow B_{\pi_{k+1}} V_k$
$k \leftarrow k+1$
Jusqu'à $\|V_{k+1} - V_k\|_\infty < \epsilon$

A chaque itération, la politique π_{k+1} est choisie comme la politique gourmande sur les valeurs de V_k. Puis la valeur suivante V_{k+1} est calculée en appliquant une fois l'opérateur de Bellman sur la valeur courante V_k. Comme $B_{\pi_{k+1}} V_k = BV_k$, chaque itération revient en fait à appliquer l'opérateur de Bellman B présenté plus haut. Comme cet opérateur est contractant et que son unique point fixe est la fonction de valeur optimale V^*, l'algorithme converge vers la valeur optimale.

1.3 Policy Iteration

Avec l'algorithme Policy Iteration (Bellman, 1957), la politique π_{k+1} est choisie comme la politique gourmande sur les valeurs de V_k, puis V_{k+1} est calculée comme la valeur de la politique π_{k+1}. Pour cela, plutôt que de résoudre l'équation de Bellman (1) analytiquement (ce qui reviendrait de résoudre un système de $|S|$ équations à $|S|$ inconnues), on peut appliquer successivement l'opérateur $B_{\pi_{k+1}}$ jusqu'à atteindre son point fixe qui est la valeur de la politique π_{k+1}.

Algorithme 2 (Policy Iteration)
$k \leftarrow 0, V_0$ *arbitraire*
Répéter :
π_{k+1} *choisie telle que* $B_{\pi_{k+1}} V_k = BV_k$
$V_{k+1} \leftarrow \lim_{m \to +\infty} B^m_{\pi_{k+1}} V_k$
$k \leftarrow k+1$
Jusqu'à $\pi_{k+1} = \pi_k$

Policy Iteration nécessite en général un plus petit nombre d'itérations que Value Iteration pour converger (Bertsekas & Tsitsiklis, 1996), grâce à la phase d'évaluation de la politique. En contrepartie, cette phase d'évaluation peut s'avérer coûteuse lorsque le nombre d'états est élevé puisqu'il faut appliquer un grand nombre de fois l'opérateur de Bellman à chaque itération. L'algorithme le mieux adapté depend de la structure du problème.

1.4 Modified Policy Iteration

Une alternative à Value Iteration et Policy Iteration consiste à appliquer l'opérateur de Bellman un nombre déterminé de fois m. Ainsi, on ne calcule pas entièrement la valeur de la politique courante π_k (contrairement à Policy Iteration), mais on peut s'approcher plus rapidement de la valeur optimale qu'avec Value Iteration. Cette méthode est intitulée Modified Policy Iteration (Bertsekas & Tsitsiklis, 1996).

Algorithme 3 (Modified Policy Iteration)
$k \leftarrow 0, V_0$ *arbitraire*
$m \in \mathbb{N}^*$
Répéter :
π_{k+1} *choisie telle que* $B_{\pi_{k+1}} V_k = BV_k$
$V_{k+1} \leftarrow B^m_{\pi_{k+1}} V_k$
$k \leftarrow k+1$
Jusqu'à $\|V_{k+1} - V_k\|_\infty < \epsilon$

Lorsque $m = 1$, on retrouve Value Iteration, et lorsque $m \to \infty$, on retrouve Policy Iteration.

1.5 λ-Policy Iteration

λ-Policy Iteration, introduit par Bertsekas & Tsitsiklis (1996), propose une autre manière de généraliser Value Iteration et Policy Iteration. Pour cela, un paramètre $\lambda \in [0, 1]$ spécifie si l'algorithme est plus proche de Policy Iteration ($\lambda = 1$) ou de Value Iteration ($\lambda = 0$).

Comme les algorithmes présentés plus haut, λ-Policy Iteration considère à chaque itération un couple (V_k, π_k). π_k est la politique courante et V_k peut être vu comme une approximation de la fonction de valeur V^{π_k}.

Algorithme 4 (λ-Policy Iteration)
$k \leftarrow 0$, *V_0 arbitraire*
$\lambda \in [0, 1]$
Répéter :
π_{k+1} *choisie telle que* $B_{\pi_{k+1}} V_k = BV_k$
$V_{k+1} \leftarrow \lim_{m \to +\infty} (1 - \lambda) \left(\sum_{i=1}^{m} \lambda^{i-1} B^i_{\pi_{k+1}} V_k \right) + \lambda^m B^m_{\pi_{k+1}} V_k$
$k \leftarrow k + 1$
Jusqu'à $\|V_{k+1} - V_k\|_\infty < \epsilon$

Comme dans les algorithmes précédents, la nouvelle politique π_{k+1} est choisie comme la politique gourmande sur V_k. La mise à jour de la fonction de valeur V_{k+1} correspond, lorsque $\lambda < 1$, à une sorte de moyenne pondérée (par les λ^i) des termes identiques à ceux de Modified Policy Iteration : $B^i_{\pi_{k+1}} V_k$. Lorsque $\lambda = 1$, le premier terme de la limite s'annule et on retrouve bien Policy Iteration.

L'intérêt de l'algorithme λ-Policy Iteration est que la phase d'évaluation de la politique π_{k+1} peut être plus facile lorsque $\lambda < 1$ que lorsque $\lambda = 1$ (ce qui correspond à Policy Iteration). En contrepartie, plus λ est petit, plus la séquence des V_k met du temps à converger vers la valeur optimale V^*.

Bertsekas & Tsitsiklis (1996) ont montré que λ-Policy Iteration converge vers un couple valeur-politique optimal pour toutes les valeurs de λ et ont fourni une vitesse de convergence asymptotique. Ces résultats sont donnés dans la proposition 1.

Proposition 1 (Convergence de λ-Policy Iteration (Bertsekas & Tsitsiklis, 1996))
Soit (V_k, π_k) la séquence de fonctions de valeurs et de politiques générées par λ-Policy Iteration. On a alors :

$$\lim_{k \to +\infty} V_k = V^*.$$

De plus, pour tout k plus grand qu'un certain index $\bar{k}$,

$$\|V^* - V_{k+1}\| \leq \frac{\gamma(1 - \lambda)}{1 - \lambda\gamma} \|V^* - V_k\|$$

$\| \cdot \|$ désigne la norme infinie sur l'espace des fonctions de valeur, c'est-à-dire $\|V\| = \max_s |V(s)|$. La modification à que nous proposons à λ-Policy Iteration dans la suite de cet article généralise ces résultats.

Implantation de λ-Policy Iteration

Bertsekas & Tsitsiklis (1996) ont introduit un opérateur noté M_k permettant d'obtenir une autre écriture de l'algorithme, et défini de la manière suivante :

Définition 1
Soit V_k la fonction de valeur à l'itération k de l'algorithme λ-Policy Iteration. On note M_k l'opérateur défini par :

$$M_k V = (1 - \lambda) B_{\pi_{k+1}} V_k + \lambda B_{\pi_{k+1}} V$$

Il a été établi (Bertsekas & Tsitsiklis, 1996) que l'opérateur M_k est contractant de facteur $\gamma\lambda$ et que son unique point fixe est la fonction de valeur V_{k+1} correspondant à l'itération suivante de λ-Policy Iteration. On a en outre :

$$M^m_k V = (1 - \lambda) \left(\sum_{i=1}^{m} \lambda^{i-1} B^i_{\pi_{k+1}} V_k \right) + \lambda^m B^m_{\pi_{k+1}} V \tag{5}$$

Ainsi, pour calculer V_{k+1}, il suffit d'effectuer des applications successives de l'opérateur M_k jusqu'à obtenir le point fixe V_{k+1}.

2 Modified λ-Policy Iteration

Nous venons de présenter l'algorithme λ-Policy Iteration (Bertsekas & Tsitsiklis, 1996), un algorithme qui généralise Policy Iteration ($\lambda = 1$) et Value Iteration ($\lambda = 0$), et d'en donner une écriture équivalente qui permet de l'implanter en utilisant une méthode itérative à l'aide de l'opérateur M_k.

En fait, au lieu de calculer le point fixe V_{k+1} de manière exacte en appliquant l'opérateur M_k un nombre de fois théoriquement infini, il est possible de se contenter d'appliquer l'opérateur M_k un nombre limité de fois. Cette modification que nous proposons à l'algorithme permet donc de rendre la phase d'évaluation de la politique plus souple, en s'arrêtant sans attendre d'avoir convergé précisément vers le point fixe de l'opérateur M_k. Nous appelons cette méthode *Modified λ-Policy Iteration*, de manière analogue à Modified Policy Iteration qui repose sur la même idée : évaluer une politique de manière incomplète, en s'arrêtant après un nombre limité d'étapes.

Algorithme 5 (Modified λ-Policy Iteration)
$k \leftarrow 0$, *V_0 arbitraire*
$\lambda \in [0, 1], m \in \mathbb{N}^*$
Répéter :
π_{k+1} *choisie telle que* $B_{\pi_{k+1}} V_k = BV_k$
$V_{k+1} \leftarrow (1-\lambda) \left(\sum_{i=1}^{m} \lambda^{i-1} B^i_{\pi_{k+1}} V_k \right) + \lambda^m B^m_{\pi_{k+1}} V_k$
$k \leftarrow k + 1$
Jusqu'à $\|V_{k+1} - V_k\|_\infty < \epsilon$

On voit, d'après l'équation (5), que la mise à jour se résume à appliquer m fois l'opérateur M_k : $V_{k+1} \leftarrow M_k^m V_k$. En observant la règle de mise à jour de V_{k+1}, on remarque ainsi que l'algorithme Modified λ-Policy Iteration unifie les quatre algorithmes du contrôle optimal présentés plus haut :

- si $m \to \infty$, alors on obtient l'algorithme λ-Policy Iteration ;
- si $\lambda = 1$, alors on obtient l'algorithme Modified Policy Iteration ;
- si les deux conditions précédentes sont réunies, alors on obtient l'algorithme Policy Iteration ;
- si $m = 1$ ou si $\lambda = 0$, alors on obtient l'algorithme Value Iteration.

Ces généralisations sont récapitulées sur le schéma de la Figure 1.

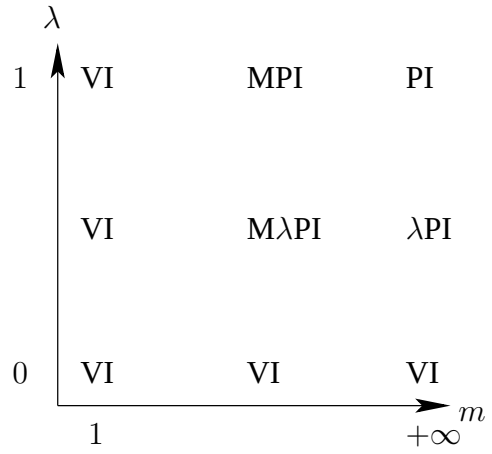

FIG. 1 – Selon la valeur des paramètres λ et m, Modified λ-Policy Iteration (MλPI) généralise les algorithmes λ-Policy Iteration (λPI), Modified Policy Iteration (MPI), Policy Iteration (PI) et Value Iteration (VI).

La proposition suivante établit que l'algorithme Modified λ-Policy Iteration converge vers un couple valeur-politique optimal et fournit des bornes de convergence. Elle généralise celle de Bertsekas & Tsitsiklis (1996) concernant λ-Policy Iteration (proposition 1), et y ajoute une vitesse de convergence non asymptotique.

Proposition 2 (Convergence de Modified λ-Policy Iteration)
Soit (V_k, π_k) la séquence de fonctions de valeurs et de politiques générées par Modified λ-Policy Iteration. On a alors :

$$\lim_{k \to +\infty} V_k = V^*.$$

De plus, pour tout k plus grand qu'un certain index $\bar{k}$,

$$\|V_{k+1} - V^*\| \leq \beta \|V_k - V^*\|, \text{ avec } \beta = \frac{\gamma(1-\lambda)(1-(\lambda\gamma)^m)}{1-\lambda\gamma} + (\lambda\gamma)^m \in [\gamma^m, \gamma].$$

Enfin, si V_0 est tel que $BV_0 \geq V_0$, alors on a pour tout k,

$$0 \leq V^* - V_{k+1} \leq \gamma(V^* - V_k).$$

Preuve Nous nous inspirons ici de la preuve page 46 dans Bertsekas & Tsitsiklis (1996) en adaptant quelques détails pour la spécificité de Modified λ-Policy Iteration par rapport à λ-Policy Iteration.

Supposons d'abord que $BV_0 \geq V_0$. Nous allons montrer par induction que pour tout k,

$$V^* \geq BV_{k+1} \geq V_{k+1} \geq BV_k \geq V_k. \tag{6}$$

D'après la définition de M_k (définition 1), et en utilisant le fait que $B_{\pi_{k+1}} V_k = BV_k$, on a

$$M_k V_k = B_{\pi_{k+1}} V_k = BV_k.$$

M_k est monotone (c'est-à-dire $V \leq V' \Rightarrow M_k V \leq M_k V'$) car $B_{\pi_{k+1}}$ est monotone. Donc pour tout $m_k \in \mathbb{N}^*$,

$$M_k^m V_k \geq M_k V_k = BV_k \geq V_k. \tag{7}$$

Or, $M_k^m V_k = V_{k+1}$. On a donc :

$$V_{k+1} \geq BV_k \geq V_k. \tag{8}$$

D'après la définition de M_k (définition 1), on a

$$\begin{aligned} M_k V_{k+1} &= (1-\lambda) B_{\pi_{k+1}} V_k + \lambda B_{\pi_{k+1}} V_{k+1} \\ &= (1-\lambda) B_{\pi_{k+1}} V_k - (1-\lambda) B_{\pi_{k+1}} V_{k+1} + B_{\pi_{k+1}} V_{k+1} \\ &= B_{\pi_{k+1}} V_{k+1} + (1-\lambda)(B_{\pi_{k+1}} V_k - B_{\pi_{k+1}} V_{k+1}). \end{aligned}$$

Or, $B_{\pi_{k+1}} V_k - B_{\pi_{k+1}} V_{k+1} \leq 0$ car $V_{k+1} \geq V_k$ (d'après (8)) et l'opérateur $B_{\pi_{k+1}}$ est monotone. Par conséquent, $B_{\pi_{k+1}} V_{k+1} \geq M_k V_{k+1}$. En remplaçant V_{k+1} par sa définition (algorithme 5, et d'après (7), on obtient :

$$M_k V_{k+1} = M_k M_k^m V_k \geq M_k^m V_k = V_{k+1}.$$

Comme $BV_{k+1} \geq B_{\pi_{k+1}} V_{k+1}$, on obtient ainsi :

$$BV_{k+1} \geq V_{k+1}. \tag{9}$$

Comme l'opérateur de Bellman B est monotone, on a pour tout $m \in \mathbb{N}^*$, $B^m V_{k+1} \geq BV_{k+1}$. En prenant la limite quand $m \to +\infty$, on obtient :

$$V^* \geq BV_{k+1}. \tag{10}$$

Si V_0 est tel que $BV_0 \geq V_0$, alors en assemblant les inégalités (8), (9) et (10), on obtient enfin l'inégalité à démontrer (6). Lorsque $k \to +\infty$, la séquence des V_k converge donc vers une limite que l'on note V_∞ et qui vérifie $V^* \geq V_\infty$. Comme $V_{k+1} - V_k \to 0$, on obtient en remplaçant dans l'inégalité (6) :

$$V_\infty \geq BV_\infty \geq V_\infty.$$

On a donc $V_\infty = BV_\infty$, ce qui signifie que V_∞ vérifie l'équation de Bellman. Ainsi, $V_\infty = V^*$.

Pour montrer la vitesse de convergence non asymptotique, nous utilisons le fait que, lorsque $BV_0 \geq V_0$, on a $M_k^m V_k \geq M_k V_k = BV_k$:

$$0 \leq V^* - V_{k+1} = BV^* - M_k^m V_k \leq BV^* - BV_k \leq \gamma(V^* - V_k).$$

Nous nous plaçons maintenant dans le cas où l'hypothèse $BV_0 \geq V_0$ n'est pas forcément vérifiée, et nous allons montrer le résultat de vitesse convergence asymptotique ainsi que la convergence vers V^* sans cette supposition. Pour la vitesse de convergence asymptotique, considérons l'index $\bar{k}$ tel que pour tout $k \geq \bar{k}$, π_{k+1} est une politique optimale si bien que $B_{\pi_{k+1}} V^* = BV^* = V^*$. Alors, en utilisant le fait que l'opérateur B_{π_k+1} est contractant de facteur γ, on a pour tout $k \geq \bar{k}$,

$$\begin{aligned}
\|V_{k+1} - V^*\| &= \left\| (1-\lambda) \left[\sum_{i=0}^{m-1} \lambda^i (B_{\pi_k+1})^{i+1} V_k \right] + \lambda^m (B_{\pi_k+1})^m V_k - V^* \right\| \\
&\leq (1-\lambda) \sum_{i=0}^{m-1} \lambda^i \gamma^{i+1} \|V_k - V^*\| + (\lambda\gamma)^m \|V_k - V^*\| \\
&= \beta \|V_k - V^*\|
\end{aligned}$$

avec

$$\beta = (1-\lambda) \sum_{i=0}^{m-1} \lambda^i \gamma^{i+1} + (\lambda\gamma)^m = \frac{\gamma(1-\lambda)(1-(\lambda\gamma)^m)}{1-\lambda\gamma} + (\lambda\gamma)^m.$$

On voit ci-dessus que β est la moyenne de termes $\gamma, \gamma^2, \ldots, \gamma^m$ (avec les poids $(1-\lambda), (1-\lambda)\lambda, (1-\lambda)\lambda^2, \ldots, (1-\lambda)\lambda^{m-2}, (1-\lambda)\lambda^{m-1} + \lambda^m$ dont la somme est 1) donc on sait que β appartient à l'intervalle $[\gamma^m, \gamma]$.

Enfin, pour montrer le résultat $V_k \to V^*$ dans le cas où l'on n'a pas $BV_0 \geq V_0$, on peut remplacer V_0 par un vecteur $\hat{V_0} = V_0 + ce$, où $e = (1, \ldots, 1)$ et c est une constante réelle positive suffisamment grande pour que $B\hat{V_0} \geq \hat{V_0}$; en effet, on peut voir que lorsque $c \geq \frac{1}{1-\gamma} \max_s (V_0(s) - BV_0(s))$, on a $ce \geq \frac{1}{1-\gamma}(V_0 - BV_0)$ et donc $BV_0 - \gamma ce \geq V_0 - ce$, ce qui équivaut à $B\hat{V_0} \geq \hat{V_0}$. Considérons alors l'algorithme Modified λ-Policy Iteration initialisé avec $(\hat{V_0}, \hat{\pi_0})$ et notons $(\hat{V_k}, \hat{\pi_k})$ la séquence des valeurs et de politiques générées. Alors il peut être montré par induction que pour tout k, on a

$$\|\hat{V_k} - V_k\| = \beta^m c$$

Ainsi, $\hat{V_k} - V_k \to 0$. Comme nous avons montré que $\hat{V_k} \to V^*$, on a bien également $V_k \to V^*$. □

Une question naturelle qui se pose est le choix des paramètres λ et m. La vitesse de convergence asymptotique de la séquence des V_k, d'après la proposition 2, est la plus rapide lorsque $\lambda = 1$: elle est alors bornée par γ^m. Comme l'indiquent Bertsekas & Tsitsiklis (1996), elle se dégrade lorsque λ diminue. Lorsque $\lambda = 0$, elle n'est bornée que par γ. Dans la prochaine section, nos expériences montrent que lorsque le nombre d'étapes internes m est limité, il est en effet préférable d'utiliser $\lambda = 1$.

3 Expériences

Nous venons de voir qu'en théorie, plus λ est éloigné de 1, plus la séquence des V_k met de temps à converger vers la valeur optimale V^*. Lorsque la fonction de valeur est représentée de manière exacte, l'intérêt d'utiliser une valeur de λ inférieure à 1 est uniquement d'accélérer la phase de calcul de V_{k+1} à chaque itération. Or, lorsqu'on limite le nombre d'étapes internes m dans les itérations, c'est également ce qui se passe, et par conséquent, comme vont le confirmer nos expériences, il n'est plus significativement utile d'avoir $\lambda < 1$.

Nous avons mené des expériences sur un problème de type navigation discrète. Un agent se déplace sur une grille en deux dimensions et se dirige dans les quatre directions principales jusqu'à atteindre un objectif. Certaines cases de la grille sont des murs et les décisions de l'agent peuvent être bruitées. Plus formellement, le PDM est défini de la manière suivante :

- L'espace d'états S est l'ensemble des cases de la grille n'étant pas des murs, auquel on ajoute un état terminal indiquant que l'objectif a été atteint ;
- L'espace d'actions A est composé des cinq actions suivantes : Nord, Sud, Est, Ouest et l'action consistant à rester sur place ;
- Notons $\mu \in [0, 1]$ le terme de bruit du PDM. La fonction de transition est, avec probabilité $1 - \mu$, le déplacement correspondant l'action choisie, et avec probabilité μ, un déplacement aléatoire choisi uniformément parmi les 4 directions. Lorsque l'action choisie consiste à rester sur place, aucun bruit n'est appliqué. Si un déplacement mène à une case occupée par un mur, alors l'agent reste sur place.

– Enfin, la récompense est de -1 à chaque étape tant que l'état terminal n'est pas atteint, 0 une fois que l'état terminal est atteint, et une pénalité de -100 en cas de collision contre un mur.

Nous avons considéré des PDM ayant de multiples de valeurs de γ et de bruit, sur lesquels nous avons exécuté l'algorithme Modified λ-Policy Iteration avec différentes valeurs de λ et de m afin de rechercher des PDM pour lesquels la meilleure valeur de λ serait différente de 1. Pour comparer les exécutions en terme de rapidité, nous avons défini une mesure de performance destinée à compter le nombre d'opérations effectuées au cours de l'exécution d'un algorithme. Nous considérons qu'une application de l'opérateur de Bellman B_π correspond à une opération (cet opérateur nécessite de parcourir tous les états du PDM). Le calcul d'une politique gourmande équivaut à $|A|$ opérations (5 dans notre cas), car pour chaque action il faut parcourir tous les états. Enfin, calculer $V_{k+1} = M_k^m V_k$ nécessite $m+1$ opérations : une opération pour calculer le terme $(1-\lambda)B_{\pi_{k+1}}V_k$, qui ne change pas lorsqu'on applique M_k plusieurs fois de suite, et m opérations pour le second terme de M_k, qui demande une opération à chaque fois que M_k est appliqué.

La figure 2 représente pour un PDM donné, le nombre d'opérations qui ont été nécessaires pour exécuter Modified λ-Policy Iteration en fonction de différentes valeurs de λ et de m. Les paramètres pour lesquels le nombre d'opération est minimal (387 opérations) sont $\lambda = 1$ et $m = 32$. Cette courbe est typique des expériences que nous avons lancées : en effet, sur toutes nos expériences, il apparaît que le nombre d'opérations est le plus faible lorsque m est limité (typiquement, $m < 100$) et $\lambda = 1$.

Pour certains PDM cependant, la valeur optimale de λ qui a été trouvée s'est avérée être légèrement inférieure à 1 (entre $0,97$ et 1 selon les cas). La figure 3 représente des résultats plus précis pour l'un de ces PDM. Ses propriétés sont $\gamma = 0,998$ et un bruit est de $0,1$. Les meilleurs paramètres trouvés étaient $\lambda = 0,99$ et $m = 4$. Nous avons relancé des expériences sur ce PDM avec des valeurs plus fines de λ et m afin d'obtenir une courbe plus précise autour de ces valeurs (voir figure 3). On remarque en fait que les résultats sont assez bruités. Cela pourrait être dû à une part de hasard qui fait qu'à cause de la structure du PDM, la séquence des politiques π_k peut aboutir à une politique optimale parfois plus tôt que d'autres. Dans ces conditions, ces résultats ne permettent donc pas de conclure qu'une valeur de λ différente de 1 puisse être significativement meilleure que $\lambda = 1$ en terme de vitesse de convergence lorsque m est fixé à une valeur donnant de bonnes performances.

L'intérêt d'utiliser $\lambda < 1$ est d'accélérer chaque itération en évaluant de manière incomplète la politique courante π_{k+1}. En contrepartie, le nombre total d'itérations augmente puisque cette évaluation n'est plus exacte. Or, dès lors que l'on fait diminuer m, c'est également ce qui se passe : on limite le nombre d'applications de l'opérateur M_k. Il apparaît donc que dans ce cas, il n'est plus utile d'avoir $\lambda < 1$ car cela ne fait que rendre plus approximative l'estimation de $V^{\pi_{k+1}}$. Utiliser $\lambda = 1$ ne pose alors plus de problème étant donné que les itérations sont déjà limitées par le nombre d'opérations m.

Conclusion et perspectives

Notre étude approfondit la version exacte de la méthode λ-Policy Iteration introduite par Bertsekas & Tsitsiklis (1996). Elle propose une modification de cet algorithme, analogue à Modified Policy Iteration. Cette modification généralise les algorithmes λ-Policy Iteration, Modified Policy Iteration, Value Iteration et Policy Iteration. Nous avons montré que ce nouvel algorithme plus général converge vers la fonction de valeur optimale et nous avons fourni des bornes sur sa vitesse de convergence. Ces bornes confirment que la vitesse de convergence asymptotique se déteriore lorsque λ diminue. Une mise en application montre que la vitesse de convergence est plus rapide lorsque m est limité, et que dans ces conditions, il est en effet préférable d'utiliser $\lambda = 1$.

Le paramètre λ révèle surtout son utilité dans un cadre d'approximation de la fonction de valeur et de simulation. Bertsekas & Tsitsiklis (1996) ont proposé une version approximative de l'algorithme λ-Policy Iteration, qui s'applique aux problèmes où le nombre d'états est élevé et où il est nécessaire de recourir à des techniques d'approximation. La fonction de valeur est alors approximée par une architecture linéaire et les états sont visités en effectuant des simulations. Dans ce contexte, le fait que la vitesse de convergence asymptotique se dégrade lorsque $\lambda < 1$ est moins crucial car la fonction de valeur optimale ne peut de toute manière pas être atteinte de manière exacte. De plus, lorsqu'on évalue la politique courante en effectuant des simulations, la variance des termes qui doivent être moyennés pour estimer la fonction de valeur est plus faible lorsque $\lambda < 1$ que lorsque $\lambda = 1$. Le nombre de simulations à effectuer s'en trouve donc réduit.

Le prolongement naturel de notre étude consiste à étendre les résultats que nous obtenus au contrôle optimal approché. Nous examinons actuellement plusieurs pistes qui pourraient permette d'améliorer ou

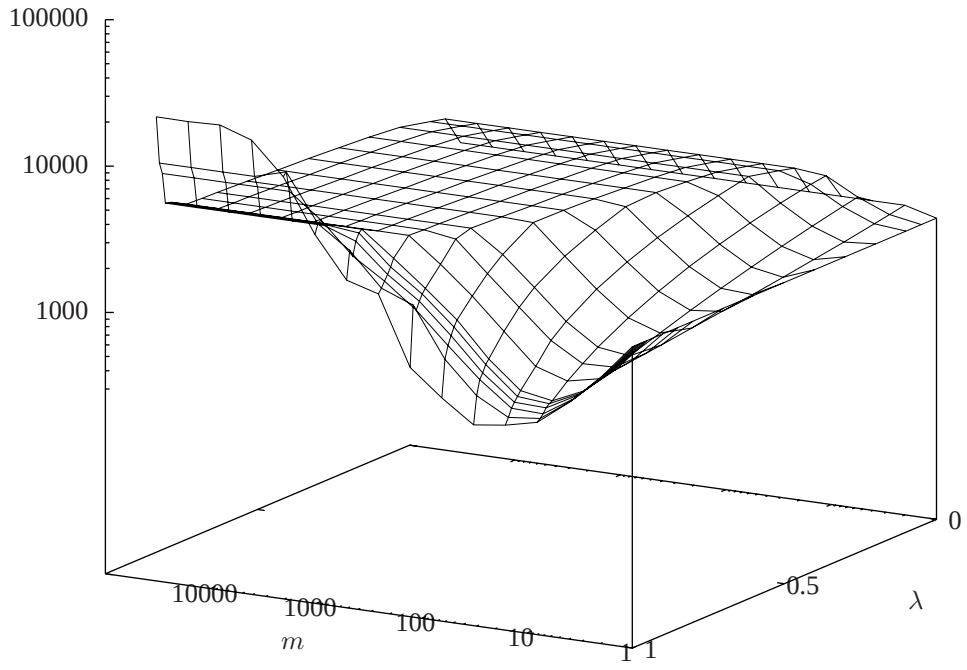

FIG. 2 – Nombre d'opérations effectuées par l'algorithme Modified λ-Policy Iteration pour converger en fonction des paramètres λ et m, sur un problème dont $\gamma = 0,999$ et le bruit est de $0,4$. Le minimum est atteint pour $\lambda = 1$ et $m = 32$, où 387 opérations ont été effectuées. Lorsque le nombre d'étapes m est en-dessous d'une certaine valeur, il n'est plus utile d'utiliser une valeur de λ autre que 1.

de généraliser la version approximative de λ-Policy Iteration. Une possibilité est de proposer une version approximative de Modified λ-Policy Iteration où le principe serait d'effectuer des simulations incomplètes pour estimer la fonction de valeur. D'autres pistes consistent à mettre au point des techniques permettant de fixer automatiquement la valeur de λ en calculant le meilleur compromis entre la vitesse de convergence asymptotique et la variance des échantillons générés par la simulation.

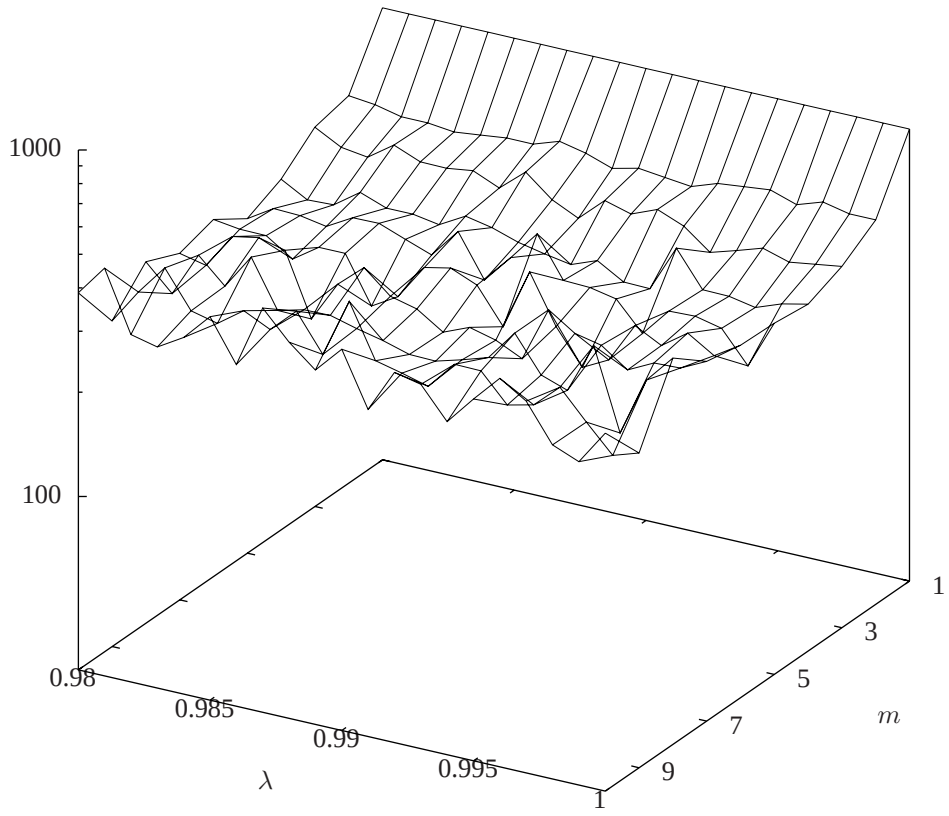

FIG. 3 – Nombre d'opérations effectuées par l'algorithme Modified λ-Policy Iteration en fonction d'un champ restreint de paramètres λ et m ($\lambda \in [0,98, 1]$ et $m \in [1, 10]$), avec $\gamma = 0,998$ et un bruit de $0,1$. Bien que le minimum soit atteint pour $\lambda = 0,994$ et $m = 6$ (avec 223 opérations effectuées), les résultats semblent bruités et ne permettent pas de conclure qu'une valeur de λ différente de 1 soit significativement meilleure.

Références

BELLMAN R. E. (1957). *Dynamic Programming*. Princeton, NJ : Princeton University Press.

BERTSEKAS D. & TSITSIKLIS J. (1996). *Neurodynamic Programming*. Athena Scientific.

PUTERMAN M. (1994). *Markov Decision Processes*. Wiley, New York.

SUTTON R. & BARTO A. (1998). *Reinforcement Learning, An introduction*. BradFord Book. The MIT Press.