Apprentissage par renforcement
Optimisation en univers incertain

Richard Lassaigne

Institut de Mathématiques de Jussieu-Paris Rive Gauche
Equipe de Logique mathématique
CNRS-Université Paris Diderot

Cet article est une introduction à certaines méthodes algorithmiques d'apprentissage par renforcement qui sont souvent analogues à celles utilisées en pratique pour les problèmes d'optimisation en univers incertain de grande dimension. Le domaine général de l'apprentissage automatique concerne différents domaines d'application comme la reconnaissance d'images, de textes, la traduction automatique, la classification automatique, mais aussi le déplacement des robots ou l'automatisation des coups dans un jeu. L'une des directions de recherche récentes est celle de l'apprentissage supervisé : à partir de quantités massives de données correctement traitées, exemples et contre-exemples, il s'agit d'élaborer des règles d'étiquetage pour des données nouvelles. Les techniques d'apprentissage profond basées sur les réseaux de neurones convolutifs ont permis d'obtenir récemment des résultats importants.

L'apprentissage par renforcement s'intéresse à une autre direction de recherche : il s'agit d'*interaction avec l'environnement* par une technique d'essais erreurs guidée par des récompenses. Ce concept d'apprentissage séquentiel, où les données sont traitées à la volée, a été modélisé en premier par Thompson, dans les années 30, pour traiter des essais cliniques. Le modèle le plus simple est celui des bandits manchots stochastiques dans un casino : il s'agit de choisir à chaque étape une action, en l'occurrence une machine, parmi différentes actions possibles ; à chaque *action* est associée une *récompense*, ou un paiement, suivant une distribution de probabilités dépendant de l'action choisie. Il existe potentiellement une machine dont le rendement est meilleur que les autres. L'objectif est de construire une *stratégie* pour jouer successivement sur telle ou telle machine, dans

le but de trouver la meilleure. La mesure de performance est appelée le *regret* : c'est la différence entre le paiement cumulé obtenu à l'aide d'une machine optimale et le paiement cumulé effectivement obtenu. Le but est de minimiser le regret, mais la difficulté est de réaliser un compromis entre acquisition et utilisation de l'information. Pour optimiser ce compromis entre exploration et exploitation, les stratégies classiques reposent sur l'idée de *renforcement*. Cependant l'analyse est délicate : trouver la proportion optimale de temps à passer sur les machines sous-optimales par rapport à l'exploration.

Ce modèle d'apprentissage séquentiel par interaction avec l'environnement a pour objectif la prédiction et l'amélioration du comportement du preneur de décision. Le problème général de la prise de décision en univers incertain peut se modéliser dans un cadre similaire à celui des problèmes d'optimisation, celui des processus de décision markoviens. Les objectifs sont différents : *prédiction* dans le premier cas et *optimisation* dans le second ; mais les fonctions d'évaluation des stratégies sont les mêmes. Le problème algorithmique clé est celui de la grande dimension : l'explosion combinatoire est due à l'étape de modélisation. Pour des raisons de complexité, les algorithmes classiques sont inutilisables en pratique. Afin de casser cette barrière de la complexité, il est nécessaire de faire appel à d'autres classes d'algorithmes : méthodes d'approximation, algorithmes probabilistes... Il est alors intéressant de constater que l'apprentissage par renforcement et les méthodes d'optimisation en univers incertain font souvent appel aux mêmes classes d'algorithmes.

La première partie de l'article permet de définir l'apprentissage par renforcement par ses caractéristiques, ses objectifs et les principaux problèmes à résoudre. La deuxième partie étudie le modèle des bandits à plusieurs bras dans le cas *stochastique* et dans le cas *adversaire*. Dans le premier cas, le principe d'optimisme face à l'incertain est mis en oeuvre par l'un des principaux algorithmes : UCB pour Upper Confidence Bound. Pour le cas adversaire, c'est la méthode des poids multiplicatifs, présentée dans un cadre de jeux, qui est utilisée par l'algorithme EXP3. Le modèle général de l'optimisation en univers incertain, celui des processus de décision markoviens, est présenté dans la troisième partie, ainsi que les deux grandes catégories d'algorithmes classiques : itération sur la *fonction valeur*, itération sur la *stratégie*. Malheureusement, ces méthodes, qui sont polynomiales dans la taille du modèle, sont souvent impraticables : en général, la phase de modélisation fait exploser la taille du modèle. La dernière partie s'intéresse brièvement aux méthodes de recherche dans les arbres qui sont utilisées dans ce cadre et dans celui de l'apprentissage par renforcement. Les algorithmes efficaces relèvent des méthodes d'*échantillonnage*, éventuellement adaptatif, et de *simulation* du type Monte-Carlo.

1 Apprentissage par renforcement : objectifs et challenges

La référence de base est le livre de Sutton et Barto [13] :

« **Reinforcement learning** is learning what to do –how to map situations to actions– so as to maximize a numerical **reward** signal in an **unknown uncertain** environment. The learner is not to told which actions to take, as in most forms of machine learning, but must discover which actions yield the most **reward** by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the **next situation** and, through that, all **subsequent rewards**. These two characteristics –**trial-and-error** search and **delayed reward**– are the most important distinguishing features of reinforcement learning. »

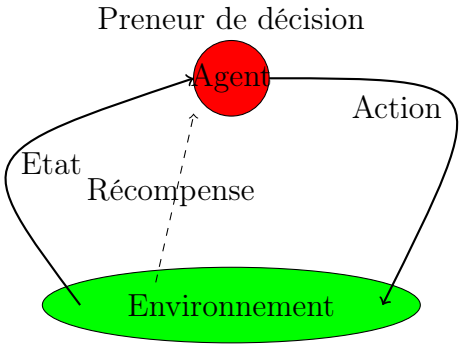

FIGURE 1 – Interaction entre un preneur de décision et l'environnement

L'environnement peut être stochastique, adversaire comme dans le cas des jeux, partiellement inconnu, partiellement observable. L'information disponible est le renforcement, calculé à l'aide de récompenses associées à chaque action. Les caractéristiques principales sont :

- la *stratégie* du preneur de décision qui définit son comportement ;
- le signal de *récompense* (numérique) envoyé par l'environnement, en réaction à une action choisie ; dans le cas général, ce signal peut être une fonction stochastique de l'état de l'environnement et de l'action choisie ;
- la *valeur* d'un état est la récompense totale que peut espérer accumuler le preneur de décision dans le futur à partir de cet état ;
- la dernière caractéristique, pas forcément présente, est un *modèle* de l'environnement.

La composante la plus importante d'un algorithme d'apprentissage est une méthode d'évaluation efficace des états, appelée *fonction d'évaluation* ou fonction valeur. Les méthodes utilisant un modèle de l'environnement sont appelées basées sur un modèle, par opposition aux méthodes sans modèle, plus simples, qui procèdent explicitement par essais-erreurs.

Les méthodes de prise de décision en univers incertain peuvent être classifiées suivant deux types de paramètres :

- elles peuvent utiliser un modèle donné ou *apprendre* le modèle par interaction avec l'environnement ;
- les actions du preneur de décision peuvent ou non *modifier* l'environnement.

Apprendre le modèle	Problèmes de bandits à plusieurs bras	Apprentissage par renforcement
Modèle stochastique donné	Théorie de la la décision	Processus de décision markoviens
	Les actions ne changent pas le monde	Les actions changent le monde

Dans la suite, on va d'abord étudier le modèle des bandits à plusieurs bras, puis le cadre général des processus de décision markoviens. La dernière partie présentera une classe de méthodes efficaces pour l'apprentissage par renforcement, élaborées à partir d'idées issues des deux domaines précédents. Afin de bien situer l'apprentissage par renforcement, il est utile de bien identifier les trois principaux problèmes posés par ce type d'apprentissage. D'abord, les *dynamiques* des états et de récompenses sont *inconnues*. Les deux approches suivantes sont alors possibles :

- l'approche sans modèle consiste à *échantillonner* pour apprendre directement une fonction d'évaluation des états ;
- dans l'approche basée sur un modèle, on apprend d'abord un modèle, puis on utilise la *programmation dynamique* pour itérer sur la fonction valeur ou sur la stratégie.

Le second problème est celui de la grande dimension : pour des modèles de grande taille, la complexité du calcul peut être prohibitive, surtout en espace. Il est alors nécessaire d'utiliser des méthodes d'*approximation* de la fonction valeur ou de la stratégie. Le dernier problème est celui de l'exploration : comment explorer l'espace des états pour construire une bonne représentation de la fonction d'évaluation inconnue ?

Dans le cas de l'apprentissage sans modèle, l'étude sera restreinte aux problèmes des bandits stochastiques ou adversaires. Pour les autres types de méthodes, comme celle des différences temporelles ou le Q-learning, on pourra se référer à [13].

2 Le problème du bandit à plusieurs bras

2.1 Le bandit stochastique à plusieurs bras

Le principal type d'application possible est celui de l'allocation séquentielle de ressources comme dans les exemples d'essais cliniques, de publicité en ligne ou de systèmes de recommandations. Le contexte du bandit stochastique à plusieurs bras est défini par :

- les paramètres connus sont le nombre K de bras (machines) et le nombre n d'étapes ;
- les machines attribuent des récompenses en réponse aux actions suivant des distributions de probabilités $\nu_k (1 \leq k \leq K)$ inconnues à support dans $[0, 1]$;
- à chaque étape t, on choisit une machine k_t et on reçoit une récompense r_t suivant la distribution ν_{k_t}.

L'objectif est de déterminer une stratégie de sélection des bras permettant de maximiser la somme espérée des récompenses. Le problème réside dans le compromis entre *exploration* de nouveaux bras et *exploitation* des meilleurs bras déjà utilisés.

La mesure de performance d'une stratégie donnée est le *regret* : c'est la différence entre la meilleure valeur espérée $n\mu^*$, où $\mu^* = max_k \mu_t$ est l'espérance d'un bras optimal, et l'espérance de la somme des récompenses associées aux bras utilisés. Le regret cumulé espéré à l'étape n est :

$$R_n = n\mu^* - \sum_{t=1}^{n} \mathbb{E}[\mu_{k_t}]$$

L'objectif est donc de déterminer une stratégie permettant de minimiser le regret.

L'algorithme UCB, pour Upper Confidence Bound, est dû à P. Auer, N. Cesa-Bianchi et P. Fisher dans un article publié dans Machine Learning en 2002 [1]. Cet algorithme repose sur un principe d'*optimisme* face à l'incertain :

- on choisit le bras qui serait le meilleur si les valeurs des bras étaient les meilleures possibles, en forte probabilité, sachant ce qui a déjà été observé ;
- on attribue à chaque machine k un indice somme de la moyenne des récompenses reçues et d'une borne supérieure de la récompense espérée μ_k en forte probabilité ;
- on sélectionne la machine avec le plus grand indice.

Algorithme UCB :

- sélectionner chaque machine une fois
- pour chaque étape $t = 1, 2, \ldots, n$:

  sélectionner la machine k qui maximise $\hat{\mu}_{k,s} + \sqrt{\frac{2log(t)}{s}}$

  où $\hat{\mu}_{k,s}$ est la récompense moyenne obtenue avec le bras k

  et $s = n_k$ est le nb de fois où la machine k a été sélectionnée

On peut remarquer que le terme $\hat{\mu}_{k,s}$ correspond à l'exploitation et le terme $\sqrt{\frac{2log(t)}{s}}$ à l'exploration.

2.2 Analyse de l'algorithme UCB

Le regret cumulé espéré à l'étape n peut être reformulé de la manière suivante :

$$R_n = (\sum_{k=1}^{K} \mathbb{E}[n_k])\mu^* - \mathbb{E}[\sum_{k=1}^{K} n_k \mu_k] = \sum_{k=1}^{K} \Delta_k \mathbb{E}[n_k]$$

où $\Delta_k = \mu^* - \mu_k$ et n_k est le nombre de fois où le bras k à été sélectionné jusqu'à l'étape n. Le principe d'optimisme dans l'incertain est exprimé sous la forme de l'hypothèse (H) : à l'instant t, les moyennes empiriques des bras sont dans leurs intervalles de confiance respectifs, c'est-à-dire

$$\mu_k - \sqrt{\frac{2log(t)}{s}} \leq \hat{\mu}_{k,s} \leq \mu_k + \sqrt{\frac{2log(t)}{s}}$$

On peut montrer que si le bras k est choisi à l'étape t, alors $\mu_k + 2\sqrt{\frac{2log(t)}{s}} \geq \mu^*$ c'est-à-dire $s \leq \frac{8log(t)}{\Delta_k^2}$. D'après l'inégalité de Chernoff-Hoeffding [6], l'hypothèse (H) est satisfaite avec probabilité $\geq 1 - \frac{2}{t^4}$ et le nombre de cas où $s > \frac{8log(t)}{\Delta_k^2}$ peut être borné par une constante $C < 4.3$. La conséquence est que chaque bras k sous-optimal est visité en moyenne, au plus $\mathbb{E}[n_k] \leq \frac{8log(t)}{\Delta_k^2} + C$. Le regret cumulé espéré peut alors être borné par

$$R_n = \sum_{k=1}^{K} \Delta_k \mathbb{E}[n_k] \leq 8\Big(\sum_{k:\Delta_k>0} \frac{log(n)}{\Delta_k}\Big) + C\Big(\sum_{k:\Delta_k>0} \Delta_k\Big)$$

Le regret est donc *logarithmique* dans le nombre d'étapes :

$$R_n = O\Big(\sum_{k:\Delta_k>0} \frac{1}{\Delta_k} log(n)\Big)$$

Il est aussi possible, mais plus difficile, d'obtenir une borne indépendante des distributions :

$$R_n = O(\sqrt{Knlog(n)})$$

2.3 Le bandit adversaire à plusieurs bras

Le cadre général est analogue à celui de la théorie des jeux : à chaque étape t,

- l'adversaire affecte une récompense $r_{k,t} \in [0,1]$ à chaque bras, sans la révéler ;
- le joueur choisit un bras k et reçoit la récompense r_{k_t}, en n'observant que celle-ci.

L'objectif du joueur est de maximiser la somme des récompenses. Le regret à l'étape n est défini par :

$$R_n = max_{1 \leq k \leq K}(\sum_{t=1}^{n} r_{k,t}) - \sum_{t=1}^{n} r_{k_t}$$

C'est une mesure de performance relativement à une meilleure stratégie fixée. La question importante est : peut-on espérer $lim_{n \to \infty} \sup_{recompenses} \mathbb{E}R_n/n = 0$? On peut montrer que si la stratégie du joueur est déterministe, alors l'espérance du regret peut être linéaire dans le nombre d'étapes n. Une solution pour le joueur consiste à utiliser des stratégies probabilistes.

L'algorithme EXP3, pour Explore-Exploit using Exponential Weights, a été conçu par P. Auer, N. Cesa-Bianchi, Y. Freund et R.E. Schapire en 2002 [2]. La stratégie du joueur est *probabiliste* :

- à chaque étape, le joueur sélectionne un bras suivant une distribution mélange de la distribution uniforme et d'une distribution affectant à chaque action une probabilité multipliée par un facteur exponentiel dans la récompense cumulée espérée ;
- l'algorithme utilise la *méthode des poids multiplicatifs* [10] pour mettre à jour les probabilités en les multipliant par un facteur dépendant des récompenses moyennes actuelles de tous les bras.

Cette méthode a tendance à donner plus d'importance aux décisions à récompenses plus fortes à long terme. Le résultat obtenu est que l'espérance du regret est *sous-linéaire* dans le nombre d'étapes :

$$\sup_{recompenses} \mathbb{E}R_n = O(\sqrt{nKlog(K)})$$

3 Les problèmes de décision markoviens

3.1 Les processus de décision markoviens

Les processus de décision markoviens constituent un cadre plus général pour la prise de décision dans lequel les actions modifient l'état de l'environnement :

- à chaque étape, un agent, ou un contrôleur, observe l'*état* du système et choisit une *action* de manière non déterministe,

- en retour, il reçoit une *récompense*, ou subit un coût, et le système évolue vers un nouvel état en suivant une *distribution de probabilités* déterminée par le choix de l'action.

Un problème de décision markovien est défini par la donnée d'un processus, d'un critère de performance associé à un horizon de temps fini ou infini, et d'un problème d'*optimisation* d'une fonction valeur associée à chaque état. Une *stratégie* du preneur de décision est une fonction associant une action à chaque état. Elle détermine ainsi une suite de transitions dont la valeur est la récompense totale ou le coût total. L'évolution du système à partir d'un état initial peut être représentée par l'*arbre des trajectoires* ou des traces d'exécution possibles à partir de cet état :

- dans un état s, on choisit une action a ;
- pour chaque couple (s, a), on reçoit une récompense ou on paie un coût $r(s, a)$, et on effectue une transition suivant une distribution de probabilités sur les états.

La *fonction valeur* V associe à chaque état une valeur suivant la trajectoire choisie. On désire calculer une fonction d'évaluation V^* qui doit fournir une valeur *optimale* pour chaque état suivant le critère de performance.

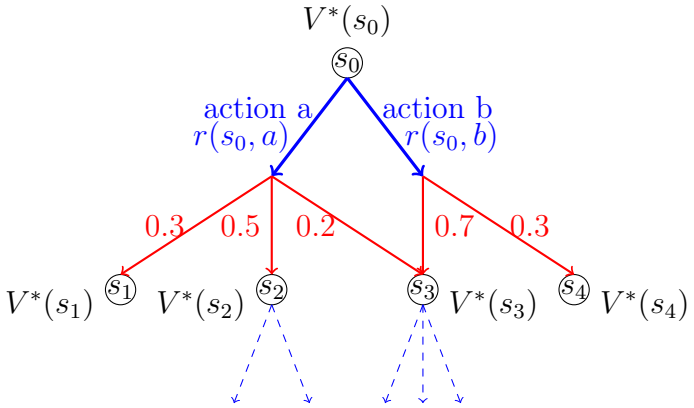

FIGURE 2 – Arbre des trajectoires à partir d'un état initial

Un processus de décision markovien $\mathcal{M} = (S, s_0, A, P, R)$ est ainsi défini par :

- un ensemble fini S d'états (le nombre d'états est n) ;
- un état initial s_0 (ou une distribution de probabilités μ sur les états) ;
- un ensemble fini A d'actions (le nombre d'actions est m) ;
- une relation de *transition* $P : S \times A \longrightarrow Distr(S)$ où $Distr(S)$ est l'ensemble des distributions de probabilités sur S ;
- une fonction de *récompense* (ou de coût) $R : S \times A \longrightarrow \mathbb{R}^+$

Pour chaque couple état-action $(s, a) \in S \times A$, on note :

- $P_{s,a}(.)$ la distribution de probabilités associée sur les états ;
- $R_{s,a}$ la récompense (ou le coût) associée à l'action a à partir de l'état s.

La résolution du non déterminisme sur les actions s'effectue de la manière suivante. Le comportement du décideur se traduit par une stratégie, supposée stationnaire c'est-à-dire toujours la même au cours du temps. Cette stratégie peut être :

- déterministe (et sans mémoire) $\pi : S \longrightarrow A$;
- probabiliste (et sans mémoire) $\pi : S \longrightarrow Distr(A)$ où $Distr(A)$ est l'ensemble des distributions de probabilités sur A.

Un résultat important montre que l'on peut se restreindre à rechercher des stratégies *déterministes* (et sans mémoire) optimales [12]. Dans l'arbre des trajectoires, on peut remarquer que chaque branche correspond à l'application d'une stratégie donnée. Si l'on se restreint à une stratégie, on obtient la *chaîne de Markov* induite par cette stratégie.

Un problème de décision markovien consiste à déterminer une stratégie optimale suivant un critère de performance définissant la fonction d'évaluation associée. Le critère de performance peut être de l'un des trois types suivants. Dans le cas d'un horizon de temps H fini, la valeur associée à chaque état s est la *récompense totale* espérée :

$$V^{\pi}(s) = \mathbb{E}_s^{\pi}(\sum_{i=1}^{H} R(s_i, \pi(s_i)))$$

où s_i est l'état résultant à l'étape i suivant la chaîne de Markov induite par la stratégie π. Dans le cas d'un horizon H infini, la valeur associée à chaque état s est la récompense totale espérée avec un *taux de discount* $0 < \gamma < 1$ sur les transitions :

$$V^{\pi}(s) = \mathbb{E}_s^{\pi}(\sum_{i=1}^{\infty} \gamma^{i-1} R(s_i, \pi(s_i)))$$

Dans le cas d'un horizon H infini, la valeur associée à chaque état s pourrait être la récompense espérée en moyenne :

$$V^{\pi}(s) = \lim_{T \to +\infty} \frac{1}{T} \mathbb{E}_s^{\pi}(\sum_{i=1}^{T} R(s_i, \pi(s_i)))$$

Dans la suite, on va s'intéresser au cas fini ou infini avec discount, en présentant les calculs pour le cas de l'*horizon infini avec discount*, celui de l'horizon fini étant beaucoup plus simple. On introduit une fonction auxiliaire d'évaluation pour chaque couple état action :

$$Q^{\pi}(s, a) = R_{s,a} + \mathbb{E}_s^{\pi}(\sum_{s' \in S} \gamma P_{s,a}(s') V^{\pi}(s'))$$

La fonction de récompense optimale est alors donnée par : $V^*(s) = max_\pi\ V^\pi(s)$. La valeur optimale de la fonction intermédiaire est utilisée pour déterminer la stratégie optimale :

$$Q^*(s,a) = max_\pi\ Q^\pi(s,a) \text{ pour tout } (s,a) \in S \times A.$$

La stratégie *optimale* π^* est définie par : $\pi^*(s) = argmax_a\ Q^*(s,a)$ pour tout $s \in S$.
La fonction de récompense optimale est solution d'une équation de point fixe dite équation de Bellman [3] :

$$V^*(s) = max_{a \in A}\ (R(s,a) + \gamma \sum_{s' \in S} P_{s,a}(s').V^*(s'))$$

Les algorithmes les plus largement utilisés pour la résolution des problèmes de décision markoviens sont des méthodes itératives dues en particulier à R. Bellman [3] et R.A. Howard [7].

3.2 Méthode d'itération sur la fonction valeur

Bellman a conçu un algorithme par approximation successives de la fonction d'évaluation optimale en supposant d'abord un horizon fini à une étape, puis à deux étapes et ainsi de suite. A chaque itération, on calcule les valeurs de la fonction intermédiaire et on en déduit les valeurs optimales de la fonction d'évaluation pour chaque état. La stratégie optimale est obtenue en associant à chaque état l'action optimisant la valeur finale de la fonction intermédiaire.

Algorithme d'itération sur la fonction valeur :
- Pour chaque $s \in S$, initialiser $V_0(s)$
- $h := 1$
- Tant que $h <$ nb maximum d'itérations
  pour chaque $(s,a) \in S \times A$,
  $Q_h(s,a) := R(s,a) + \gamma \sum_{s' \in S} P_{s,a}(s').V_{h-1}(s')$
  $V_h(s) := max_{a \in A} Q_h(s,a)$
  $h := h + 1$
- Pour chaque $s \in S$, $\pi^*(s) := argmax_{a \in A} Q_h(s,a)$
- Retourner π^*

Le nombre maximum d'itérations est déterminé :
- soit par l'horizon fini H ;
- soit par un critère d'arrêt dans le cas d'un horizon infini

$$max_{s \in S} \mid V_h(s) - V_{h-1}(s) \mid \leq \varepsilon' = \varepsilon \frac{(1-\gamma)}{2\gamma}$$

Ce critère garantit que la stratégie résultante π^* est ε-*optimale*. Le temps de calcul pour chaque itération est $O(mn^2)$. Il reste à montrer que le nombre d'itérations requis pour la convergence vers une stratégie presque optimale est polynomial, ce qui résulte d'une analyse plus fine, réalisée également pour la méthode suivante d'itération sur la stratégie.

3.3 Méthode d'itération sur la stratégie

La méthode d'itération sur la stratégie due à Howard alterne entre une phase d'évaluation de la stratégie courante et une phase où l'on tente d'améliorer cette stratégie.

Algorithme d'itération sur la stratégie :

- Soit π_0 une stratégie déterministe
- Boucle :

  $\pi := \pi_0$
  Déterminer, pour chaque $s \in S$, $V^\pi(s)$ par résolution de l'équation de Bellman (à n inconnues)
  Pour chaque $s \in S$, s'il existe $a \in A$ t.q. $(R(s,a) + \gamma \sum_{s' \in S} P_{s,a}(s').V^\pi(s')) < V^\pi(s)$, alors $\pi_0(s) := a$, sinon $\pi_0(s) := \pi(s)$
  Répéter la boucle si $\pi \neq \pi_0$
- Retourner π

On peut remarquer qu'il existe au plus m^n stratégies distinctes. Comme chaque stratégie améliore la précédente [12] l'algorithme termine en au plus un nombre exponentiel d'étapes. Comme on l'a vu, l'algorithme procède en deux phases :

- la détermination de la fonction valeur pour une stratégie donnée s'effectue par résolution d'un système d'équations linéaires en $O(n^3)$ étapes ;
- l'amélioration éventuelle de la stratégie requiert $O(mn^2)$ étapes.

Plusieurs résultats récents ont permis de montrer que l'algorithme d'Howard est de complexité en temps fortement polynomial. Hansen, Miltersen et Zwick [5] ont obtenu que le nombre d'itérations est en $O(\frac{m}{1-\gamma} log(\frac{n}{1-\gamma}))$.

3.4 Le problème de la grande dimension

Ces méthodes classiques d'optimisation en univers incertain sont souvent impraticables en pratique, en particulier dans un cadre d'apprentissage par renforcement. En effet, elles sont *polynomiales* dans la taille du modèle, mais la phase de modélisation conduit à une *explosion combinatoire* de l'espace des états. Pour tenter de franchir cette barrière de

complexité, plusieurs méthodes de recherche dans l'arbre des trajectoires possibles ont été élaborées en utilisant des techniques d'échantillonnage ou de simulation du type Monte-Carlo.

4 Méthodes de recherche dans les arbres

4.1 Méthodes de recherche par échantillonnage

La première méthode consiste à faire de la planification en ligne dans un processus de décision markovien à partir d'un état initial. Elle a été mise au point par M. Kearns, Y. Mansour et A.Y. Ng [8] et utilise un algorithme d'*échantillonnage clairsemé* :

- pour chaque action a, on engendre C successeurs de l'état s ;
- l'arbre des trajectoires est alors étendu jusqu'à un horizon H ;
- à une profondeur h, on estime récursivement la valeur $V_h^*(s)$ par

$$\hat{V}_h^*(s) = max_a \left(R(s,a) + \frac{\gamma}{C}\sum_{i=1}^{C} \hat{V}_{h-1}^*(succ_i(s,a))\right)$$

Les paramètres H et C sont choisis de manière à obtenir une stratégie presque optimale et sont *indépendants* de la taille n de l'espace des états.. L'arbre des trajectoires ainsi construit est de taille $(mC)^H$ et le temps de calcul est exponentiel dans l'horizon H.

La seconde méthode est due à H.S. Chang, M.C. Fu et S.I. Marcus [4] et utilise un algorithme d'*échantillonnage adaptatif* (AMS pour Adaptive Multi-stage Sampling) :

- plutôt qu'échantillonner uniformément pour chaque action, l'algorithme UCB permet de choisir l'action à utiliser ;
- les meilleure actions sont ainsi essayées plus souvent mais les autres actions sont aussi explorées ;
- à chaque étape, on procède de manière analogue à la résolution d'un problème de bandit stochastique à plusieurs bras ;
- le nombre d'états échantillonnés est adapté à chaque étape.

Ces deux algorithmes utilisent un *générateur probabiliste* de trajectoires. La complexité est indépendante du nombre d'états.

4.2 Méthodes de recherche par simulation Monte-Carlo

Un exemple de telle méthode est l'algorithme UCT, pour Upper Confidence Bound for Trees, dû à L. Kocsis et C. Szepesvari [9]. L'idée de base est la suivante :

- le choix d'un noeud successeur est traité comme un problème de bandit à plusieurs bras ;
- la *stratégie UCB* est appliquée sur les noeuds intérieurs ;
- à chaque noeud, est associée une statistique composée d'une récompense moyenne et d'un compteur du nombre de visites ;
- la récompense d'un noeud successeur est la moyenne obtenue comme approximation par une *simulation* du type Monte-Carlo à partir de ce noeud.

La méthode de parcours de l'arbre des trajectoires est telle que si le noeud sélectionné par la stratégie UCB a des successeurs non explorés, l'un des successeurs est choisi aléatoirement. La simulation de type Monte-Carlo à partir d'un noeud est uniformément aléatoire. L'analyse de cet algorithme permet de montrer que la probabilité de sélectionner une action sous-optimale converge de manière asymptotique vers 0. Cependant, le temps nécessaire pour découvrir finalement une branche optimale peut être très long. Une référence de base pour une étude plus complète des méthodes dites MCTS, pour Monte-Carlo Tree Search, est l'article de R. Munos [11].

5 Conclusion

L'apprentissage par renforcement a connu un renouveau récent dû à la fois aux applications potentielles à l'allocation séquentielle de ressources, ainsi qu'aux résultats impressionnants dans le domaine des jeux. Dans le cadre le plus simple, l'importance de la modélisation par *bandits manchots* a permis de mettre au point un compromis entre le problème d'exploration et celui d'exploitation, illustré par un principe d'optimisme dans l'incertain. Le cadre le plus général pour l'étude des problèmes d'optimisation en univers incertain est celui des *processus de décision markoviens*. Les méthodes classiques, qui sont de complexité en temps polynomial dans la taille du modèle, sont cependant très souvent impraticables en pratique, pour cause d'explosion combinatoire de la phase de modélisation. Les méthodes de recherche dans les arbres par *échantillonnage*, éventuellement adaptatif, ont permis d'obtenir des algorithmes dont la complexité est indépendante de la taille du domaine. Pour l'apprentissage par renforcement proprement dit, les méthodes de recherche utilisent la *simulation* du type Monte-Carlo pour évaluer les meilleures branches dans l'arbre des trajectoires possibles. Ces méthodes alliées à des techniques d'apprentissage profond, utilisant les réseaux de neurones convolutifs, ont permis d'obtenir les succès récents des programmes de jeux, comme AlphaGo et surtout AlphaGo zéro.

Références

[1] P. Auer, N. Cesa-Bianchi, and P. Fisher. Finite time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3) :235–256, 2002.

[2] P. Auer, N. Cesa-Bianchi, P. Fisher, and R.E. Schapire. The nonstochastic multi-armed bandit problem. *SIAM Journal of Computing*, 32 :48–77, 2002.

[3] R. Bellman. Dynamic Programming. Princeton University Press, 1957.

[4] H.S. Chang, M.C. Fu, and S.I. Marcus. An Adaptive Sampling Algorithm for Solving Markov Decision Processes. *Operations Research*, 53(1) :126–139, 2002.

[5] T.D. Hansen, P. B. Miltersen, and U. Zwick. Strategy iteration is strongly polynomial. In *Innovations in Computer Science*, pages 253–263, 2011.

[6] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301) :13–30, 1963.

[7] R.A. Howard. Dynamic Programming and Markov process. MIT Press, 1960.

[8] M. Kearns, Y. Mansour, and A.Y. Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning*, 49(2–3) :193–208, 2002.

[9] L. Kocsis and C. Szepesvari. Bandit based Monte-Carlo planning. In *17th European Conference on Machine Learning*, pages 282–293, 2006.

[10] R. Lassaigne. La méthode des poids multiplicatifs : un algorithme générique pour l'apprentissage et l'optimisation. *Bulletin de l'Union des Professeurs de classes préparatoires Scientifiques*, 257 :42–53, 2017.

[11] R. Munos. From Bandits to Monte-Carlo Tree Search : The Optimistic Principle Applied to Optimization and Planning. *Foundations and Trends in Machine Learning*, 7(1) :1–129, 2014.

[12] M. Puterman. Markov Decision Processes. John Wiley and Sons, 1994.

[13] R.S. Sutton and A.G. Barto. Reinforcement Learning : An introduction. Bradford Book. MIT Press, 1998.