Apprentisage par renforcement: une approche markovienne pour le jeu Pierre-Feuille-Ciseaux

Danilo Fernandes

danilo.marinho-fernandes@polytechnique.edu

Keywords: Apprentisage par renforcement, Markov Decision Process.

Résumé. La présente étude analyse des algorithme pour le jeu Pierre-Feuille-Ciseaux, où nous considèrons que l'adversaire induit un processus de décision markovien (MDP) dont les états sont des suites de paires de choix algorithme-adversaire de taille à choisir, ce qui est une hypothèse raisonnable pour un joueur humain [4]. L'algorithme se base sur l'approche exploration-exploitation, qui consiste à soit explorer un mouvement (dont le choix sera optimisé de façon à maximiser le gain d'information sur la stratégie de l'adversaire), soit à exploiter, c'est à dire faire le mouvement avec la plus haute probabilité de gain selon l'information déjà acquise. Nous essayons ensuite d'ajuster le taux d'exploration selon l'évolution du jeu, afin d'optimiser le gain initial tout en convergeant vers la sratégie optimale. Pour cela nous introduisons deux algorithmes du type Q-Learning: le ϵ-greedy et l'algorithme UCB.

Abstract. The present study analyzes algorithms for the Rock-Paper-Scissors game, where we consider that the opponent induces a Markov decision process (MDP) whose states are sequences of algorithm-opponent choice pairs of size to be chosen, which is a reasonable assumption for a human player [4]. The algorithm is based on the exploration-exploitation approach, which consists in either exploring a move (whose choice will be optimized in order to maximize the gain of information on the opponent's strategy), or exploiting, i.e. making the move with the highest probability of gain according to the information already acquired. We then try to adjust the exploration rate according to the evolution of the game, in order to optimize the initial gain while converging towards the optimal strategy. For this we introduce two Q-Learning algorithms: the ϵ-greedy and the UCB algorithm.

Sommaire

Introduction

Les règles du jeu Pierre-Feuille-Ciseaux sont simples : à chaque tour, deux adversaires choisissent simultanément parmi pierre, feuille ou ciseaux. La feuille bat la pierre, la pierre bat les ciseaux et les ciseaux battent la feuille. Si les coups sont choisis de manière totalement aléatoire, aucun des joueurs ne devrait avoir d'avantage et les probabilités que l'un des joueurs gagne ou que le tour se termine par une égalité sont toutes aussi probables.

Il s'avère néanmoins que les humains ne sont généralement pas capables de faire des choix selon une distribution uniforme parfaite, et qu'un joueur pourrait apprendre les probabilités de transition entre les coups précédents de l'adversaire et son choix pour le coup suivant [1]. Compte tenu de ce fait, de nombreuses approches markoviennes ont été appliquées pour créer des algorithmes efficaces contre les humains. Il s'avère que le jeu Pierre-Feuille-Ciseaux est fondamental pour de nombreux problèmes de théorie des jeux, et des approches récentes utilisant des intelligences artificielles ont été créées, dont certaines ont gagné contre plus de 95% des joueurs humains lors de 300 tours continus [2].

Nous supposons dans cet article que l'adversaire o agit selon une machine à états finis ayant une taille de mémoire de K. K est la taille de la mémoire de o si la séquence d'actions conjointes des K étapes précédentes détermine complètement le prochain profil d'action stochastique de o. Cela nous permet de considérer le jeu comme un MDP ([3]) où l'on note:

- a l'algorithme, o l'adversaire.

- États: $S_t \in \{\{R, P, S\}X\{R, P, S\}\}^K$ réprésente les derniers K paires conjoints de la forme $(coup\,de\,l'algorithme, coup\,de\,l'adversaire)$, au temps t. L'espace d'états $\mathcal{S}$ est constitué par tous les états possibles.

- Actions: $A_t = A_{a,t} \in \{R, P, S\}$ réprésente le dernier coup de l'algorithme, et $A_{o,t} \in \{R, P, S\}$ le dernier coup de l'adversaire au temps t. L'espace d'actions $\mathcal{A} = \{R, P, S\}$ est constitué par toutes les actions possibles.

- Politique: $\pi_o : (\{R, P, S\}X\{R, P, S\})^K \to [0, 1]^3$ représente la stratégie de o, c'est à dire les probabilités $\mathbb{P}(pierre), \mathbb{P}(papier), \mathbb{P}(ciseaux)$ de l'adversaire jouer pierre, feuille ou ciseaux après une suite de K coups conjoints.

- Fonction de transition: la probabilité de passer de l'état S_1 à l'état S_2 en effectuant l'action A_a est la suivante :
  1) Pour les états S_2 qui se terminent avec a jouant A_a, la probabilité de transition est la probabilité de $\pi_o(S_1)$ jouer A_o, où (A_a, A_o) est la dernière paire d'action conjointe dans l'état S_2.
  2) Pour tous les états S_2 qui ne se terminent pas avec a jouant A_a, la probabilité de transition vaut 0.

- Fonction de récompense: la récompense obtenue lors de la transition vers l'état S_2 en effectuant l'action A_a dans l'état S_1 est notée par $R_{S_1}(A_a, A_o)$, où (A_a, A_o) est la dernière paire d'actions conjointes dans l'état S_2. On remarque qu'elle vaut 1 si l'algorithme gagne, -1 s'il perd et 0 s'il fait match nul.

Nous utiliserons $k = 2$ pour simplicité, vu que cette valeur donne un résultat satisfaisant et permet une convergence plus rapide de l'algorithme qu'un grand k.

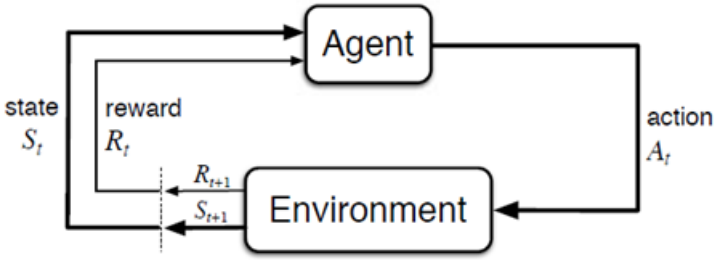


Fig. 1: L'interaction agent-environnement dans un MDP.

Notre algorithme est ouvert à plusieurs améliorations, notamment en ce qui concerne la vitesse d'adaptation à un éventuel changement de stratégie de l'adversaire. Nous verrons néanmoins qu'il est très efficace contre un joueur dont la stratégie ne change pas avec le temps.

Analyse des algorithmes

Notre algorithme est basé sur une approche d'exploration-exploitation, qui consiste essentiellement à soit faire un coup qui maximise le gain attendu compte tenu de l'information actuelle (exploitation), soit à faire un coup pour gagner plus d'information sur le modèle (exploration). Il existe une solution bayésienne pour apprendre et jouer de manière optimale à condition de disposer d'un modèle approprié de l'environnement [6], qui est conceptuellement correcte et résout implicitement et correctement le trade-off entre exploration et exploitation. Cependant, nous avons choisi une approche du type Q-Learning en raison de sa simplicité et de ses performances relativement bonnes pour les problèmes d'apprentissage par renforcement.

Le Q-learning est un algorithme qui cherche à trouver la meilleure action à entreprendre compte tenu de l'état actuel. Il ne nécessite pas de modèle de l'environnement et peut traiter des problèmes avec des transitions et des récompenses stochastiques sans nécessiter d'adaptations. Pour tout MDP fini, le Q-learning essaie de trouver une stratégie optimale dans le sens où elle maximise la valeur attendue de la récompense totale sur toutes les étapes successives, à partir de l'état actuel. Pour cela, nous définissons la fonction action-valeur Q d'une action A partant d'un état S comme la récompense moyenne de l'action, c'est-à-dire,

$$Q(A, S) = \mathbb{E}[R|A, S]$$

La valeur optimale dans un état est définie comme

$$V^*(S) = \max_{A \in \mathcal{A}} Q(A, S)$$

Le regret est une mesure de la perte d'opportunité pour une étape

$$l_t = \mathbb{E}[V^*(S_t) - Q(A_t, S_t)]$$

Le regret total est la perte d'opportunité totale

$$L_t = \mathbb{E}[\sum_{\tau=1}^{t} V^*(S_\tau) - Q(A_\tau, S_\tau)]$$

L'objective de notre algorithme est de maximiser la récompense totale, ce qui est équivalent à minimiser le regret total. Pour cela, nous essayons d'abord d'estimer, à chaque temps t, la valeur $Q_t(A) = Q(A, S_t)$ pour chaque $A \in \mathcal{A}$. Une approche du type *Monte-Carlo* consiste à utiliser un estimateur

$$\hat{Q}_t(A) = \frac{1}{N_t(A)} \sum_{\tau=1}^{t} R_\tau 1(A_\tau = A, S_\tau = S_t)$$

Où $N_t(A)$ est le quantité de fois où l'action A a été exécutée avant le temps t en partant d'un état équivalent à l'état S_t. En résumé, nous considérons comme plus prometteuses les actions qui ont été plus récompensées dans le passé. Nous allons désormais analyser deux approches qui essaient de minimiser le regret total: l'algorithme ϵ-greedy et l'algorithme UCB.

L'algorithme ϵ-greedy

L'algorithme greedy execute l'action A_t avec la plus haute valeur de $\hat{Q}_t(A)$, c'est à dire,

$$A_t^* = \underset{A \in \mathcal{A}}{\text{argmax}}\ \hat{Q}_t(A)$$

Cet algorithme peut se bloquer sur une action sous-optimale pour toujours. Une solution pour cela est d'utiliser l'algorithme ϵ-greedy, qui va toujours explorer de nouveles actions selon une probabilité ϵ:

- Avec probabilité ϵ, l'algorithme prend une action aléatoire.
- Avec probabilité $1 - \epsilon$, l'algorithme prend $A = \underset{A \in \mathcal{A}}{\text{argmax}}\ \hat{Q}(A)$.

Le calcul du regret à chaque étape donne

$$l_t = \mathbb{E}[V^*(S_t) - Q(A_t, S_t)]$$

$$\geq \mathbb{E}[\min_{S \in \mathcal{S}}(V^*(S) - Q(A_t, S))]$$

$$\geq \frac{\epsilon}{|\mathcal{A}|} \sum_{A \in \mathcal{A}} \mathbb{E}[\min_{S \in \mathcal{S}}(V^*(S) - Q(A, S))] = \frac{\epsilon}{|\mathcal{A}|} \sum_{A \in \mathcal{A}} \Delta_A$$

$\Delta_a \geq 0$, d'où le regret total est linéaire en N (nous pouvons faire une borne supérieure constante pour l_t de manière analogue).

Une alternative est de prendre ϵ_t qui dépend de t. Nous pouvons démontrer que, si $\epsilon_t \sim \frac{1}{t}$, l'algorithme ne va pas se bloquer sur une action sous-optimale, c'est à dire,

$$\mathbb{P}(\lim_{t \to +\infty} A_t = \underset{A \in \mathcal{A}}{\text{argmax}}\ Q(A, S_t)) = 1$$

Démonstration: si l'on considère les événements $(X_t)_{t \in \mathbb{N}}$, où X_t correspond à l'algorithme explorer au temps t, nous voyons que $\lim_{t \to +\infty} \sum_{\tau=1}^{t} \mathbb{P}(X_\tau) = \infty$. Ces événements sont indépendants, d'où, par le théorème de Borel-Cantelli, $\mathbb{P}(limsup_{t \to +\infty}(X_t)) = 1$, c'est à dire, l'algorithme explore une infinité de fois avec probabilité 1. Or, il est facile de démontrer que, lorsque la quantité d'explorations tend vers l'infini, la probabilité de faire un choix sous-optimale tend vers zéro.

On peut également démontrer que le regret total est logarithmique en t, ce qui découle de l'expansion de L_t dans une série harmonique.

L'algorithme UCB

Bien que l'algorithme *e-greedy* avec $\epsilon_t = \frac{1}{t}$ converge vers le choix d'action optimal, il présente un défaut : l'exploration est faite de manière aléatoire, sans aucune tentative de maximiser le gain d'informations sur la politique de l'adversaire, c'est-à-dire d'explorer les actions dont on est moins sur de la valeur pour le moment.

L'algorithme *UCB* (Upper Confidence Bound) essaie d'estimer une valeur de confiance supérieure d'une action $\hat{U}_t(A)$ telle que

$$Q(A) \leq \hat{Q}_t(A) + \hat{U}_t(A)$$

avec haute probabilité.

Cela depend de la quantité de fois $N(A)$ que A a été sélectionnée:

- Si $N_t(A)$ est petit, $\hat{U}_t(A)$ est grand, car la valeur estimée de A est moins certaine.
- Si $N_t(A)$ est grand, $\hat{U}_t(A)$ est petit, car la valeur estimée de A est plus certaine.

Notre bout est donc de sélectionner une action qui maximise le Upper Confidence Bound (UCB):

$$A_t = \underset{A \in \mathcal{A}}{\text{argmax}}(\hat{Q}_t(A) + \hat{U}_t(A))$$

Pour trouver une borne supérieure pour la probabilité de $Q(A)$ être plus grand que $\hat{Q}_t(A) + \hat{U}_t(A)$ nous allons utiliser le résultat suivant:

Théorème (Inégalité de Hoffding) Soit X_1, ..., X_t des variables aléatoires i.i.d. à valeurs dans $[0, 1]$, et $\overline{X}_t = \frac{1}{t} \sum_{\tau=1}^{t} X_\tau$ leur moyenne. Alors

$$\mathbb{P}(\mathbb{E}[X] > \overline{X}_t + u) \leq e^{-2tu^2}$$

Une application directe de ce théorème nous donne

$$\mathbb{P}(Q(A) > \hat{Q}_t(A) + \hat{U}_t(A))) \leq e^{-2N_t(A)\hat{U}_t(A)^2}$$

Nous pouvons prendre $e^{-2N_t(A)\hat{U}_t(A)^2} = t^{-2}$ (vu que nous sommes plus certains de la valeur des actions lorsque le temps avance), qui donne, pour $N_t(A) \geq 1$

$$\hat{U}_t(A) = \sqrt{\frac{log(t)}{N_t(A)}}$$

L'algorithme va donc prendre au temps t l'action

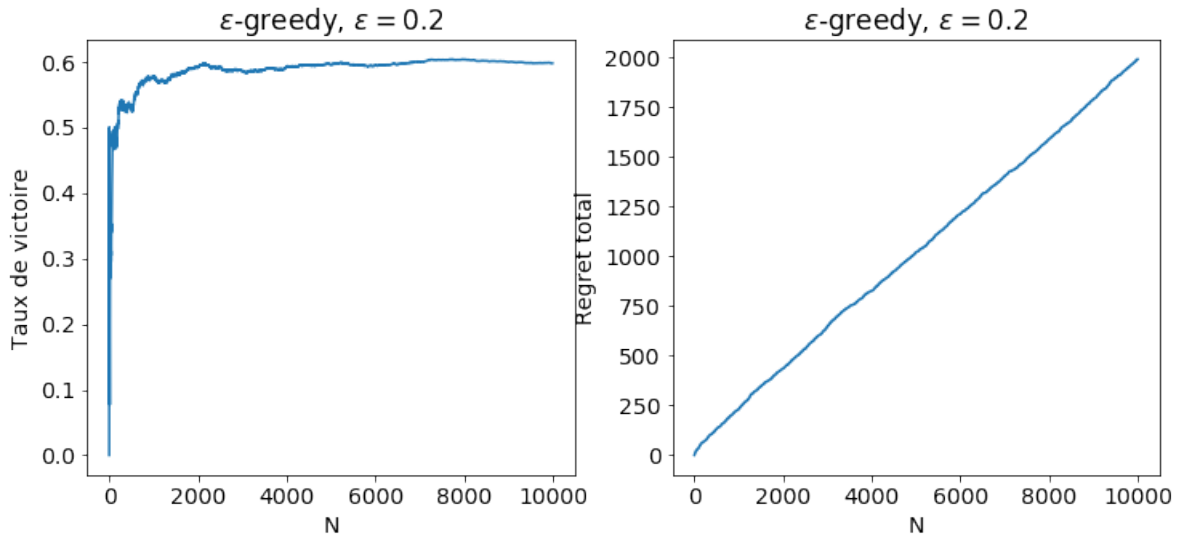
$$A_t = \underset{A \in \mathcal{A}}{\text{argmax}} \hat{Q}_t(A) + \sqrt{\frac{log(t)}{N_t(A)}}$$

Il est possible de démontrer que l'algorithme *UCB* a un regret asymptotique logarithmique.

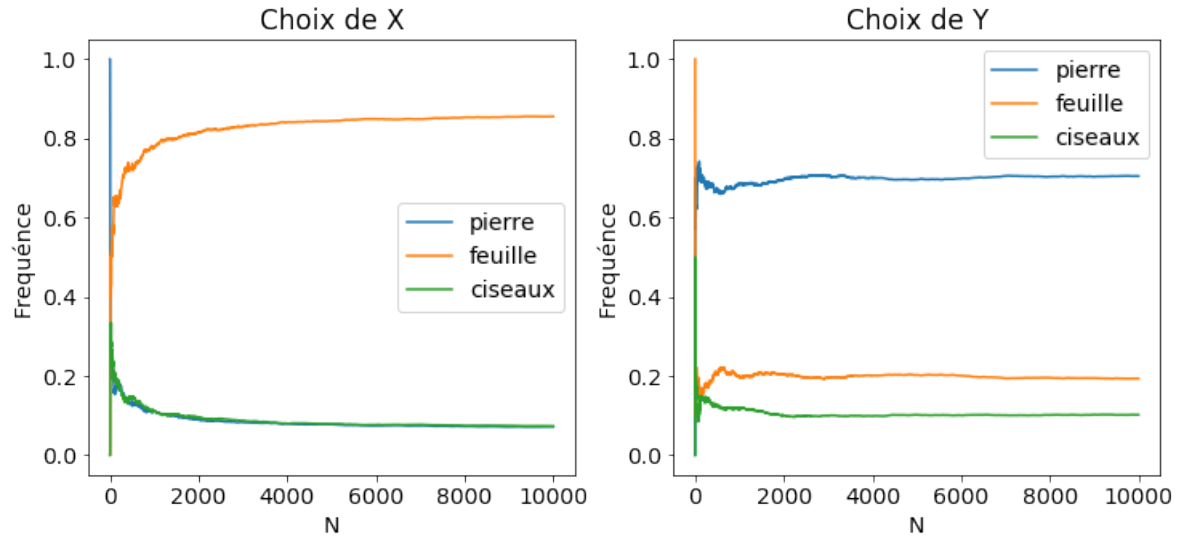
Résultats

Une première simulation des algorithmes a été faite contre un adversaire qui adopte une stratégie simple: jouer pierre, feuille ou ciseaux avec probabilités 0.7, 0.2 et 0.1, respectivement. Cet exemple permet d'observer clairement l'apprentisage de l'algorithme et l'évolution du regret avec le temps.

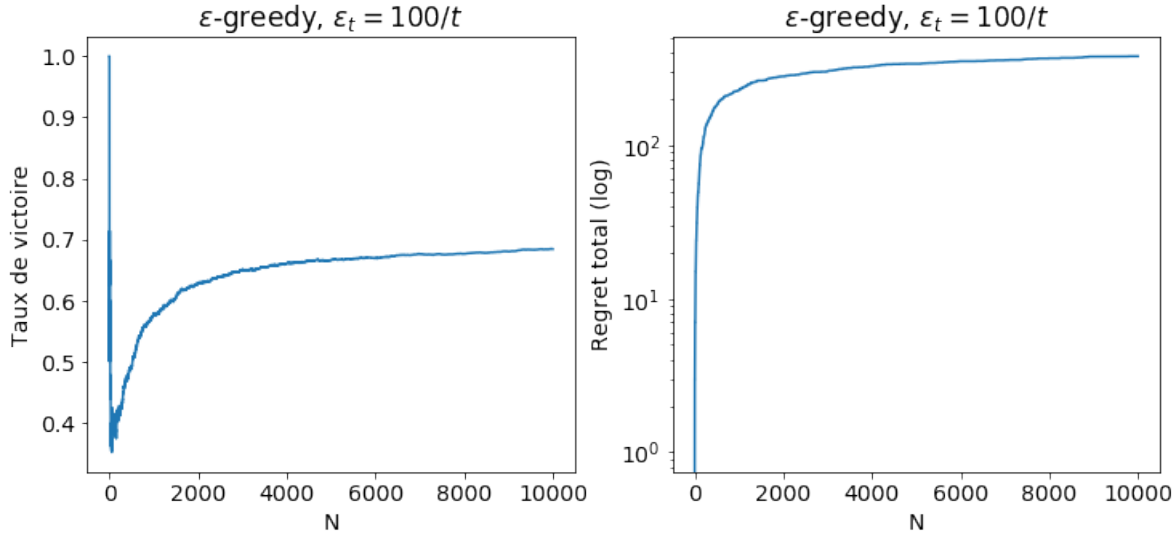
Pour un total de $N = 10000$ simulations, l'algorithme ϵ-greedy avec $\epsilon = 0.2$ a atteint un taux de victoire proche de 0.6, correspondant à soit jouer aléatoirement avec probabilité ϵ, soit jouer selon la stratégie optimale (toujours feuille) avec probabilité $1 - \epsilon$. Nous voyons également le profil linéaire du regret total, comme prévu.



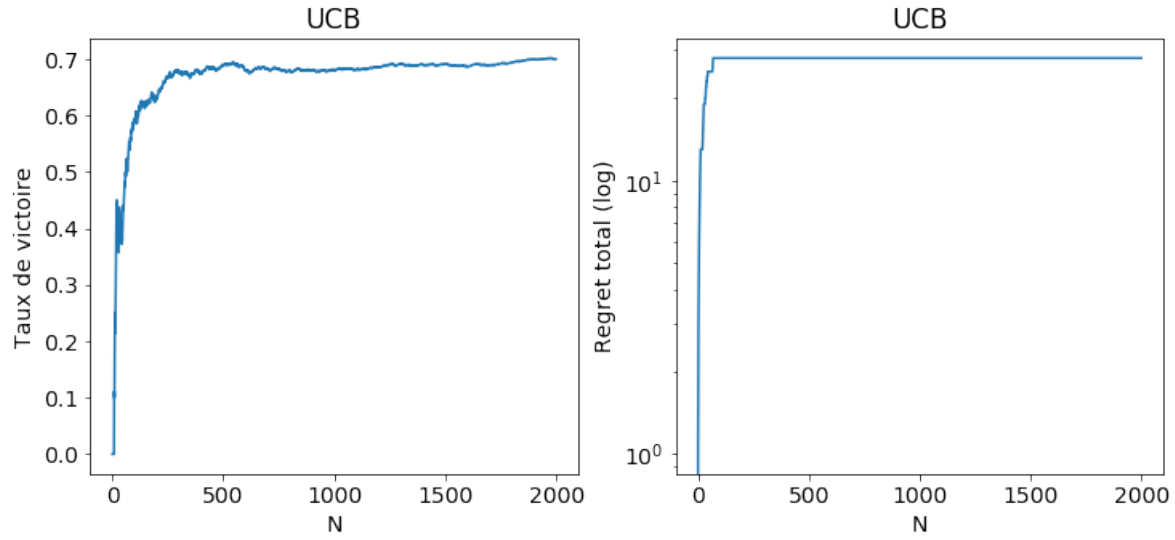
Une simulation de l'evolution des coups joués par l'algorithme ϵ-greedy permet de voir qu'avec probabilité d'environ $1 - \epsilon + \frac{\epsilon}{3} = 0.87$, l'algorithme choisit l'action optimale (feuille), et sinon choisit pierre ou ciseaux avec probabilité égale.



Ensuite, la simulation de l'algorithme ϵ-greedy avec $\epsilon_t = \frac{100}{t}$ montre un taux de réussite asymptotiquement égal au taux optimal, correspondant à la stratégie optimale de jouer toujours feuille. Le choix de $\epsilon_t = \frac{100}{t}$ au lieu de $\epsilon_t = \frac{1}{t}$ a été fait pour accélérer l'exploration au début, et ainsi avoir une convergence plus rapide. Nous voyons aussi que la convergence est plus rapide que l'algorithme avec ϵ constant, et que le regret total est approximativement linéaire en $log(t)$ pour t assez grand.



L'algorithme UCB présente la convergence la plus rapide, de telle sorte que le regret total reste constant après $t = 100$, c'est à dire, l'algorithme ne fait plus de choix sous-optimal après ce temps.



Conclusion

En considérant le jeu Pierre, Feuille, Ciseaux comme un MDP dont les états sont des suites de paires de choix algorithme-adversaire de taille $k = 2$, les algorithmes du type Q-Learning réussissent à trouver la stratégie optimale contre un adversaire dont la politique ne change pas. Nous voyons que pour l'algorithme ϵ-greedy avec $\epsilon_t = \frac{1}{t}$, il est possible d'avoir un regret logarithmique en t, et donc un taux de choix sous-optimaux qui tend vers zéro quand t tend vers l'infini. Il est possible de faire encore mieux avec l'algorithme UCB, qui optimise l'exploration en choisissant des actions dont la valeur est moins certaine.

Il reste, néanmoins, plusieurs possibilités d'amélioration de l'algorithme. Une approche baysienne, où l'on part d'un modèle pour la politique de l'adversaire à fin de minimiser à chaque étape l'espérance du regret total lorsque le temps tend vers l'infini, serait optimale. Cépendant, cette aproche est plus complexe à implémenter et demande plus de temps de calcul. Une autre amélioration possible de l'algorithme consiste à l'adapter à un possible changement de stratégie de l'adversaire ([2]). Les algorithmes présentés sont aussi applicables à plusieurs autres jeux et problèmes qui partagent le cadre des MDP, d'où leur large applicabilité.

Bibliographie

[1] N. Pomerleau. Rock Paper Scissors - Playing the Game with Markov Chains. Worcester Polytechnic Institute, 2013.

[2] L. Wang, W. Huang, Y. Li, J. Evans, S. He. Multi-AI competing and winning against humans in iterated Rock-Paper-Scissors game. Nature, 2020.

[3] D. Chakraborty, P. Stone. Online Model Learning in Adversarial Markov Decision Process. Proc. of 9th Int. Conference of Autonomous Agents and Multiagent Systems (AAMAS 2010).

[4] D. Chakraborty, P. Stone. Online Multiagent Learning Against Memory Bounded Adversaries. ECML, pages 211–226, Antwerp,Belgium, 2008.

[5] Sutton, R.S. Barto, A.G., 2018. Reinforcement learning: An introduction, MIT press.

[6] Duff, M.O.: Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes. PhD thesis, University of Massachusetts Amherst (2002)

[7] Silver, D.: Introduction to Reinforcement Learning, 2015. https://deepmind.com/learning-resources/-introduction-reinforcement-learning-david-silver.