Des casinos à l'intelligence articficielle

TER M1 DS 2021

Adrien Maitammar
Maxime Le Paumier

11 mai 2021

Sommaire

1 Introduction

Le premier objectif de ce projet est l'étude de l'algorithme UCB (Upper Confidence Bound). Cet algorithme a été proposé en 2002 dans le cadre du problème du bandit à K bras.

1.1 Le problème du bandit à K bras

Le problème du bandit manchot ou, dans sa version générale, du bandit à K bras, peut se formuler ainsi : un agent est face à des machines à sous (portant le nom de "bandits manchots") dont il ne connait pas les récompenses moyennes. Son objectif est alors de maximiser ses gains.

Ce problème est en fait un exemple d'apprentissage par renforcement dans lequel l'agent doit faire des compromis entre l'exploitation (de la machine qui empiriquement lui rapporte le plus) et l'exploration (pour espérer trouver une machine rapportant plus).

L'agent peut bien sûr adopter une approche naïve et jouer la machine maximisant la récompense moyenne sur les coups joués mais cette stratégie peut le conduire à ignorer certaines machines, qui pourraient être plus intéressantes à jouer.

Peter Auer, Nicolò Cesa-Bianchi et Paul Fischer développent dans leur article *Finite-time Analysis of the Multiarmed Bandit Problem* une stratégie permettant à l'agent de continuer à explorer les machines tout en maximisant ses gains. Plutôt que de procéder naïvement, celui-ci joue les machines selon la procédure UCB1 ci-dessous.

UCB1
Initialisation : Jouer chaque machine une fois
Boucle : Jouer la machine j qui maximise $\bar{x_j} + \sqrt{2\frac{\ln n}{n_j}}$ avec $\bar{x_j}$ la récompense moyenne de la machine j jusqu'à présent, n_j le nombre de fois où la machine j a été jouée et n le nombre de coups total.

FIGURE 1 – Algorithme UCB1

On retrouve dans cette quantité un terme d'exploitation (la moyenne des gains de la machine) ainsi qu'un terme d'exploration donnant plus d'importance aux machines ayant été visitées moins de fois pondéré par $\sqrt{2}$, que l'on appelle constante d'exploration (sur laquelle nous aurons à revenir dans le cadre du jeu de Hex).

1.2 Le jeu de Hex

Dans le cadre de ce projet, nous avons cherché à appliquer cet algorithme à un joueur artificiel du jeu de Hex. Il s'agit d'un jeu de société pour deux joueurs dont le but est de relier les deux côtés du plateau correspondant à sa couleur par une ligne continue composée de pièces hexagonales. Avec des règles pourtant simples, la complexité de ce jeu est comparable à celle des échecs, dans le sens où les nombres de coups possibles sont comparables.

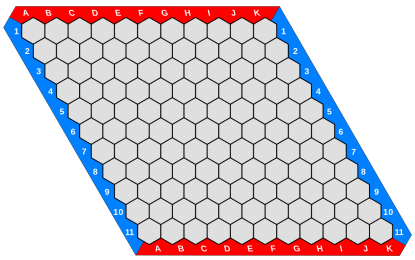

FIGURE 2 – Plateau de jeu 11x11

Nous verrons dans un premier temps une manière d'implémenter ce jeu sous python puis nous détaillerons deux méthodes de programmation d'un joueur artificiel : une méthode Monte-Carlo puis une méthode UCB appliqué à une recherche arborescente Monte-Carlo que nous appellerons UCT (Upper Confidence bound applied to Trees).

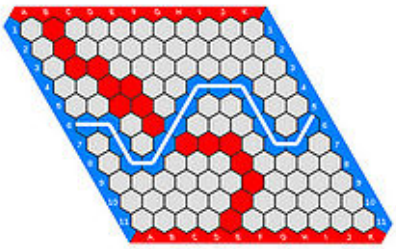

FIGURE 3 – Configuration gagnante pour le joueur bleu

2 Implémentation du jeu de Hex

2.1 subsection

3 La méthode Monte-Carlo

3.1 subsection

4 La méthode UCT

La méthode Monte-Carlo, bien que relativement efficace, présente pour principal défaut un nombre de calcul très important. Là où cette dernière simule tous les coups possibles, on va plutôt chercher à ne simuler que certains coups, choisis par le critère UCB, dans le cadre d'une recherche arborescente Monte-Carlo.

4.1 Recherche arborescente Monte-Carlo

La recherche arborescente Monte-Carlo (Monte Carlo tree search) ...

4.2 Implémentation de la méthode UCT

Dans le cas du jeu de Hex, on ne cherche plus à maximiser des gains d'argent, mais des taux de victoires.

```
bestValue = float("-inf")
bestNodes = []

for child in node.children.values():
   nodeValue = child.totalReward / child.numVisits + explorationValue *
        sqrt(log(node.numVisits) / child.numVisits)

       if nodeValue > bestValue:
          bestValue = nodeValue
          bestNodes = [child]
        elif nodeValue == bestValue:
          bestNodes.append(child)

return random.choice(bestNodes)
```

4.3 Recherche de la constante d'exploration optimale

4.4 Améliorations possibles

Références

[1] Peter Auer, Nicolò Cesa-Bianchi & Paul Fischer, *Finite time Analysis of the Multiarmed Bandit Problem*

[2] Levente Kocsis & Csaba Szepesvàri, *Bandit Based Monte-Carlo Planning*