Probabilité, Statistiques et Informatique : Exploration/Exploitation et Puissance 4

SORBONNE UNIVERSITÉ (2023/2024) - Participants :

Lazrak Ali 28605235

Bouchaal Samia 28610654

Sommaire

Introduction

Le but de ce projet est d'étudier différents algorithmes du dilemme d'exploration vs exploitation pour des IAs de jeux. Nous comparerons ces algorithmes au travers de différentes expériences dont les résultats seront regroupés dans ce rapports.

Dans un premier temps, nous ferons une analyse combinatoire du jeu Puissance 4, puis nous analyserons dans la partie d'après l'algorithme de Monte-Carlo sur le même jeu, ensuite nous ferons une comparaison des algorithmes aléatoire, Monte-Carlo, glouton, ε-greedy et UCB (Upper Confidence Bound) sur l'exemple des bandits-manchots et enfin, nous verrons l'algorithme UCT sur le jeu de Puissance 4.

Combinatoire du puissance 4

Étude de la distribution du nombre de coups avant une victoire

Pour l'étude combinatoire du jeu Puissance 4, nous avons implémenté une simulation générique du jeu avec Python dans le but de pouvoir changer les dimensions du plateau à notre guise.

Après implémentation d'un joueur qui joue de manière aléatoire parmis les coups possibles, on veut étudier la distribution du nombre de coups avant une victoire lorsque les deux joueurs jouent aléatoirement en différenciant lorsque le premier joueur gagne ou le second joueur gagne.

Pour cela, on peut simuler n parties, avec n un nombre très grand (nous avons ici choisi 10 000), et avoir une approximation de la distribution.

Pour l’étude des résultat de cette expérience, on calcule la moyenne des coups avant une victoire pour chaque joueur, ainsi que la variance, la médiane et l’écart-type.

Dans un premier temps, nous calculons les moyennes $\overline{x_1}$ et $\overline{x_2}$ des nombres de coups $n1_i$ et $n2_i$ avant les victoires des joueur 1 et 2 respectivement, avec i, le numéro de la partie.

$$\overline{x_1} = \frac{\sum_{i \to nb_{victoires joueur 1}}^{i=0} n1_i}{nb_{victoires joueur 1}} = 10.90$$

$$\overline{x_1} = \frac{\sum_{i \to nb_{victoires joueur 2}}^{i=0} n2_i}{nb_{victoires joueur 2}} = 10.81$$

On remarque que la moyenne ne change pas grandement d’un joueur à l’autre. Ça n’est pas surprenant en considérant qu’ils placent tous les deux leurs jetons de manière aléatoire et indépendante, donc sur un grand nombre de parties, les actions des joueurs ne varient pas grandement. Pour le plateau de dimension 6x7, on peut considérer qu’en moyenne il faudrait entre 10 et 11 coups pour gagner.

Les médianes M_1 et M_2 que l’on calcule avec une fonction de main.py à partir des résultats sont les mêmes pour chaque joueurs :

$$M_1 = 11$$

$$M_2 = 11$$

Ces nombres nous montrent que les deux joueurs, si ils jouent aléatoirement ont tendance à gagner au bout de 11 coups. Ce nombre étant proche de la moyenne aussi, on peut en déduire que les nombres de coups avant une victoire pour les deux joueurs sont également réparties autour de 11 coups.

Pour mieux mesurer la dispersion, on cherche aussi la variance σ² et l’écart-type $\sqrt{\sigma^2}$ des résultats.

$$\sigma^2{}_1 = \frac{1}{nb_{victoires joueur 1} - 1} \sum_{i=1}^{nb_{victoires joueur 1}} (n1_i - \overline{x_1})^2 = 13.415$$

$$\sigma^2{}_2 = \frac{1}{nb_{victoires joueur 2} - 1} \sum_{i=1}^{nb_{victoires joueur 2}} (n2_i - \overline{x_2})^2 = 13.279$$

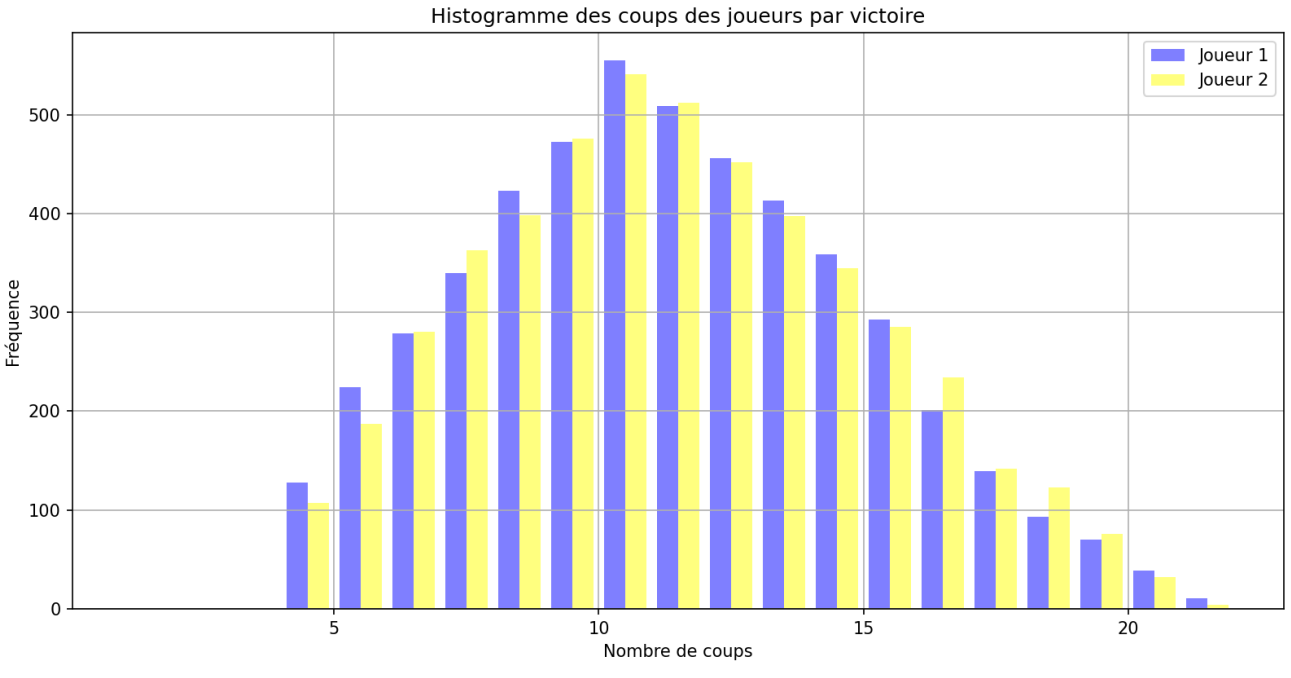
$$\sqrt{\sigma^2{}_1} = 3.66$$

$$\sqrt{\sigma^2{}_2} = 3.64$$

Pour le premier joueur, la variance est approximativement égale à 13.4 donc le nombre de coups du joueur 1 sont plutôt dispersées par rapport aux 11 coups moyens et la plupart des victoires sont généralement à moins de 3.66 coups de la moyenne, soit entre 7 (7.34) coups et 15 (14.66) coups.

En étudiant ces valeurs pour le joueur 2, avec une variance d’environ 13.3, il gagne généralement les parties entre 7 (7.36) et 15 (14.64) coups.

Après avoir noter les résultats obtenus dans un fichier .txt, nous avons créer un histogramme (avec la bibliothèque matplotlib) pour les deux ensembles de données qui résultent de l’expérience que voici :



On observe donc une très légère différence dans la distribution du nombre de coups avant une victoire entre les deux joueurs qui est normale; le premier joueur gagne le plus en moins de coups car joue le premier. Cette distribution semble suivre une loi normale de paramètre $\mu \approx 11$ et $\sigma^2 \approx 13$.

Probabilité d’une partie nulle

Une expérience qu’on peut réaliser pour trouver la probabilité d’une partie nulle lorsque les deux joueurs jouent aléatoirement peut être de simuler un très grand nombre de parties et de compter le nombre de parties qui se terminent sans aucun gagnant et de ce fait nulles.

On choisi aussi ici de simuler 10000 parties et sur ces 10000 parties, 33 d’entre elles sont nulles.

À partir de ce nombre, on peut facilement calculer une approximation de la probabilité pour l’évènement I **:** “La partie est nulle” avec m, le nombre de parties simulées nulles, et n, le nombre de parties simulées au total:

$$P(I) = \frac{m}{n} = \frac{33}{10000} \approx 0.003$$

On en déduit qu’il est peu probable d’obtenir une partie nulle en jouant aléatoirement sur un plateau de dimensions 7x6, avec une probabilité d’environ 0,4%.

Questions ouvertes

On veut donner une borne théorique à la probabilité d’une partie nulle quand les joueurs jouent aléatoirement. On suppose que le plateau est toujours celui de dimension 6x7, on définit donc un nombre maximal de coups qui sera de 42 et on fait l’hypothèse que les joueurs jouent jusqu’à ce que le plateau soit plein même après la victoire d’un des joueurs.

Avec ces hypothèses, le nombre de parties au total n_{total} est de 2^{42} et chaque joueurs et on peut approximer le nombre de façons de remplir le plateau sans victoire avec $n_{nul} = \binom{42}{21}$.

On note M, la borne supérieure théorique à la probabilité d’une partie nulle :

$$M = \frac{\binom{42}{21}}{2^{42}} \approx 0,1$$

On veut de plus donner une borne théorique du nombre de parties différentes qui peuvent être joués par deux joueurs jouant aléatoirement. L’expérience qui va nous servir à faire une approximation de ce nombre est de considérer chaque emplacement du plateau comme une variable qui peut prendre 3 valeurs différentes. Ces valeurs correspondantes au cas où le premier joueur a placé un jeton dans cet emplacement, au cas où c’est le second joueur qui a placé un jeton et le cas où l’emplacement est vide.

On note M', la borne supérieure théorique du nombre de partie différentes qui peuvent être jouées :

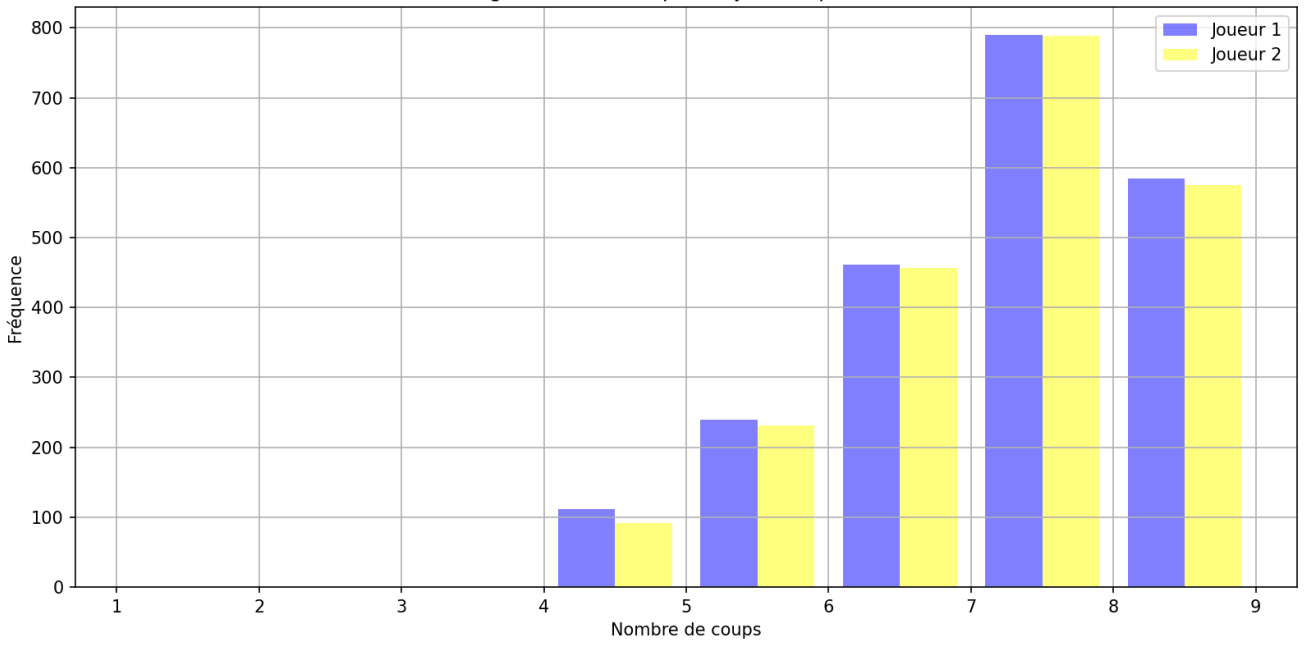
$$M' = 3^{42} \approx 1.09 \cdot 10^{20}$$

Enfin, nous faisons varier le nombre de colonnes et de lignes par rapport au plateau du jeu de Puissance 4 classique. On choisit un plateau de dimensions 4x4 et un autre de dimension 16x12.

Dans le cas d'un plateau 4x4, on obtient les valeurs suivantes : $\overline{x_1} = 6.7$, $\overline{x_2} = 6.7$, $M_1 = 7$, $M_2 = 7$, ${\sigma^2}_1 = 1.18$, ${\sigma^2}_2 = 1.24$, $\sqrt{{\sigma^2}_1} = 1.08$ et $\sqrt{{\sigma^2}_2} = 1.11$.

Les deux joueurs doivent faire en moyenne entre 6 et 7 coups avant de gagner une partie. Les valeurs sont aussi très serrées autour de la moyenne ce qui n'est pas choquant en considérant que l'on a seulement 16 cases sur le plateau.

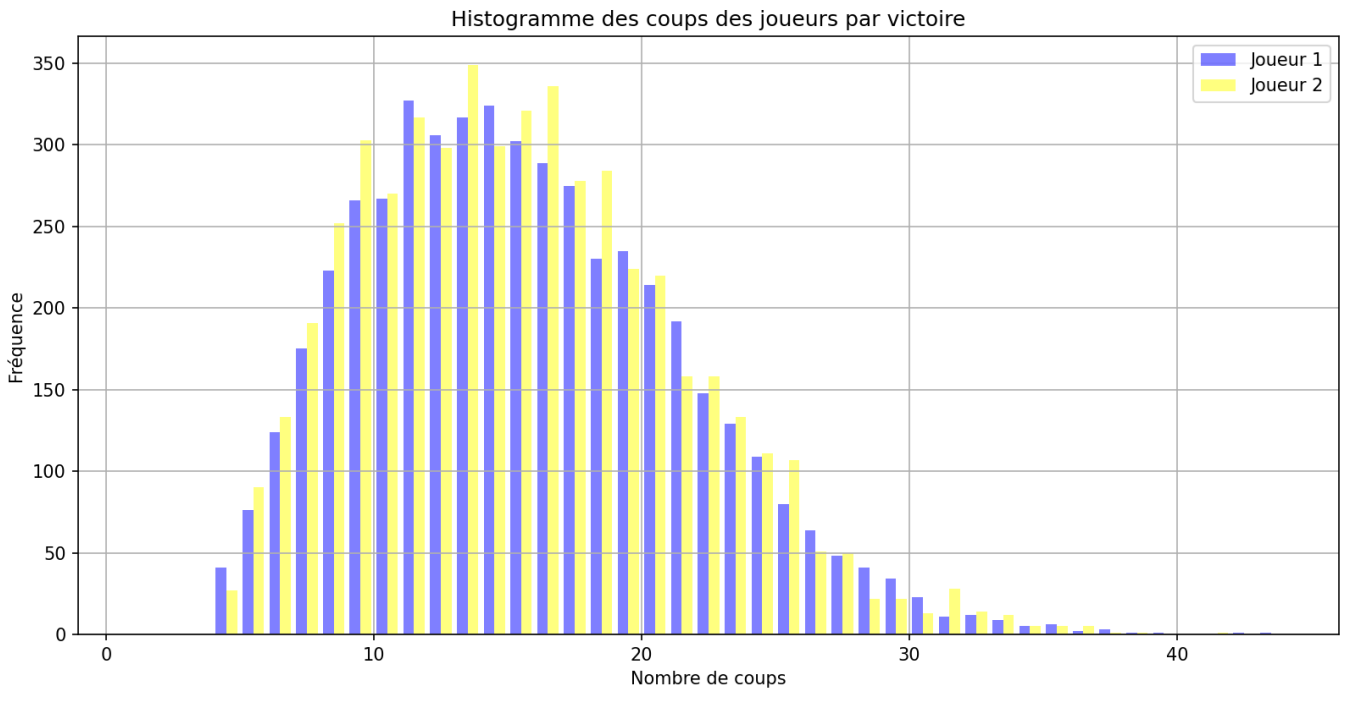
Cette configuration de taille (bien plus petit par rapport au plateau original) est aussi remarquable au niveau du nombre de parties nulles sur 10 000 itérations du jeu. Dans notre expérience, sur 10 000 parties jouées par les joueurs aléatoires, 5660 parties sont nulles. Dans ce cas, $P(I) = \frac{5660}{10000} \approx 0,57$.



Dans le cas du plateau 16x12, on obtient les valeurs suivantes : $\overline{x_1} = 15.2$, $\overline{x_2} = 15.0$, $M_1 = 15$, $M_2 = 15$, ${\sigma^2}_1 = 35.5$, ${\sigma^2}_2 = 34.7$, $\sqrt{{\sigma^2}_1} = 5.95$, et $\sqrt{{\sigma^2}_2} = 5.89$.

Sur ce plateau, il faudrait en moyenne 15 coups avant une victoire et le nombre de d'actions avant de gagner varie approximativement entre 9 et 21, ce qui est cohérent avec les résultats obtenus sur le plateau original.

Cette configuration est aussi remarquable lorsque l'on s'intéresse au nombre de parties nulles sur 10 000 itérations du jeu. Ici, le nombre de partie nulle sur 10 000 est égal à 0. On peut donc se poser la question de la possibilité d'avoir une partie nulle en jouant aléatoirement sur un plateau de dimension bien plus grande que sur le plateau original du jeu de Puissance 4.



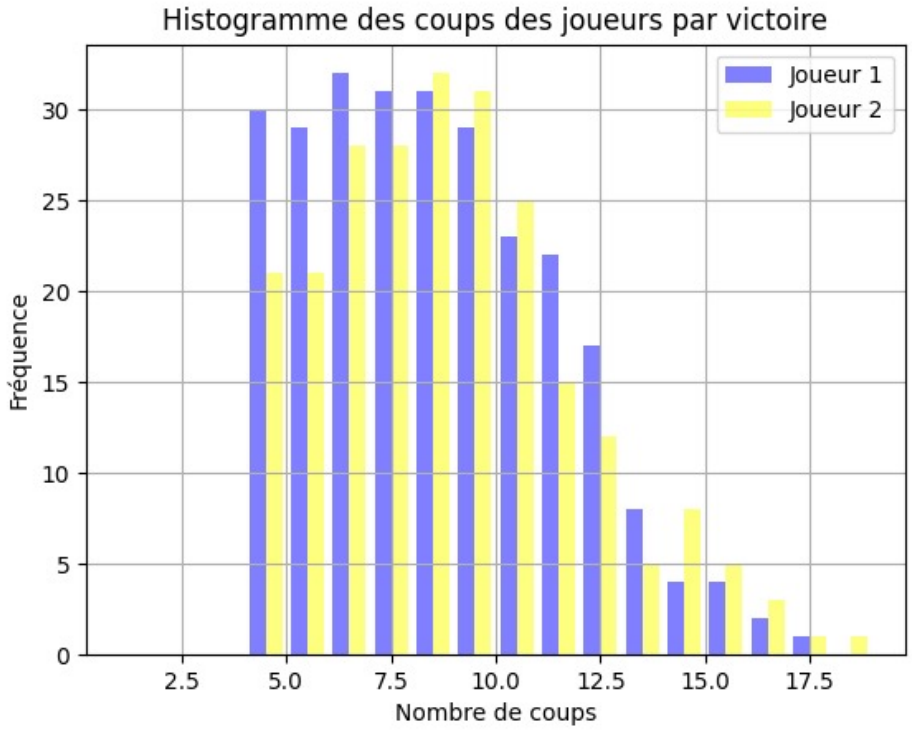
Algorithme de Monte-Carlo

Dans cette partie, on veut refaire une étude combinatoire du jeu (avec l'implémentation de Puissance 4 faite dans la partie précédente), en ajoutant un joueur qui utilise cette fois-ci l'algorithme de Monte-Carlo pour déterminer ses actions. Cet algorithme se base sur le choix d'une action en fonction de la récompense de cette action.

Test du joueur Monte-Carlo contre lui-même

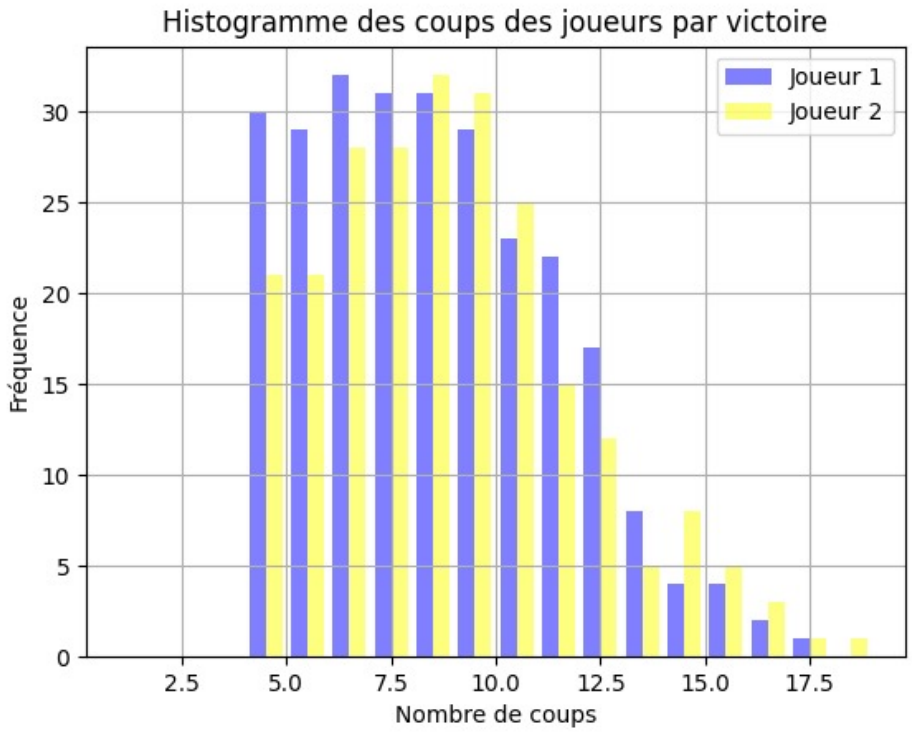
On note les deux joueurs Monte-Carlo joueur 1 et joueur 2. Chacun de ces joueurs fait 50 simulations de parties et on les fait jouer l'un contre l'autre 50 fois.

Les résultats des analyses sur la distribution du nombre de coups par victoire pour le joueur 1 et le joueur 2 sont les suivants: $\overline{x_1} = 7.9$, $\overline{x_2} = 8.5$, $M_1 = 7$, $M_2 = 7$, ${\sigma^2}_1 = 4.84$, ${\sigma^2}_2 = 9.62$, $\sqrt{{\sigma^2}_1} = 2.20$ et $\sqrt{{\sigma^2}_2} = 3.10$.



Les deux joueurs utilisant le même algorithme, les résultats sont comparables. On peut cependant noter que le joueur 1 gagne généralement en moins de coup, supposément car il joue en premier.

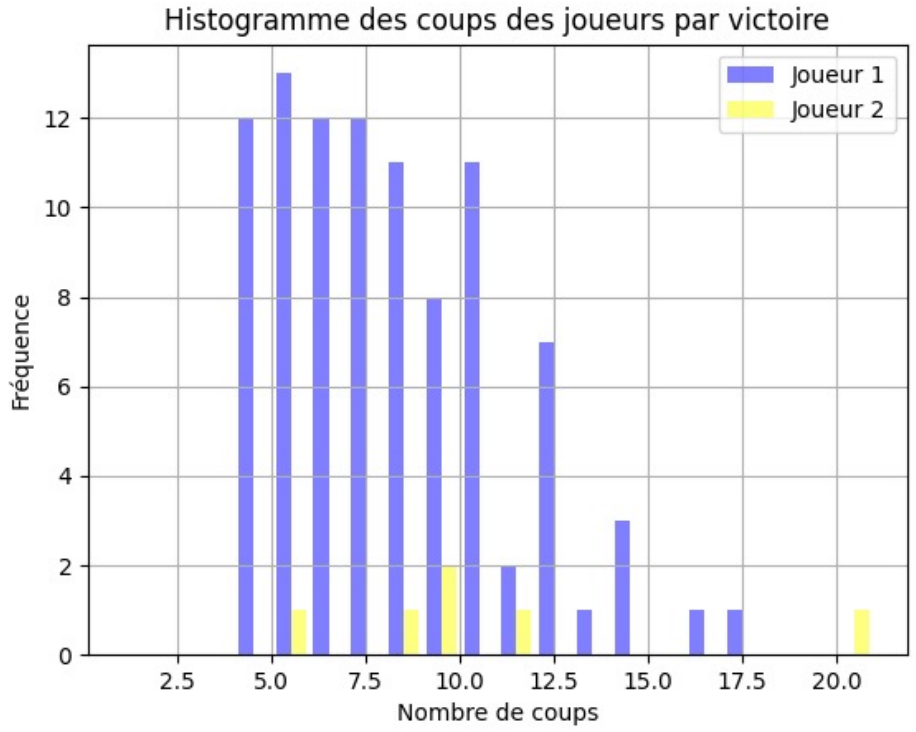
Sur ces 50 parties jouées, le joueur 1 gagne 27 fois contre 23 fois pour le joueur 2. On note aucune partie nulle. On peut supposer l'existence d'un match nul sur un plus grand nombre d'itérations de partie. On refait donc l'expérience avec 500 parties : $\overline{x_1} = 8$, $\overline{x_2} = 8.4$, $M_1 = 7$, $M_2 = 7$, ${\sigma^2}_1 = 8.54$, ${\sigma^2}_2 = 9.01$, $\sqrt{{\sigma^2}_1} = 2.92$ et $\sqrt{{\sigma^2}_2} = 3$. Sur ces 500 parties, on a 263 victoires pour le joueur 1, 236 victoires pour le joueur 2 et un match nul.



Test du joueur Monte-Carlo contre le joueur aléatoire

On note joueur 1 le joueur Monte-Carlo et, joueur 2, le joueur aléatoire. Le joueur Monte-Carlo fait 50 simulations de parties et on jouer les deux joueurs l'un contre l'autre au cours de 100 parties.

Les résultats des analyses sur la distribution du nombre de coups par victoire pour le joueur 1 et le joueur 2 sont les suivants: $\overline{x_1} = 7.8$, $\overline{x_2} = 10.3$, $M_1 = 7$, $M_2 = 7$, ${\sigma^2}_1 = 8.87$, ${\sigma^2}_2 = 26.27$, $\sqrt{{\sigma^2}_1} = 2.98$ et $\sqrt{{\sigma^2}_2} = 5.13$.



Le joueur 1 gagne bien plus de parties (94 contre 6, ces nombres varient). Il est normal que la stratégie de choix du meilleur coup batte dans la grande majorité des cas une stratégie de choix aléatoire. Comme vu précédemment, la probabilité qu'aucun des deux ne gagne est nulle.

Bandits-manchots

Dans cette partie, nous allons étudier le problème du dilemme exploration/exploitation dans le jeu des bandits manchots. Un joueur a le choix entre N leviers et chacun de ces levier est une action possible à un instant t qui a un rendement stationnaire dans le temps. Ces rendement suivent une loi de Bernouilli de paramètre $\mu_{i \in [1..N]}$.

On représente $i \in N = 10$ leviers donc les paramètres des rendements μ_i sont les suivants :

Levier i	μ_i
1	0.0
2	0.1
3	0.5
4	0.15
5	0.2
6	0.4
7	0.3
8	0.25
9	0.9
10	0.5

On choisi 10 leviers comme première expérience puis on choisira de ne garder que les 3 derniers leviers (i=8, i=9 et i=10) dans une seconde expérience. On fait $T = 100$ itérations de parties. On note a_t l'action jouée au temps t.

Algorithme aléatoire

Il va s'agir de notre baseline et cet algorithme va choisir a_t de manière uniforme. Parmis les algorithmes étudiés dans cette partie, cet algorithme utilisé par le joueur a le gain le plus faible lors de la première expérience :

$$G_T = \sum_{t=0}^{T} r_t = 39$$

Lors de la seconde expérience, il obtient un gain bien plus grand : $G_T = 67$. On remarque néanmoins qu'il a une chance sur 3 de tomber sur l'action au rendement le plus élevé.

Algorithme greedy

L'algorithme greedy repose purement sur l'exploitation des données acquise avec notre baseline (avec un nombre limité d'itérations consacrées à l'exploration). Le joueur choisi le levier qu'il estime avoir le rendement le plus grand : $a_t = argmax_{i \in [1..N]} \widehat{\mu_t^i}$. On choisit de faire 10 000 simulations avec 500 itérations consacrées à l'explorations.

$$G_T = 67$$

Algorithme Є-greedy

L'algorithme ϵ-greedy explore continuellement avec une probabilité ϵ de choisir au hasard parmi les actions possibles et une probabilité $1 - \epsilon$ d'appliquer l'algorithme greedy. On fait de même 10 000 simulations et on choisi $\epsilon = 0.5$.

$$G_T = 46$$

Algorithme UCB

L'action choisie par le joueur suite à l'utilisation de l'algorithme UCB est $a_t = argmax_{i \in [1..N]}(\widehat{\mu_t^i} + \sqrt{\frac{2log(t)}{N_t(i)}})$.

$$G_T = 81$$

Pour ces trois algorithmes, le levier choisi par le joueur dans les deux expériences est le levier 9.

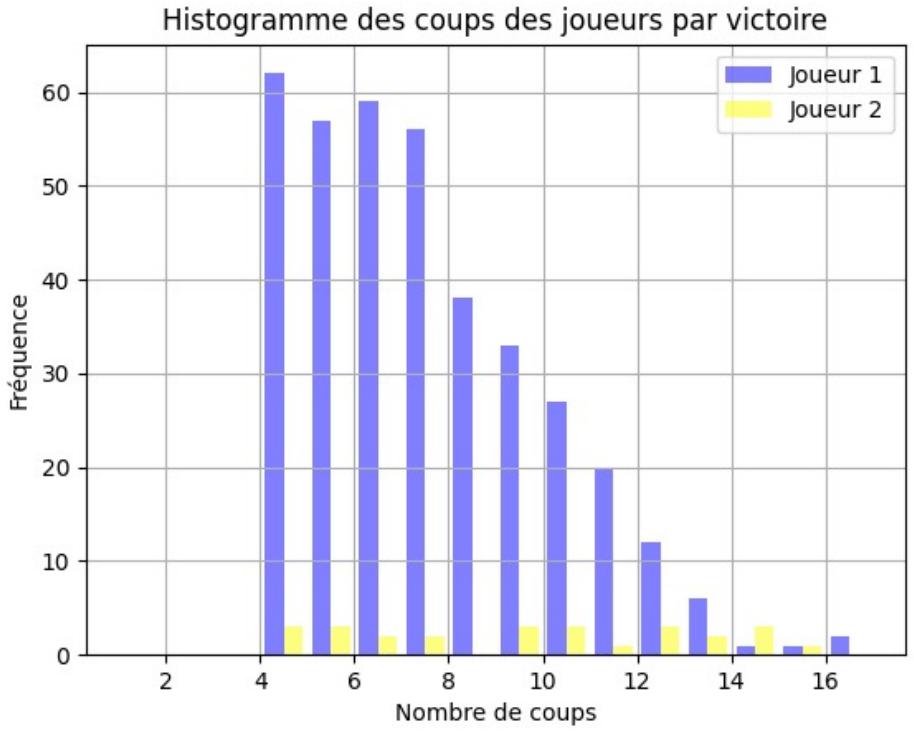
Arbre d'exploration et UCT

Dans cette dernière partie, on réitère l'étude faite dans les parties 1 et 2, cette fois-ci avec un joueur qui implémente un algorithme UCB adapté aux arbres de jeu. Le principe est d'utiliser l'algorithme UCB à chaque embranchement possible de l'arbre de la partie sur l'algorithme de Monte-Carlo dans le but d'équilibrer l'exploration et l'exploitation. On appelle ce joueur Joueur UCT et pour étudier les probabilités de victoire de ce joueur, on le fait jouer contre Joueur Aléatoire et Joueur Monte-Carlo.

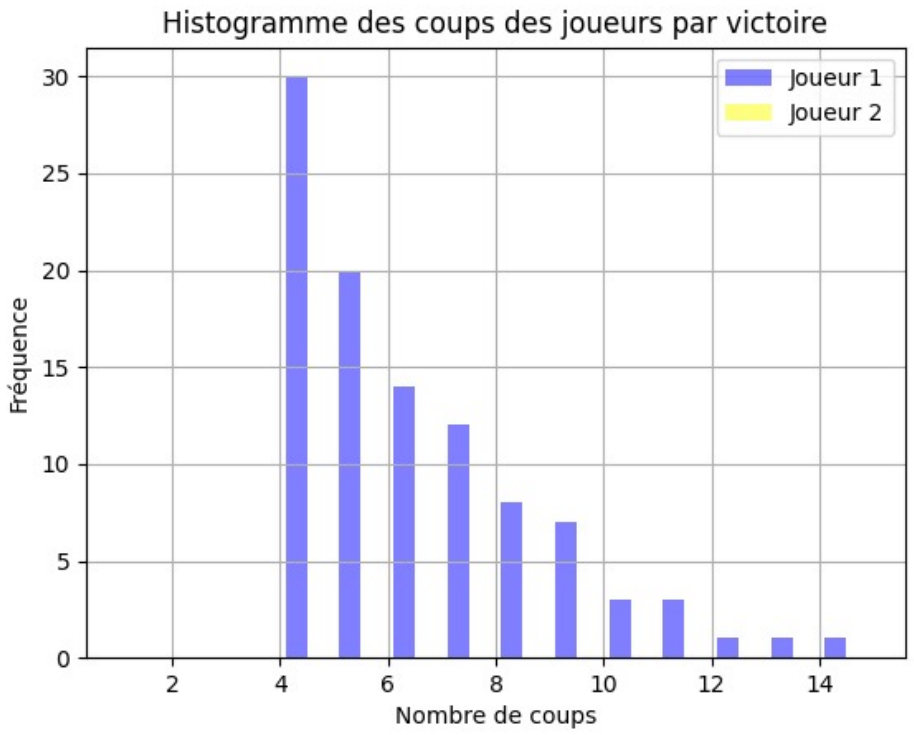
Joueur UCT vs Joueur Aléatoire

On note joueur 1 le joueur UCT et joueur 2 le joueur Aléatoire qu'on fait s'affronter sur 400 parties. Le joueur UCT simule 70 itérations de parties pour le choix de son action.

Les résultats des analyses sur la distribution du nombre de coups par victoire pour le joueur 1 et le joueur 2 sont les suivants: $\overline{x_1} = 7.1$, $\overline{x_2} = 9.2$, $M_1 = 7$, $M_2 = 9.5$, $\sigma^2{}_1 = 6.34$, $\sigma^2{}_2 = 12.98$, $\sqrt{\sigma^2{}_1} = 2.51$ et $\sqrt{\sigma^2{}_2} = 3.60$. Le joueur UCT gagne 374 fois contre 26 fois pour le joueur aléatoire.



On fait ensuite 100 parties et on change le nombre d'itérations de partie à 100 pour joueur UCT: $\overline{x_1} = 6.1$, $M_1 = 5.5$, $\sigma^2{}_1 = 5.21$ et $\sqrt{\sigma^2{}_1} = 2.28$. Joueur Aléatoire ne gagne pas dans ce cas et toutes les parties résultent en une victoire de Joueur UCT. On peut en déduire que dans le cas où le nombre d'itération est élevée et que le nombre de parties que l'on consacre à l'exploration suit proportionnellement cette tendance, UCT gagne facilement contre un algorithme aléatoire.

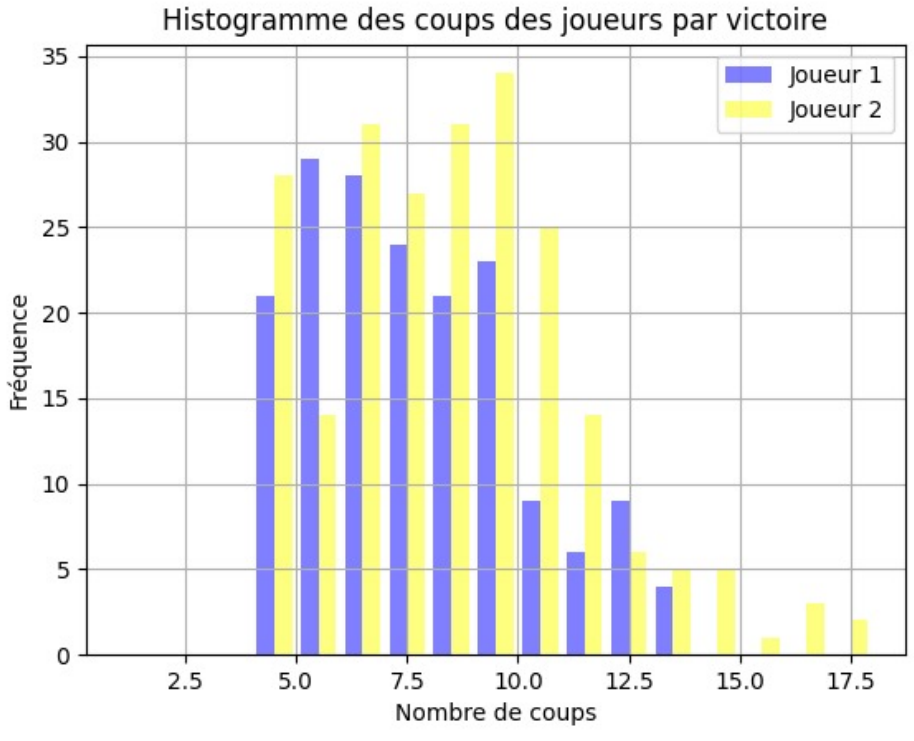


Joueur UCT vs Joueur Monte-Carlo

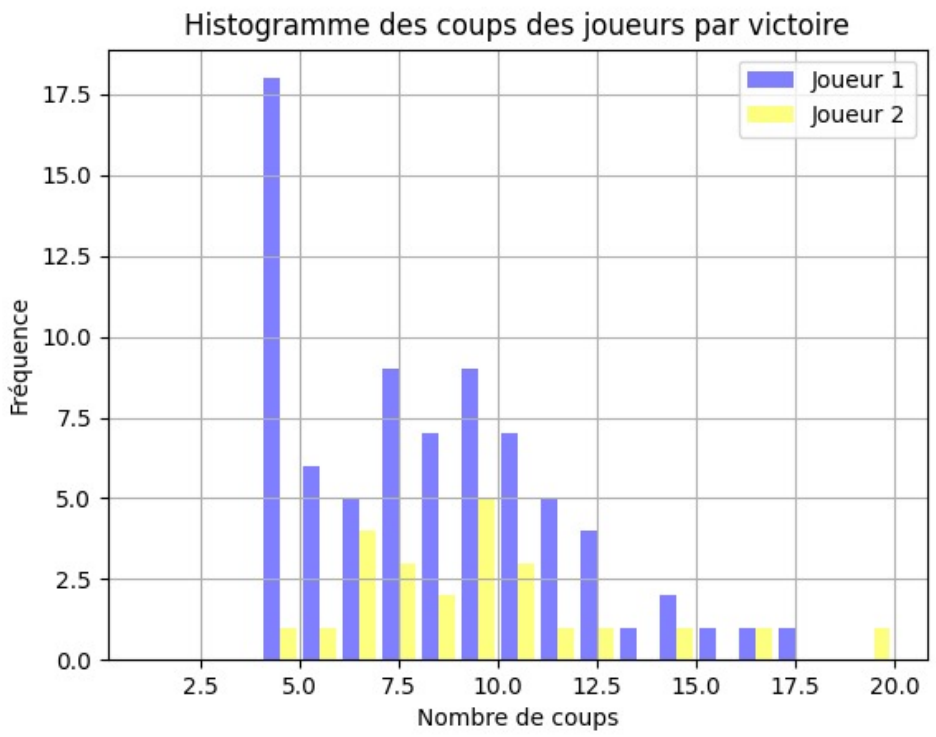
L'implémentation de UCT avec Monte-Carlo jouant contre un algorithme Monte-Carlo a une complexité temporelle trop élevée, on a donc décidé de réaliser une implémentation de UCT en prenant Joueur Aléatoire.

On note joueur 1 le joueur UCT et joueur 2 le joueur Monte-Carlo. Chacun de ces joueurs fait 50 simulations de parties et on les fait jouer l'un contre l'autre 400 fois.

Les résultats des analyses sur la distribution du nombre de coups par victoire pour le joueur 1 et le joueur 2 sont les suivants: $\overline{x_1} = 7.2$, $\overline{x_2} = 8.0$, $M_1 = 7$, $M_2 = 8$, $\sigma^2{}_1 = 5.68$, $\sigma^2{}_2 = 7.95$, $\sqrt{\sigma^2{}_1} = 2.38$ et $\sqrt{\sigma^2{}_2} = 2.82$. On obtient 226 victoires pour le joueur Monte-Carlo et 174 victoires pour le joueur UCT.



Pour 100 parties et 100 simulations (30 itérations pour l'exploration de UCT) avant choix de l'action des deux joueurs, on obtient les résultats suivants : $\overline{x_1} = 7.8$, $\overline{x_2} = 9.0$, $M_1 = 7.5$, $M_2 = 8$, $\sigma^2{}_1 = 10.74$, $\sigma^2{}_2 = 12.2é$, $\sqrt{\sigma^2{}_1} = 3.27$ et $\sqrt{\sigma^2{}_2} = 3.5$. UCT gagne 76 fois.



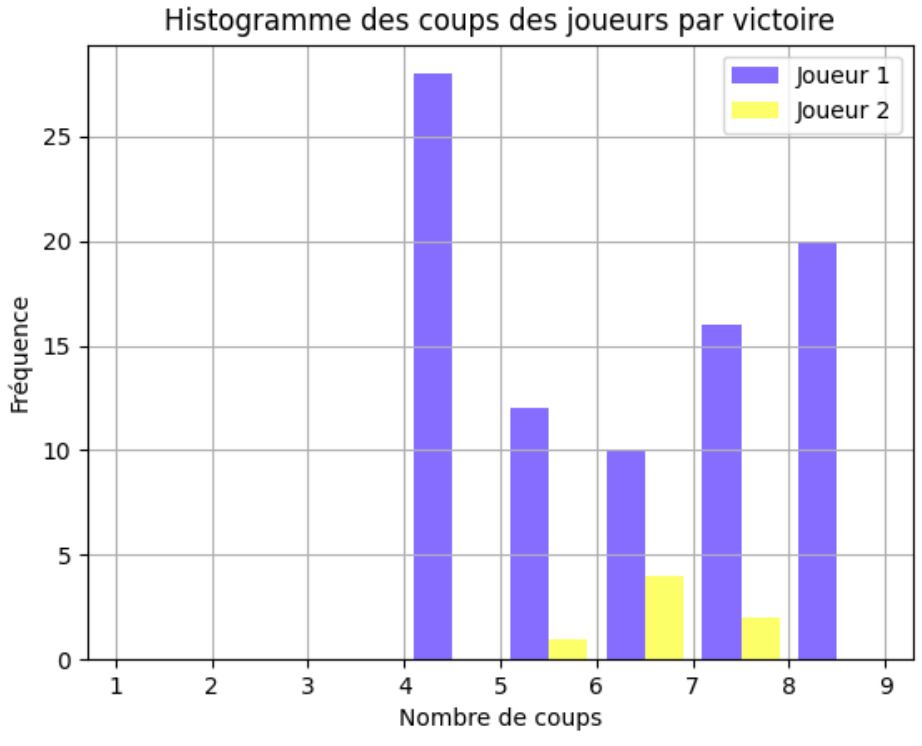
On remarque qu'avec un nombre d'itérations de simulations faible, l'algorithme Monte-Carlo est plus efficace. Mais lorsque l'on a un nombre de simulations bien plus élevé, UCT est très efficace dans l'obtention d'une victoire et le fait de minimiser le nombre de coups avant une victoire.

Encore une fois, pour ces algorithmes ayant une stratégie estiment des probabilités et visant à jouer le meilleur coup, les matchs nuls sont quasi-inexistants.

Plateau 4x4 sur 100 parties

- Joueur UCT vs Joueur Aléatoire

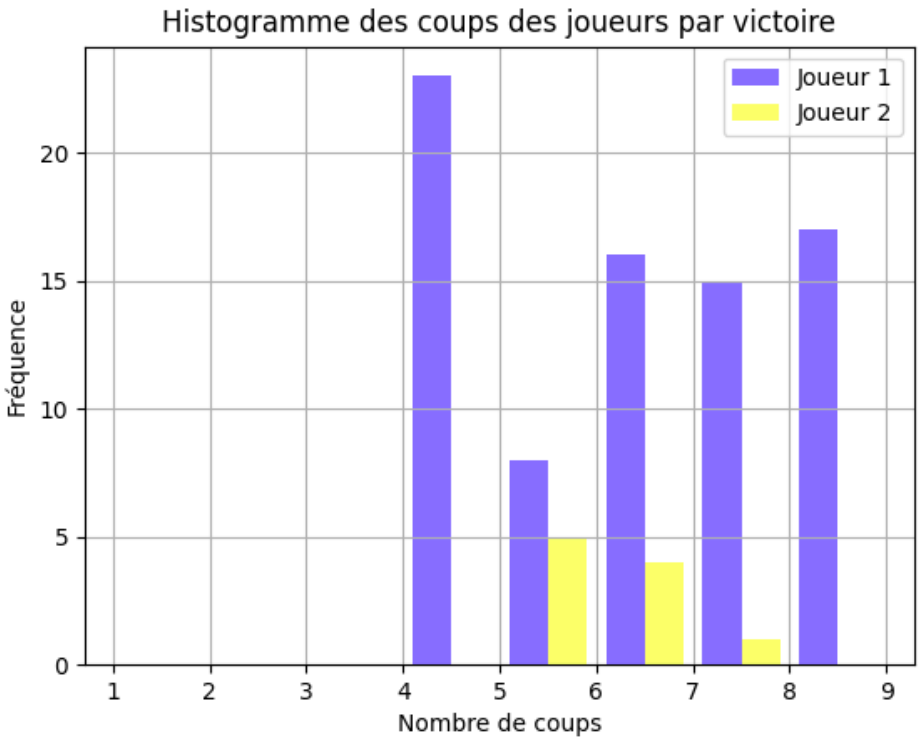
  Les résultats des analyses sur la distribution du nombre de coups par victoire pour le joueur 1 (70 simulations de parties) et le joueur 2 sont les suivants: $\overline{x_1} = 5.86$, $\overline{x_2} = 6.14$, $M_1 = 6$, $M_2 = 6$, $\sigma^2{}_1 = 2.57$, $\sigma^2{}_2 = 8.48$, $\sqrt{\sigma^2{}_1} = 1.60$ et $\sqrt{\sigma^2{}_2} = 0.69$.



- Joueur UCT vs Joueur Monte-Carlo

  Les deux joueurs font 70 simulations de parties.

  Les résultats des analyses sur la distribution du nombre de coups par victoire pour le joueur 1 et le joueur 2 sont les suivants: $\overline{x_1} = 5.9$, $\overline{x_2} = 5.6$, $M_1 = 7$, $M_2 = 7$, ${\sigma^2}_1 = 2.34$, ${\sigma^2}_2 = 0.48$, $\sqrt{{\sigma^2}_1} = 1.53$ et $\sqrt{{\sigma^2}_2} = 0.69$.



On remarque que pour un plateau plus petit, l'algorithme UCT mène au plus grand nombre de victoire dans les deux cas, même avect un nombre de simulations de parties faible.

Pour un plateau bien plus grand, la puissance de calcul et le temps nécéssaire sont bien trop grands pour qu'on puisse réaliser un match comprenant un joueur UCT.