Sorbonne Université

LU3IN005
2023/2024

Rapport 1 : Exploration/Exploitation et Puissance 4

MAMLOUK Haya [21107689]
RAMAHATAFANDRY Maeva [21104443]

Sommaire

Introduction

L'enjeu fondamental du dilemme entre exploration et exploitation traverse de multiples domaines de l'intelligence artificielle, notamment le machine learning. Cette question centrale se pose ainsi : lorsque nous sommes confrontés à plusieurs choix possibles, devrions-nous privilégier l'exploitation des connaissances existantes en optant pour l'action que nous estimons la plus rentable, ou plutôt persévérer dans l'exploration de nouvelles possibilités pour enrichir nos informations ? L'exploitation consiste à prendre des décisions basées sur les données disponibles, tandis que l'exploration vise à élargir notre champ de compréhension. Il est fréquent, surtout au début d'un processus, que l'on doive renoncer à l'option a priori la plus rentable afin d'optimiser les gains à long terme. Néanmoins, la question cruciale demeure : quand doit-on mettre fin à l'exploration, signifiant que nous avons recueilli suffisamment d'informations et que la poursuite de cette exploration n'apportera pas de nouvelles connaissances ?

L'objectif de notre projet est d'explorer divers algorithmes visant à résoudre ce dilemme de l'exploration par rapport à l'exploitation, en se concentrant spécifiquement sur l'application de ces concepts dans des intelligences artificielles dédiées aux jeux. Notre première étape consistera à étudier le jeu Puissance 4, un jeu caractérisé par une complexité relativement limitée.

Ce rapport est organisé en plusieurs sections, chacune correspondant à une phase spécifique du projet. Nous débuterons par une analyse du jeu Puissance 4, en explorant tant ses aspects théoriques que pratiques. Par la suite, nous nous pencherons sur la mise en œuvre d'un algorithme de Monte-Carlo dans le but de favoriser le gain. Les sections suivantes se concentreront sur l'étude des algorithmes classiques destinés à résoudre le dilemme exploration/exploitation, en prenant comme point de départ un contexte simplifié, le jeu des Bandits Manchots.

L'objectif ultime de ce rapport est d’évoquer les différents aspects du dilemme de l'exploration versus l'exploitation, dans le cadre des jeux et de l'intelligence artificielle.

I. Puissance 4

Le jeu Puissance 4 offre un terrain d'expérimentation pour explorer le dilemme fondamental de l'exploration par rapport à l'exploitation. Ce jeu de société classique est composé d'un plateau sur lequel les joueurs déposent des jetons pour former des alignements de quatre de leurs pièces. Puissance 4 est un jeu de stratégie combinatoire qui, malgré sa simplicité apparente, présente un certain nombre de défis pour les joueurs et les intelligences artificielles.

Cette partie de notre projet se concentrera sur la manière dont nous avons analysé et abordé Puissance 4 dans le cadre du dilemme exploration/exploitation.

A. Implémentation du jeu

Le jeu Puissance 4 est un jeu conçu pour deux joueurs qui se déroule sur un plateau composé de 7 colonnes et 6 lignes. Chaque joueur dispose d'un ensemble de 21 jetons. L'objectif du jeu est de parvenir à aligner quatre de ses propres jetons, que ce soit horizontalement, verticalement ou en diagonale, sur le plateau de jeu. Les joueurs jouent à tour de rôle, plaçant un jeton dans l'une des sept colonnes du plateau. Le jeton ainsi inséré descend jusqu'à atteindre la première case libre dans la colonne. La partie se termine dès qu'un joueur réussit à aligner quatre de ses jetons ou lorsque toutes les cases du plateau sont occupées.

Nous avons réalisé l'implémentation du jeu en utilisant une représentation basée sur un espace discret, sous la forme d'un tableau de 6 lignes et 7 colonnes. La structure de notre implémentation est la suivante : le jeu démarre, et les deux joueurs interagissent à tour de rôle en sélectionnant une colonne disponible où placer leur jeton. À chaque étape, c'est-à-dire à chaque tour de jeu, nous effectuons une vérification pour déterminer si l'un des joueurs a remporté la partie. Pour ce faire, nous examinons l'existence de combinaisons gagnantes composées de quatre jetons alignés, que ce soit horizontalement, verticalement ou en diagonale. Nous vérifions également si le plateau de jeu est complètement rempli, c'est-à-dire s'il ne reste plus aucune colonne disponible, ce qui résulterait en une partie nulle.

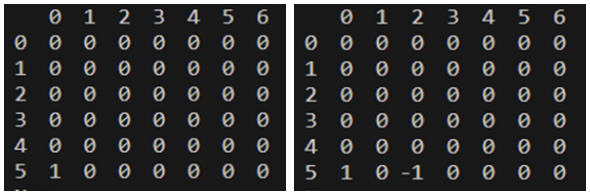


Figure 1 : Tour 1

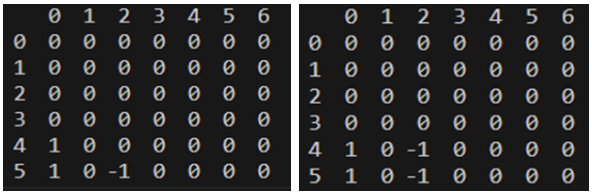


Figure 2 : Tour 2

```
   0  1  2  3  4  5  6
0  0  0 -1  0  0  0  0
1  0  0  1  1  0  0  0
2 -1  0 -1  1  0  0 -1
3 -1 -1  1  1  0  0  1
4  1 -1 -1 -1  0  0  1
5  1  1 -1 -1 -1  1  1
```

Figure 3 : Etat final du plateau

Voici un exemple du jeu Puissance 4. Dans cette représentation, une case avec la valeur 0 indique qu'elle est inoccupée, une case avec la valeur 1 représente un jeton du Joueur 1, et une case avec la valeur -1 représente un jeton du Joueur 2. La partie se termine avec la victoire du Joueur 2, qui a réussi à aligner 4 jetons en diagonale en suivant les positions (2,0), (3,1), (4,2), et (5,3).

B. Algorithme Aléatoire : Exploration et Modélisation Probabiliste

Lorsque nous examinons le jeu Puissance 4 à travers le prisme du dilemme exploration/exploitation, nous sommes immédiatement confrontés à des enjeux essentiels. Dans ce contexte, l'exploration impliquerait de choisir des coups qui, bien que moins évidents ou immédiatement moins rentables, pourraient révéler des informations pour les étapes à venir du jeu. Cela revient à établir un équilibre entre l'incertitude du résultat et le potentiel de découverte de stratégies gagnantes qui n'ont pas encore été jouées. En d'autres termes, il s'agit de jouer de manière aléatoire. Pour explorer ce concept plus en profondeur, il semble judicieux de mener N expériences afin d'étudier la distribution du nombre de coups nécessaires avant qu'une victoire ne soit obtenue lorsque les deux joueurs jouent de manière aléatoire. Voici les graphes représentant les résultats de nos expériences pour différentes valeurs de N.

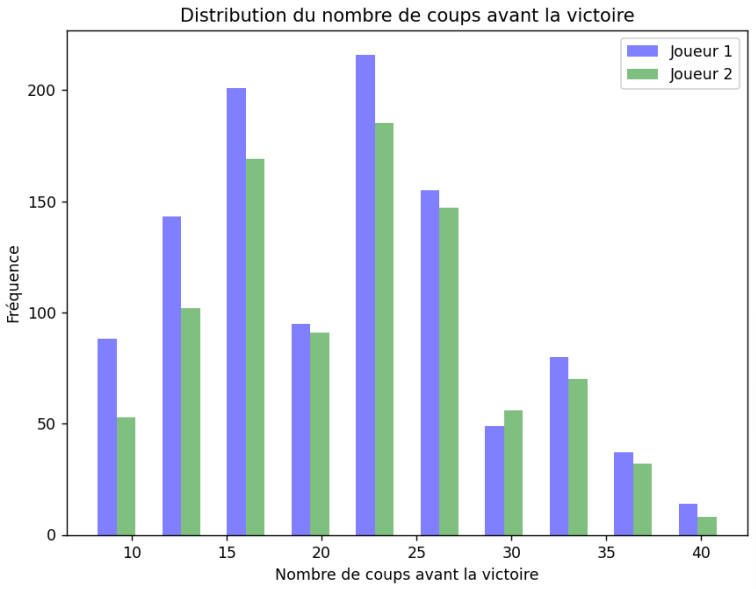


Figure 4 : Histogramme (N = 2000)

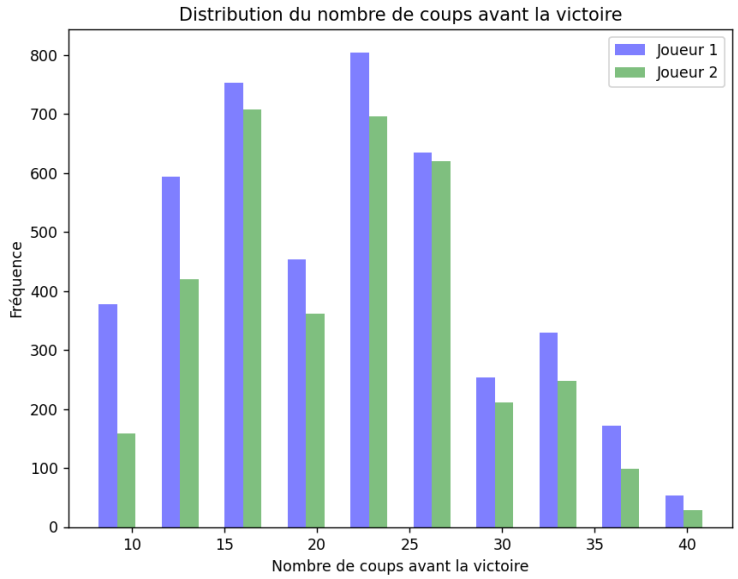


Figure 5 : Histogramme (N = 8000)

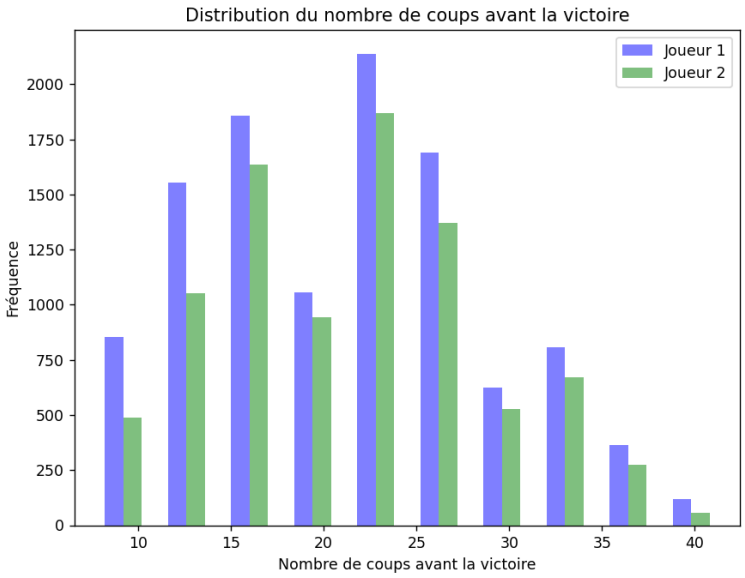


Figure 6 : Histogramme (N = 20 000)

Nous distinguons 3 différents événements :

A : “Le Joueur 1 gagne la partie”

B : “Le Joueur 2 gagne la partie”

C : “Partie nulle, aucun joueur ne gagne”

Et nous considérons :
C_A : nombre de coups moyens avant la victoire du joueur 1
C_B : nombre de coups moyens avant la victoire du joueur 2
C_0 : nombre de coups moyens avant une partie nulle

N	**2000**	**8000**	**20 000**
Nb victoires joueur 1	1078	4424	11 068
Nb victoires joueur 2	913	3548	8 892
Nb parties nulles	9	28	40
P(A)	0.539	0.553	0.5534
P(B)	0.4565	0.4435	0.4446
P(C)	0.0045	0.0035	0.002
C_A	21	21	21
C_B	23	22	22
C_0	42	42	42

Nous pouvons définir une variable aléatoire X qui représente le nombre de coups avant une victoire, que ce soit pour le joueur 1 (A) ou le joueur 2 (B). Les valeurs possibles de X seraient des nombres entiers positifs, compris entre 0 et 42 (avec 42 jetons, le plateau est plein et les joueurs ne disposent plus de jetons) . Etant donné l'expérience, on peut approximer les probabilités P(X = k) de cette variable aléatoire tel que:

- P(X < 7) = 0 : un joueur gagne si et seulement si le nombre de coups qu'il a effectué est supérieur à 4. En considérant les coups du joueur adverse et en supposant que c'est le premier joueur, 7 coups ont été effectués avant sa victoire.
- P(X >= 7) = P(A) U P(B) : si l'un des joueurs gagne, alors c'est bon.
- P(X = 42) = P(C) : Le plateau est complet donc 7x6 = 42 coups ont été joués.

En examinant nos données, nous pouvons remarquer que, en moyenne, le nombre de coups avant la victoire du joueur 1 (C_A) est relativement constant autour de 21 coups, et que le nombre de

coups avant la victoire du joueur 2 (C_B) présente une variation faible, généralement autour de 22 ou 23 coups.

Dans ce cas, l'observation d'une différence relativement faible entre C_A et C_B est normale, car il est raisonnable de s'attendre à ce que les deux joueurs aient des performances similaires lorsqu'ils jouent de manière aléatoire.

La légère différence entre les probabilités de victoire entre les deux joueurs est également cohérente avec cette observation. En moyenne, le joueur 1 gagne légèrement plus souvent, avec une probabilité d'environ 0,55, tandis que le joueur 2 gagne avec une probabilité d'environ 0,45.

En ce qui concerne l'identification d'une loi de probabilité usuelle pour ces données, il est en effet difficile de déterminer une distribution spécifique à partir des données limitées dont nous disposons. Cependant, il est intéressant de noter certaines caractéristiques de nos observations.

Tout d'abord, nous avons remarqué que le nombre de victoires élevé pour les deux joueurs se situe généralement autour de la moyenne, ce qui signifie que les parties sont généralement remportées après un nombre de coups compris entre 15 et 27 coups.

De plus, nous avons observé que la valeur moyenne du nombre de coups avant une victoire, que ce soit pour le joueur 1 ou le joueur 2, se rapproche de la valeur médiane de la variable aléatoire X. La médiane représente la valeur centrale d'un ensemble de données, ce qui signifie que la moitié des observations se situe au-dessus de cette valeur et l'autre moitié en dessous.

De plus, en examinant les valeurs de P(C) (la probabilité d'une partie nulle) et C_0 (le nombre moyen de coups avant une partie nulle), nous pouvons remarquer que l'occurrence d'une partie nulle est peu probable lorsque les deux joueurs jouent de manière aléatoire. De plus, il est intéressant de noter que la valeur de C_0 est constamment égale à 42. Cette constance de 42 est cohérente avec le fait qu'une partie nulle survient lorsque le plateau est entièrement rempli, c'est-à-dire lorsque les 42 emplacements sont occupés par les 21 jetons de chaque joueur. Cette observation souligne le lien entre le nombre de coups nécessaires pour remplir le plateau et l'occurrence d'une partie nulle, renforçant ainsi la probabilité relativement faible de cet événement lorsque les joueurs jouent de manière aléatoire.

Bien que nous ne puissions pas identifier une loi de distribution usuelle pour ces données en raison de leur nature aléatoire, nos observations soulignent l'importance de l'exploration dans le contexte du dilemme exploration/exploitation. Les valeurs moyennes et médianes proches indiquent une certaine stabilité dans les résultats malgré l'exploration aléatoire, ce qui suggère que les joueurs peuvent parfois découvrir des stratégies gagnantes non encore exploitées grâce à l'exploration.

C. Algorithme Monte-Carlo : Exploitation et Modélisation Probabiliste

Dans le contexte du dilemme exploration/exploitation, l'exploitation impliquerait une stratégie consistant à opter pour les coups les plus évidents et les plus favorables en fonction des informations disponibles jusqu'à présent. Cette approche vise à minimiser les risques tout en exploitant au maximum les connaissances actuelles afin d'augmenter les chances de victoire. Dans notre projet, nous avons choisi de mettre en œuvre une stratégie où l'un des joueurs utilise l'algorithme Monte-Carlo pour prendre ses décisions.

Concrètement, à chaque tour de jeu, le joueur qui utilise l'algorithme Monte-Carlo simule T parties à partir de l'état actuel du jeu et choisit le coup qui semble le plus favorable en fonction des résultats de ces simulations. Cette approche combine à la fois l'exploration, car elle permet de simuler différentes possibilités de jeu, et l'exploitation, car elle favorise les coups qui mènent théoriquement à la victoire.

Afin d'explorer l'équilibre entre l'exploration et l'exploitation dans ce contexte, nous avons réalisé plusieurs expériences en variant les paramètres tels que T (le nombre de simulations par tour), N (le nombre total de parties simulées), et la stratégie des joueurs (aléatoire ou basée sur Monte-Carlo). Ces variations nous permettent d'analyser comment ces facteurs influencent les performances des joueurs et de mieux comprendre comment trouver le bon compromis entre exploration et exploitation dans le jeu du Puissance 4.

Il convient de prendre en considération que l'algorithme que nous utilisons présente une complexité significative en termes de temps d'exécution et d'utilisation de la mémoire. En raison de ces facteurs, nous sommes contraints de travailler avec des valeurs relativement petites pour garantir des performances acceptables. Dans les exemples suivants, le joueur 1 joue aléatoirement, et le joueur 2 utilise l’algorithme de Monte-Carlo pour la prise de décision.

1. Joueur utilisant l’algorithme de Monte-Carlo

Pour N=1000, et un nombre de parties par tour T=10, nous avons collecté le nombre de coups nécessaires avant chaque victoire du joueur 2. Les valeurs collectées sont les suivantes : [0, 0, 0, 0, 0, 0, 0, 0, 606, 0, 81, 0, 26, 0, 36, 0, 83, 0, 31, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

En définissant X comme une variable aléatoire représentant le nombre de coups avant la victoire du joueur 2 avec Monte-Carlo, où X prend des valeurs dans {8, 10, 12, 14, 16, 18} (possible grâce aux répétitions de la simulation N=1000 et T=10 à plusieurs reprises), nous trouvons :

- $P(X=8) = 606/1000 \approx 0.606$

- P(X=10) = 81/1000 ≈ 0.081

- P(X=12) = 26/1000 ≈ 0.026

- P(X=14) = 36/1000 ≈ 0.036

- P(X=16) = 83/1000 ≈ 0.083

- P(X=18) = 31/1000 ≈ 0.031

L'espérance E(X) est d'environ 8.36, ce qui signifie qu' en moyenne, le joueur 2 utilisant l'algorithme Monte-Carlo gagne la partie avec un nombre de coups moyen d'environ 9.

La variance Var(X) est d'environ 9.51, ce qui indique une certaine dispersion des valeurs autour de la moyenne. La variance suggère que dans la majorité des cas nous utilisons moins de coups pour gagner comparé à la partie Aléatoire versus Aléatoire.

En analysant ces résultats, il est difficile d'attribuer une loi de probabilité usuelle spécifique à ces données. Cependant, nous pouvons conclure que l'algorithme Monte-Carlo produit des résultats efficaces comparés à la partie précédente (Aléatoire versus Aléatoire) , avec un nombre moyen de coups pour la victoire inférieur et une dispersion limitée autour de cette moyenne. Cela suggère une efficacité de l'algorithme dans la prise de décision dans le contexte du jeu Puissance 4.

2. Équilibre entre exploration et exploitation avec Monte-Carlo

Les histogrammes suivants représentent des simulations avec des paramètres N et T différents :

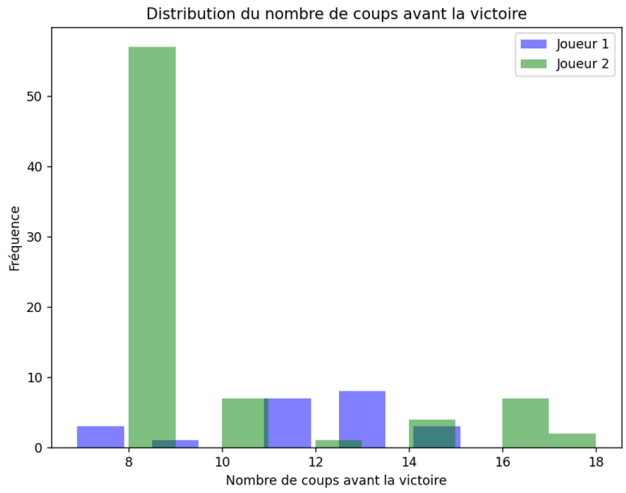


Figure 7 : Histogramme (N = 100, T = 10)

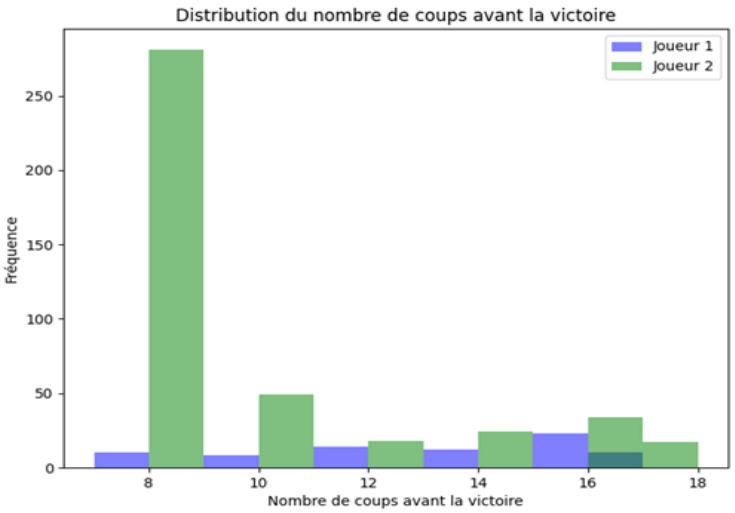


Figure 8 :Histogramme (N = 500, T = 10)

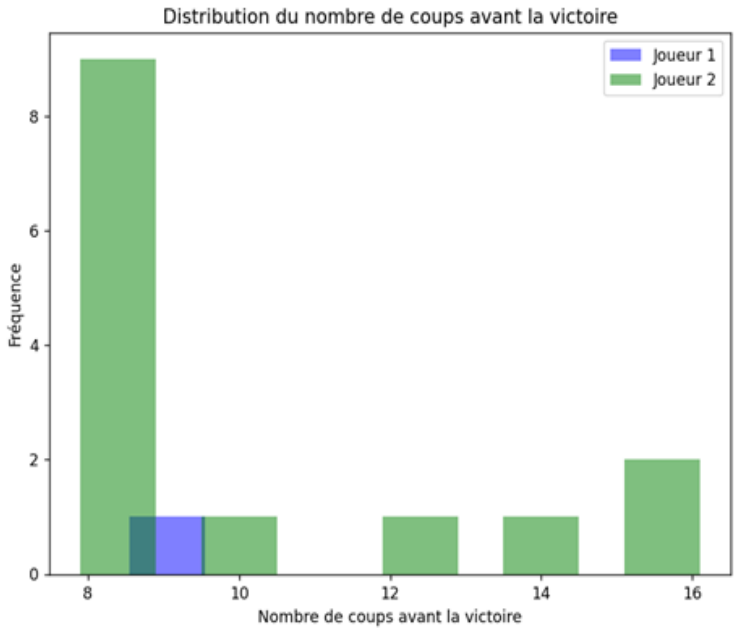


Figure 9 : Histogramme (N = 15, T = 20)

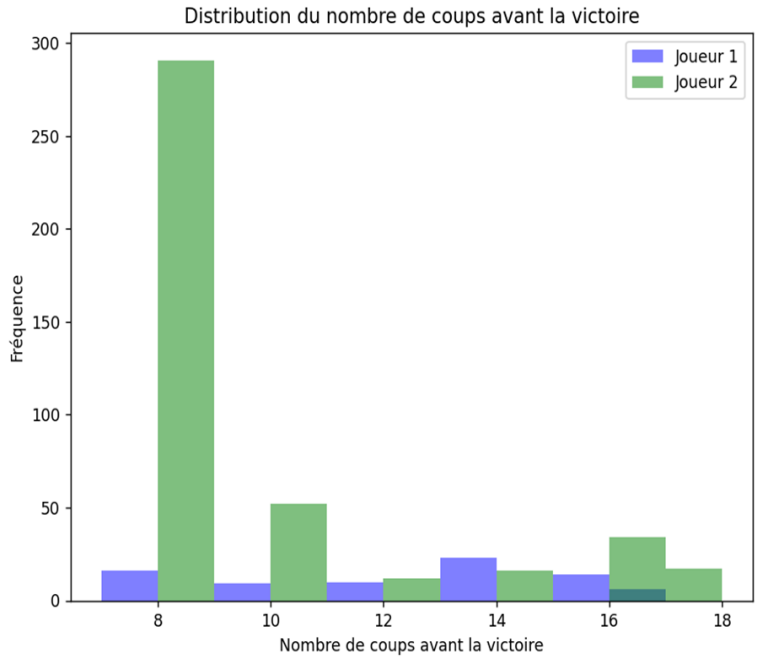


Figure 10 : Histogramme (N = 500, T = 100)

N	100	500	15	500
T	10	10	20	100
Nb victoires joueur 1	22	77	1	78
Nb victoires joueur 2	78	423	14	422
Nb parties nulles	0	0	0	0
P(A)	0.22	0.154	0.0666	0.156
P(B)	0.78	0.846	0.9333	0.844
P(C)	0.0	0.0	0.0	0.0
C_A	12	13	12	12
C_B	10	10	10	10
C_0	0	0	0	0

Les variables A, B, C, C_A, C_B et C_0 représentent les mêmes éléments que précédemment (voir pages 6/7).

En examinant les valeurs de P(A), il devient évident que l'algorithme Monte-Carlo assure un taux de réussite élevé, avec P(A) fluctuant entre 0,78 et 0,933, ce qui est considérablement supérieur aux valeurs obtenues lors des confrontations aléatoires.

En outre, il est important de noter que le nombre moyen de coups nécessaires avant la victoire du joueur utilisant l'algorithme Monte-Carlo reste constant à 10 coups. Cela met en évidence la stabilité et l'efficacité de l'algorithme Monte-Carlo ainsi que de la stratégie d'exploitation qui l'accompagne.

Cependant, il convient de souligner que nos données révèlent une tendance intéressante : lorsque T (représentant le niveau d'exploration) augmente par rapport à N, la probabilité de victoire (P(A)) augmente également (par exemple : N=15 et T=20). Par conséquent, il est judicieux de noter que pour maximiser l'efficacité de notre stratégie d'exploitation, une période d'exploration plus longue est nécessaire.

De plus, nous observons que la moyenne du nombre de coups avant la victoire du joueur 1, qui joue de manière aléatoire, est limitée à 13, ce qui est supérieur à la performance du joueur 2, mais nettement inférieur à celle d'une partie aléatoire contre aléatoire. Cela met en évidence le fait qu'un joueur utilisant l'algorithme Monte-Carlo parvient à réduire le nombre de coups nécessaires pour conclure une partie, ce qui est confirmé par l'absence de parties nulles dans nos simulations.

3. Joueurs Monte-Carlo vs Monte- Carlo

Nous avons décidé de tester des parties où les deux joueurs utilisent l'algorithme Monte-Carlo dans le jeu Puissance 4. En raison de la complexité de telles parties, nous avons dû utiliser des paramètres avec des valeurs plus faibles. Nous avons constaté que le nombre moyen de coups avant que le premier joueur gagne est d'environ 20 coups, tandis que celui du joueur 2 est d'environ 22 coups. Ces valeurs sont proches l'une de l'autre, mais significativement plus élevées que lorsqu'un seul joueur utilise Monte-Carlo, ce qui donne en moyenne environ 10 coups pour gagner.
Nous avons remarqué que lorsque les deux joueurs utilisent l'algorithme Monte-Carlo, le nombre moyen de coups avant que l'un des joueurs ne gagne est similaire à la moyenne de 22 coups lorsque les deux joueurs jouent de manière aléatoire. Cela indique que dans le contexte du jeu Puissance 4, l'algorithme Monte-Carlo n'apporte pas d'avantage significatif lorsqu'il est utilisé par les deux joueurs par rapport à une stratégie aléatoire.

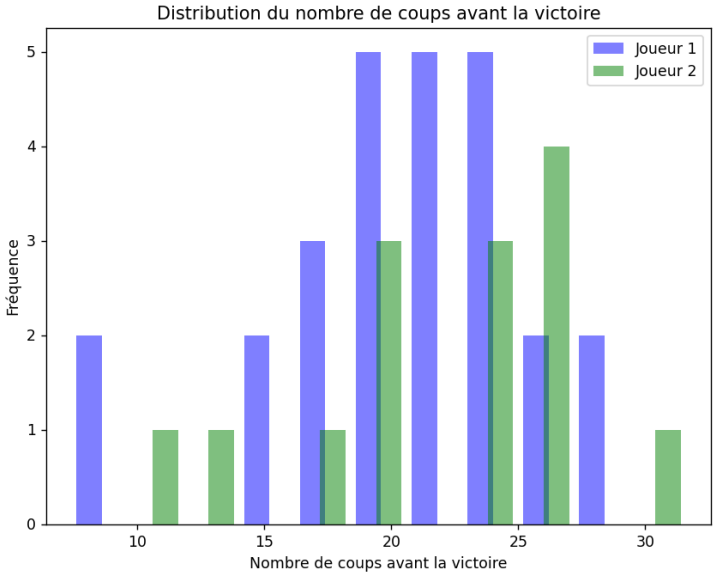


Figure 11 : Histogramme (N = 40, T = 20)

II. Bandits-manchots

Le jeu du bandits-manchots, également connu sous le nom de machines à sous, constitue un jeu de hasard intrinsèquement lié dilemme de l'exploration contre l'exploitation. Il consiste à placer une mise qui permet d'actionner un levier quelconque parmi les N leviers de la machine. Le levier choisi fera tourner des rouleaux, et selon la combinaison de rouleaux obtenue, une récompense sera attribuée au joueur.

A. Grandeurs observées

Dans le contexte de notre analyse, nous envisageons une machine à sous comprenant N leviers. Chaque levier est identifié par un numéro allant de 1 à N, et le levier activé à l'instant t sera représenté par a_t . Dans un souci de modélisation, nous supposons par la suite que la récompense associée à chaque levier i suit une distribution de Bernoulli tel que :

- μ^i représente le paramètre du succès qui est défini par l'obtention d'une récompense de 1
- 1 - μ^i représente l'échec, l'obtention d'une récompense de 0

La récompense obtenue à un temps t sera notée r_t : une variable aléatoire suivant une loi de Bernoulli de paramètre μ^{at}. Par ailleurs, les paramètres de cette expérience ne changent pas pendant la durée du jeu, c'est-à-dire que les probabilité μ^i de gagner sont constantes pour chaque levier.

Cette expérience est répétée T fois, l'objectif du joueur étant de maximiser son gain qui est défini par la somme des récompenses obtenues à chaque instant t du jeu: $G_T = \Sigma^T_{t=0} r_t$. De ce fait, il doit

identifier le levier au rendement le plus élevé dénoté $i^* = \text{argmax}_{i \in \{1, \ldots, N)} \mu^i$ et le rendement associé μ^*.

Le gain maximal est modélisé par la somme des récompenses obtenues après avoir joué avec le levier i*, T fois. Il s'écrit $G^*_T = \Sigma^T_{t=0} r^*_t$ tel que r^*_t. Ainsi, l'objectif du joueur peut se traduire par la minimisation de la différence entre le gain maximal espéré et le gain du joueur, que nous appelons regret. Il est défini par l'équation: $L_T = G^* - \Sigma^T_{t=0} r_t$, donc L_T vaut : $\Sigma^T_{t=0} (r_{t*} - r_t)$.

Notre démarche expérimentale vise à évaluer l'efficacité d'algorithmes dans la gestion équilibrée de l'exploration et de l'exploitation dans le contexte des bandits-manchots. Dans cette optique, notre première étape consistera à examiner un algorithme aléatoire qui effectue des actions de manière uniforme sur tous les leviers disponibles. Par la suite, nous élaborerons un algorithme glouton, également connu sous l'appellation "greedy", caractérisé par une phase d'exploration des solutions possibles sur un nombre déterminé d'itérations, suivie d'une exploitation du maximum obtenu pour le reste de l'expérience.

Le troisième algorithme à considérer combinera les caractéristiques des deux précédents. Il utilisera la fonction aléatoire avec une probabilité ϵ pour explorer et la fonction greedy avec une probabilité de $1 - \epsilon$ pour exploiter. Enfin, notre analyse portera sur le comportement d'un quatrième algorithme qui prend ses décisions en se basant sur une équation prédéfinie.

Pour observer les différentes variations entre les quatre algorithmes de l'expérience, nous avons réalisé quatre analyses différentes basées sur les caractéristiques de chaque algorithme que nous souhaitons mettre en valeur:

- Une comparaison des récompenses estimées et des récompenses réelles que nous avons préalablement fixées.
- Une confrontation du nombre de coups contre le nombre de victoires de chaque algorithme.
- L'évolution du regret de chaque algorithme en fonction du temps.
- L'étude du gain obtenu par chaque algorithme par rapport au gain maximal.

Cette approche méthodologique nous permettra d'évaluer la performance relative de ces différents algorithmes dans la maximisation des gains dans un contexte de bandits-manchots, en mettant en lumière les stratégies d'exploration et d'exploitation adoptées par chacun.

Par ailleurs, dans le cadre des analyses qui vont suivre, nous considérerons que N = 10 est le nombre de leviers de la machine.

B. Stratégie Aléatoire

L'algorithme que nous allons étudier dans cette partie consiste à choisir l'action a_t de façon uniforme un numéro de levier entre 0 et N-1. Autrement dit, il explore chaque possibilité de façon équitable pendant le nombre d'itérations T.

La stratégie aléatoire ne s'adapte pas aux variations de l'environnement et ne tient pas en compte des retours passés. Il ne fait aucun apprentissage sur les données et continue d'explorer les possibilités les unes après les autres.

1. Comparaison des récompenses estimées et des récompenses réelles

Cette première analyse nous a permis de constater que l'algorithme aléatoire est le plus performant pour déterminer les récompenses réelles de la façon la plus exacte possible. En effet, vu le nombre de tentatives qu'elle utilise équitablement sur tous les leviers, le rapport entre le nombre de coups total et le nombre de coups réussis est plus exact par rapport aux algorithmes qui n'explorent que certaines actions afin de maximiser le gain. Cependant, étant donné que cette stratégie ne fait aucun apprentissage, la découverte des probabilités liées à chaque levier ne peut pas être utilisée pour des actions plus réfléchies.

Le nombre de coups total affecte faiblement les estimations liées à l'algorithme aléatoire. Comme nous pouvons le constater sur les histogrammes suivants avec N = 10, les récompenses estimées sont très proches des récompenses réelles pour T = 100 mais aussi pour T = 1000. Ceci dit, nous pouvons tout de même constater que plus le nombre d'itérations est élevé, plus les approximations sont exactes.

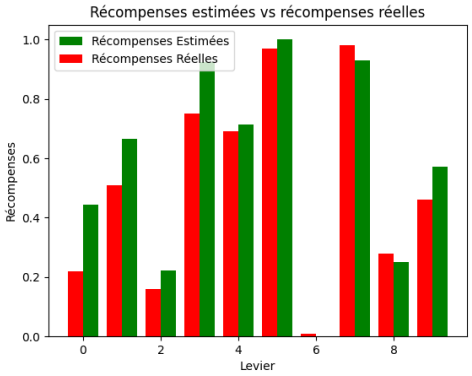


Figure 12 : Histogramme (T = 100)

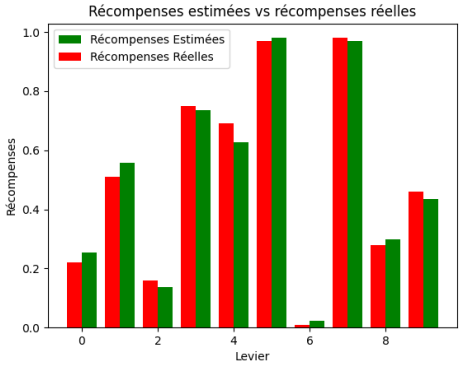


Figure 13 : Histogramme (T = 1000)

2. Comparaison du nombre de coups et du nombre de victoires

Afin de pouvoir étudier la distribution du nombre de coups, nous avons d'abord réalisé une expérience qui enregistre le nombre de victoires de l'algorithme aléatoire. Ensuite, pour

contextualiser ces victoires nous avons tracé les histogrammes du nombre de coups et du nombre de victoires avec différents tableaux de récompenses initialisées aléatoirement, avec deux différentes valeurs d'itérations N1 = 100 et N2 = 1000. Les histogrammes suivants représentent les résultats d'une de nos expériences.

Pour une récompense établie à **[0.86, 0.13, 0.65, 0.73, 0.57, 0.39, 0.08, 0.52, 0.34, 0.91]**:

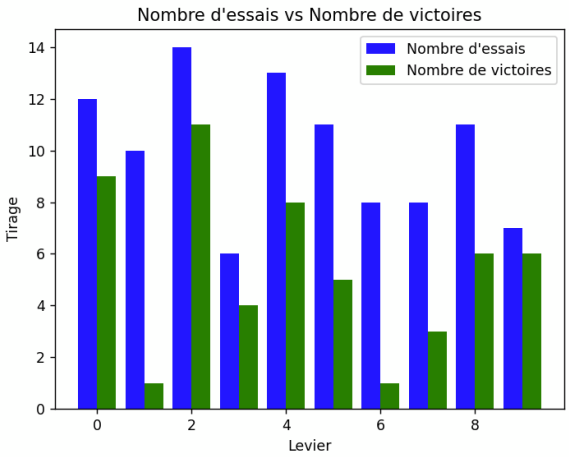


Figure 13 : Histogramme (T = 100)

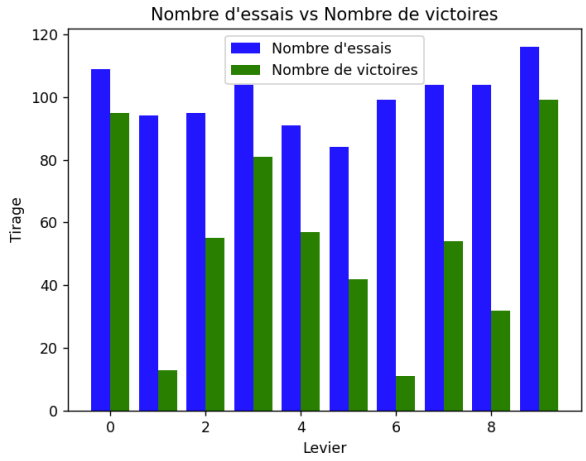


Figure 14 : Histogramme (T = 1000)

Suite à cette expérience, nous pouvons observer que sur différentes valeurs de récompenses, plus le nombre d'itération est élevé, plus le nombre de victoires de l'algorithme augmente car les coups qu'il effectue sont répartis plus équitablement sur tous les leviers. Ainsi, pour minimiser le regret à l'aide de l'algorithme aléatoire, il sera nécessaire d'utiliser un nombre d'itération très élevé, ce qui n'est pas forcément optimal comparé à potentiellement autres stratégies.

C. Stratégie Greedy

Comme expliqué précédemment, la stratégie greedy consiste à appliquer l'algorithme aléatoire pendant un nombre d'itérations fixé, notée ici T-greedy. Nous calculons ensuite l'indice de la récompense maximale calculée selon les données obtenues à partir des itérations de collection de données puis pour la suite des itérations, nous choisissons répétitivement l'indice calculé.

1. Cadre de l'expérience et formalisation des objectifs

Soit A l'événement défini comme suit : "On a tiré sur le levier avec la récompense maximale parmi les autres leviers." Nous supposions initialement que ce tirage est effectué par un algorithme aléatoire, suivant une distribution uniforme. Ainsi, la probabilité de succès de l'événement A est de 1/N, où N est le nombre de leviers.

En effectuant la première partie de l'algorithme greedy avec les premières itérations de collections de données, on réalise le tirage décrit précédemment T-greedy fois. La probabilité de tomber sur le rendement maximal est de 1/N. En caractérisant ce tirage par une variable aléatoire X, tel que X représente la récompense obtenue pour un levier actionné, X suit une loi binomiale de paramètre T-greedy et $\frac{1}{T-greedy}$, la probabilité de succès.

La recherche de la récompense k, la récompense maximale parmi les leviers possibles s'écrit:

$$P(x = k) = \binom{k}{tg} p^{k}(1 - p)^{tg - k}$$

Où *tg* représente T-greedy.

Notre objectif est de maximiser cette probabilité $P(x = k)$ afin de se rapprocher le plus possible du rendement maximal parmi les récompenses possibles.

2. Evaluation de l'exploitation et de l'exploration

Suite à un tirage successifs sur différents paramètres, nous avons pu déduire que le paramètre T-greedy est crucial pour le fonctionnement de l'algorithme. En effet, si le nombre d'itérations sur lequel nous explorons les récompenses possibles est trop bas, la probabilité $P(x = k)$ est moins élevée. Ce constat s'explique par le fait que l'algorithme greedy n'a pas eu le temps d'estimer les récompenses pour tous les leviers.

Dans le scénario où le nombre d'itérations est relativement restreint, l'algorithme peut converger vers un optimum local parmi les différentes possibilités de récompenses. Cela se produit lorsque l'algorithme, en raison d'un nombre limité d'itérations, opte pour l'exploitation d'une donnée qu'il estime être la meilleure, au détriment de l'exploration plus approfondie d'autres options. Il est crucial de souligner que cette tendance à l'exploitation prématurée peut conduire l'algorithme à négliger des options potentiellement plus avantageuses en raison de la sous-exploration de l'espace des solutions. Par conséquent, dans le contexte de la maximisation de la probabilité $P(x = k)$. Pour se rapprocher du rendement maximal, une attention particulière doit être accordée à l'équilibre entre exploitation et exploration afin d'éviter de converger vers des optima locaux sous-optimaux.

Lorsque le nombre d'itérations consacrées à l'exploration des rendements est élevé et que nous atteignons l'optimum global, l'algorithme démontre une performance accrue en termes d'exploitation, générant ainsi des gains optimaux. Ces conclusions sont étayées par les histogrammes résultant d'expériences exécutées sur une machine à 10 leviers, avec 3000 itérations et des rendements spécifiques [**0.22, 0.51, 0.16, 0.75, 0.69, 0.97, 0.01, 0.8, 0.28, 0.46]**, présentés ci-dessous.

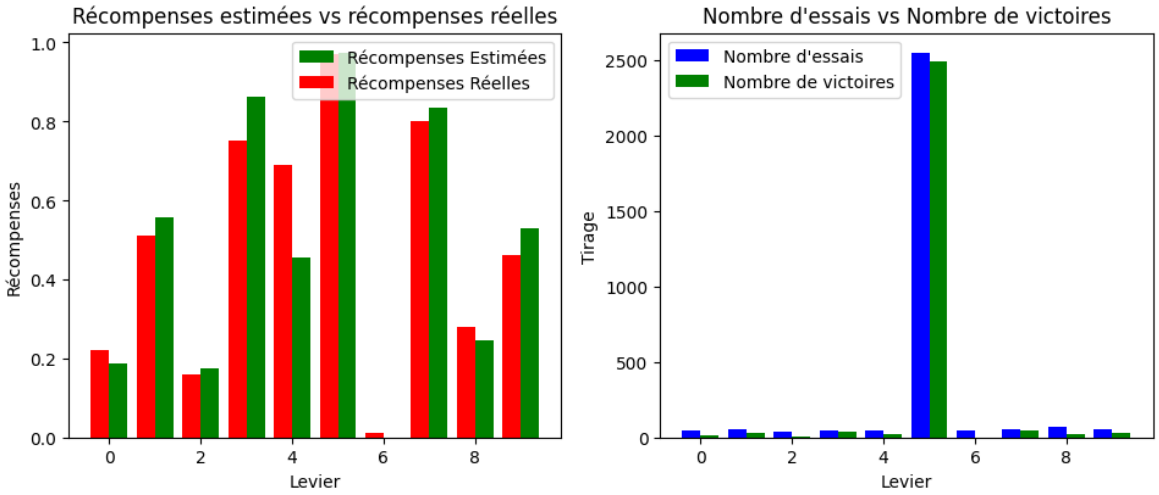


Figure 15 : Histogrammes (T-greedy = 500)

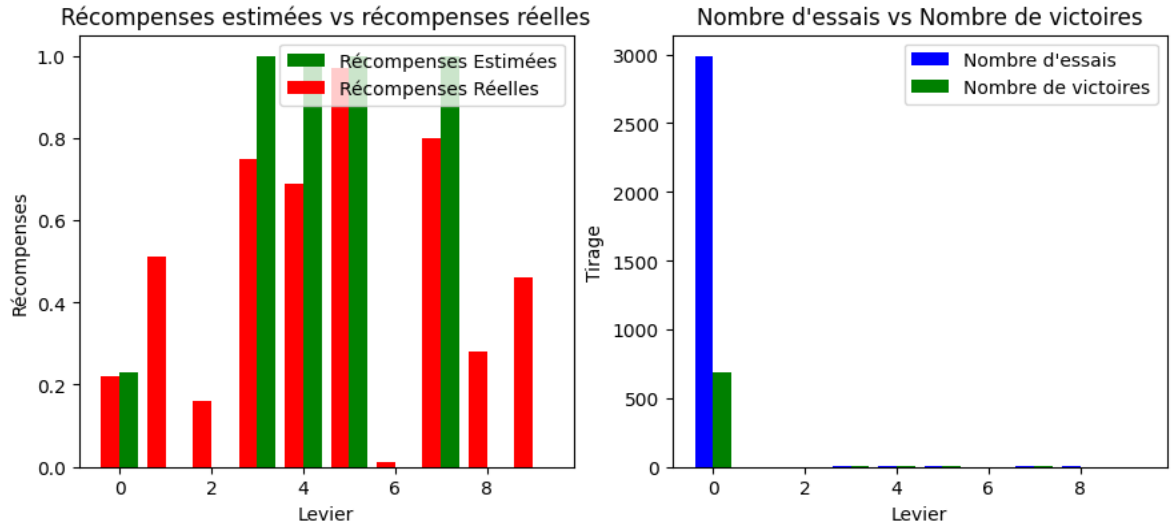


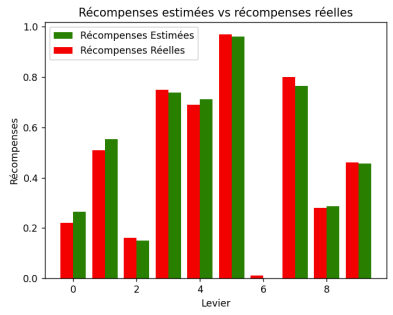
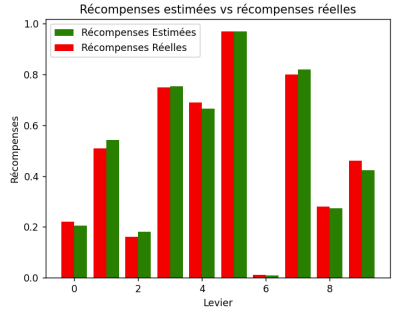
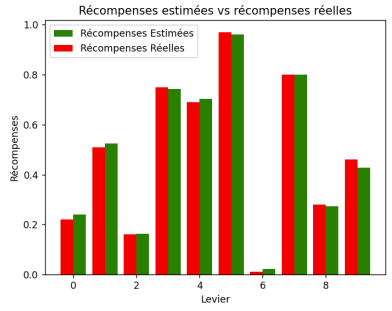
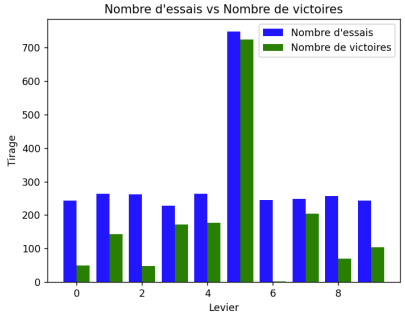
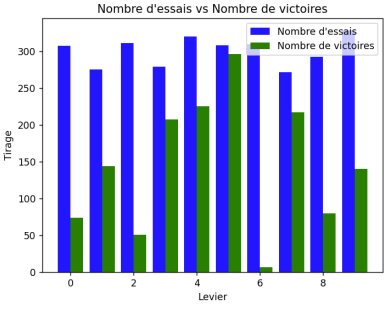
Figure 16 : Histogrammes (T-greedy = 10)

D. ε-Greedy

L'algorithme ε-greedy utilise les avantages d'exploration de l'algorithme aléatoire et ceux d'exploitation de la stratégie greedy pour équilibrer sa stratégie de jeu. Il est paramétré par ε qui est la probabilité d'exploration, et donc d'utiliser l'algorithme aléatoire. Autrement dit, 1 - ε représente la probabilité d'utiliser la stratégie greedy et de favoriser l'exploitation. L'expérience menée pour analyser le comportement de cette fonction est centrée sur le changement de la valeur de ε. Dans ce qui suit, nous utiliserons un nombre d'itération T = 3000, 10 leviers, un tableau de rendements initialisé à **[0.22, 0.51, 0.16, 0.75, 0.69, 0.97, 0.01, 0.8, 0.28, 0.46]**. Par ailleurs, afin d'optimiser le fonctionnement de l'algorithme greedy dans le cas échéant, nous utiliserons un nombre d'itérations 1000 pour T-greedy.

Notre expérimentation a été conduite en utilisant trois valeurs distinctes pour le paramètre εε : 0.2, 0.5 et 0.8.
Les résultats obtenus indiquent une amélioration globale de l'apprentissage des récompenses par rapport à l'algorithme greedy, malgré une probabilité d'exploitation supérieure à la probabilité d'exploration. Il est essentiel de noter que les estimations résultent de la combinaison des itérations employées dans les appels à la stratégie Greedy et celles utilisées lorsque ε généré correspond à l'appel de la fonction aléatoire. Cette observation suggère que le processus d'apprentissage orchestré par l'algorithme demeure dynamique et ne connaît pas de cessation.

De manière générale, il est bien établi que plus la valeur de ε est élevée, plus l'algorithme tend à privilégier l'exploration. Le choix de ce paramètre doit être méticuleusement ajusté en fonction des exigences spécifiques du problème. Une valeur de ε=0,5 s'avère particulièrement judicieuse, garantissant simultanément un équilibre entre l'exploitation et l'exploration. Ces constats sont illustrés par le tableau suivant:

ε=0,2	ε=0,5	ε=0,8
		
		

E. Stratégie UCB

L'algorithme UCB (Upper Confidence Bound) a été spécifiquement conçu pour équilibrer l'exploration et l'exploitation lors de la sélection des actions parmi les possibilités de leviers possibles. Le paramètre utilisé dans le cadre du projet est basée sur l'estimation de la

moyenne des récompenses pour chaque levier μ^i et un terme qui encourage l'exploration: $\sqrt{\frac{2\,log(t)}{Nt(i)}}$.

Dans le cadre de cette dernière expérience, nous avons utilisé 3000 itérations (T), 10 leviers (N). Nous faisons ensuite varier le nombre d'itérations utilisées pour explorer les récompenses du jeu, que l'on notera T_{UCB}, avant d'appliquer l'algorithme. Nous pouvons observer les regrets suivants:

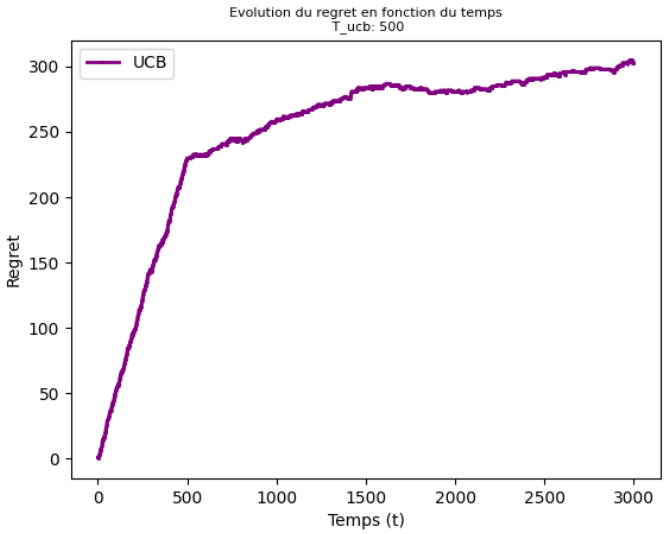


Figure 17 : Graphique (T_{UCB}= 500)

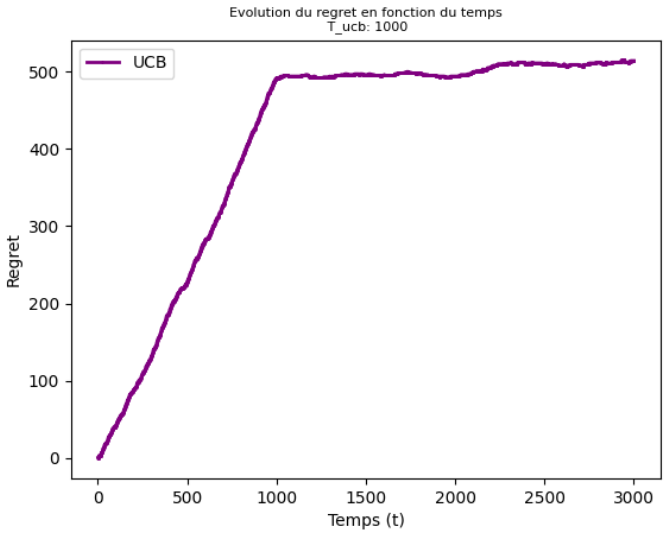


Figure 18 : Graphique (T_{UCB}= 1000)

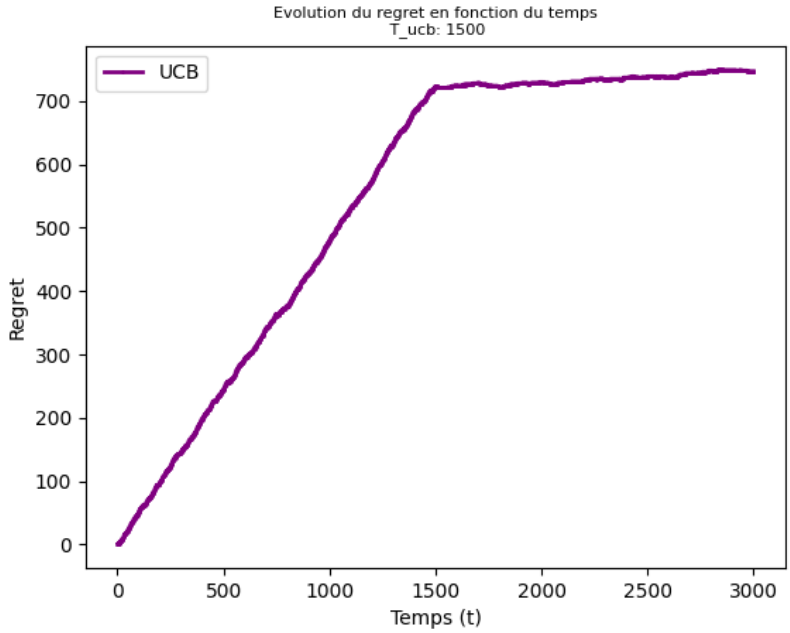


Figure 19 : Graphique (T_{UCB}= 1500)

Les résultats indiquent que l'algorithme UCB parvient à équilibrer l'exploration initiale avec une transition vers une exploitation plus optimale à mesure que T_{UCB} augmente. Cette transition est cruciale pour maximiser les gains, car elle permet une meilleure estimation des récompenses associées à chaque levier. Nous remarquons que T_{UCB} est l'itération durant laquelle la transition se produit.

L'algorithme UCB se positionne comme une stratégie prometteuse pour les problèmes d'exploration-exploitation en optimisant le compromis entre exploration initiale et exploitation ultérieure. Son adaptation dynamique en fonction du nombre d'itérations d'exploration le rend particulièrement robuste pour divers scénarios, soulignant son potentiel dans des environnements complexes et évolutifs.

F. Comparaison des regrets

Dans cette dernière expérience, nous avons tracé les fonctions qui représentent l'évolution du gain au cours des itérations. Il est défini par l'équation: $L_T = G^* - \Sigma^T_{t=0} r_t$, donc L_T vaut : $\Sigma^T_{t=0} (r_{t*} - r_t)$.
Nous avons appliqué cette expérience avec 3000 itérations et 10 leviers, puis avec T-greedy initialisé à 1000 itérations, ε à 0.5 et le nombre d'itérations d'exploration d'UCB à 1500. Nous obtenons le graphe suivant:

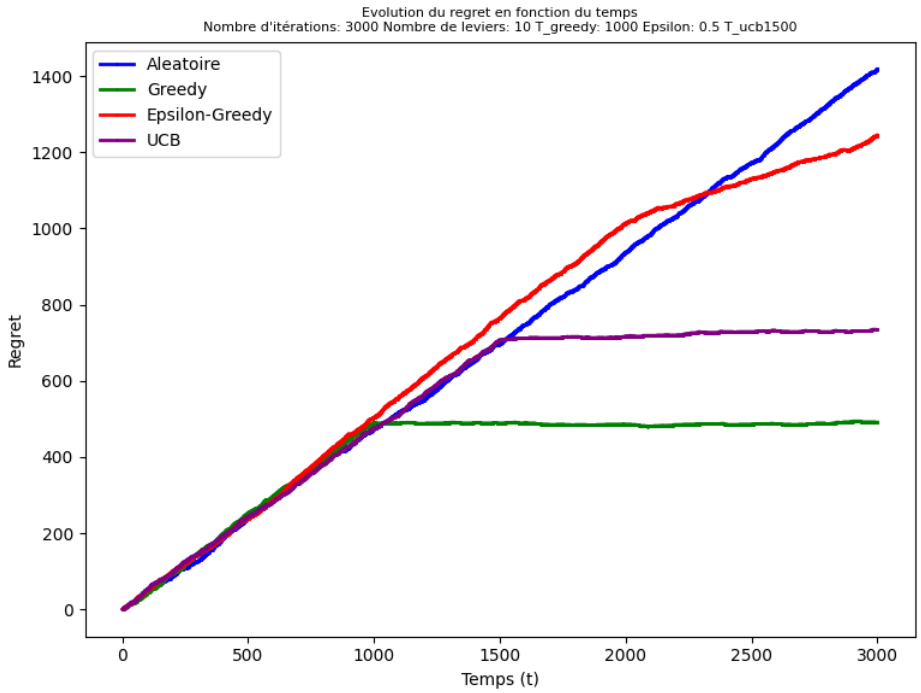


Figure 20 : Graphique (T-greedy = 1000, ε = 0.5, T_{UCB}= 1500)

Nous observons que le regret de la fonction aléatoire augmente progressivement, et sa pente reste constante. Cela s'explique par une sous-exploitation des données collectées et par la recherche constante d'une nouvelle action, ce qui limite l'obtention du gain maximal.

Pour l'algorithme greedy, nous notons un changement de pente lors de la millième itération, avec une inclinaison devenue plus faible. Ce changement est attribuable à l'arrêt des itérations d'exploration et à la transition vers l'exploitation du maximum trouvé. Ainsi, cette fonction présente le regret le plus bas.

La stratégie ε-greedy est moins prévisible que les autres fonctions. Nous observons un changement d'inclinaison lors des dernières itérations. Étant donné que ε est initialisé à 0.5, il module simultanément le comportement des algorithmes greedy et aléatoires.

L'algorithme UCB reproduit de manière optimale les caractéristiques d'exploitation de l'algorithme aléatoire et celles d'exploration de l'algorithme greedy. Nous constatons que les inclinaisons des algorithmes aléatoire et UCB sont similaires pendant les itérations de découverte de UCB. Cependant, à partir de l'itération 1500, l'algorithme UCB adopte un comportement similaire à celui de l'algorithme greedy, privilégiant l'exploitation.

Conclusion

D’après les analyses que nous avons effectuées sur les deux terrains d’expérimentation du jeu de puissance 4 et du bandits-manchots, l’étape d’exploration est aussi importante que celle d’exploitation. En effet, une bonne connaissance des éléments du jeu est importante pour pouvoir déterminer les meilleures combinaisons gagnantes, qui seront ensuite exploitées plusieurs fois. L’enjeu réside dans le choix du moment de transition entre les deux méthodes.

On a pu constater dans le jeu de Puissance 4 que par le biais de l'analyse de l'algorithme aléatoire met en lumière l'importance de l'exploration dans la découverte de stratégies gagnantes. Les données montrent une stabilité dans les résultats malgré l'exploration aléatoire, indiquant que les joueurs peuvent découvrir des stratégies efficaces par le biais de l'exploration. Bien que la distribution exacte des résultats soit difficile à déterminer, certaines caractéristiques, telles que la stabilité autour de la moyenne, sont notables. En introduisant l'algorithme Monte-Carlo, axé sur l'exploitation, nous observons une efficacité notable dans la prise de décision. L'algorithme parvient à réduire le nombre de coups nécessaires pour la victoire, avec une stabilité dans les performances. Les simulations mettent en évidence l'importance de trouver un équilibre entre exploration et exploitation, soulignant que des périodes d'exploration plus longues peuvent améliorer l'efficacité de la stratégie d'exploitation.

Ensuite, notre exploration des stratégies pour le jeu de bandits-manchots a révélé des dynamiques cruciales dans la gestion de l'exploration et de l'exploitation. Les itérations clés où les algorithmes basculent entre ces deux phases ont été précisément quantifiées, mettant en lumière des caractéristiques importantes de chaque algorithme. Pour l'algorithme Greedy, le nombre d'itérations d'exploration suit une distribution binomiale, soulignant son adaptation à la dynamique du jeu. L'ε-Greedy, avec sa probabilité ε, offre une flexibilité d'ajustement entre exploration et exploitation. Enfin, l'UCB, avec son approche dynamique basée sur l'estimation de la moyenne des récompenses, démontre une adaptabilité particulière. Chaque algorithme présente des avantages spécifiques, soulignant l'importance du choix selon les caractéristiques du problème. En somme, cette étude fournit des perspectives essentielles pour comprendre les mécanismes de prise de décision dans les jeux de hasard, avec des implications potentielles pour l'apprentissage automatique et l'optimisation de ressources.

L'algorithme UCB, émergeant comme la stratégie optimale dans le contexte des bandits-manchots, se positionne comme un choix judicieux pour la mise en œuvre d'un troisième joueur dans notre étude. Comparé à l'algorithme Monte Carlo, l'UCB offre une optimisation notable des ressources mobilisées. Son approche équilibrée entre exploration et exploitation, associée à une adaptation dynamique basée sur l'estimation de la moyenne des récompenses, confère à cet algorithme une efficacité accrue. En intégrant l'UCB en tant que troisième joueur, nous pourrions non seulement maximiser les gains potentiels, mais également minimiser l'utilisation de ressources, soulignant ainsi son potentiel pour des applications pratiques dans des contextes plus larges.