Exploitation vs Exploration : Le Dilemme du Puissance 4

Kitoko David & Bourenane Mohamed Elyes

6 octobre 2023

Table des matières

1 Introduction

Le dilemme de l'exploration par rapport à l'exploitation est un problème fondamental qui se pose dans divers domaines de l'intelligence artificielle, en particulier dans le domaine de l'apprentissage automatique. Il se pose lorsque l'on doit choisir entre différentes actions possibles. La question cruciale est de savoir s'il est préférable d'exploiter les connaissances déjà acquises en choisissant l'action qui semble la plus rentable à court terme, ou s'il vaut mieux continuer à explorer de nouvelles actions pour recueillir davantage d'informations. L'exploitation consiste à prendre la meilleure décision en fonction des données actuelles, tandis que l'exploration vise à acquérir des informations supplémentaires.

Il peut être avantageux, notamment au début d'un processus, de renoncer à choisir l'action potentiellement la plus rentable à court terme afin d'améliorer les gains à long terme. Cependant, la question cruciale reste de déterminer quand il est approprié de mettre fin à la phase d'exploration, c'est-à-dire quand considère-t-on avoir collecté suffisamment d'informations, au-delà desquelles l'exploration n'apportera pas de connaissances supplémentaires significatives.

L'objectif de ce projet vise à analyser le jeu Puissance 4 sous un angle mathématique en examinant les modèles probabilistes sous-jacents, en étudiant les stratégies optimales pour les joueurs tout en

prenant en compte le dilemme exploration vs exploitation inhérent au processus de prise de décision.

2 Description du code

2.1 First Stat Project

— **First Stat Project/** : Le dossier racine du projet.

2.2 Dossier "First Stat Project/"

— **BanditManchot.py** : Fichier contenant le code nécessaire demandé pour la partie Bandit-Manchot.
— **Constante** : Fichier possèdant toutes les confgurations du jeu et les conventions de programmation.
— **Data.py** : Fichier contenant diférentes données telles que les résultats des joueurs, des images ou des ressources nécessaires à la bonne excécution du programme.
— **Jeu.py** : Fichier contenant le code la simulation du jeu puissance 4 entre 2 joueurs.
— **Joueur.py** : Fichier contenant la classe Joueur, qui initialise les attributs d'un joueur.
— **Node.py** : Fichier contenant le code nécessaire demandé pour la partie UCT.
— **Plateau.py** : Fichier possédant le code qui initiliase le plateau de jeu puissance 4.
— **UCT.py** : Fichier contenant le code nécessaire demandé pour la partie UCT.
— **main.py** : Fichier principal exécutant le jeu.
— **RapportStat1.pdf** : Fichier contenant le rapport du projet.
— **Readme.txt** : Fichier contenant les réponses aux questions.
— **Readme.md** : Fichier contenant le tutoriel pour l'exécution du main.

3 Le Modèle Probabiliste du Puissance 4

3.1 Modélisation du Jeu en Tant que Processus Stochastique

Le Puissance 4 est un jeu dynamique où chaque coup joué par un joueur influence l'état global du jeu. L'objectif majeur de cette partie est d'explorer comment le Puissance 4 peut être formalisé en tant que processus stochastique. En utilisant une distribution binomiale négative, couramment employée pour modéliser le nombre de tentatives nécessaires avant qu'un événement spécifique se produise :

$$P(X = k) = \binom{k + r - 1}{k} p^r (1 - p)^k \tag{1}$$

Où :

$P(X = k)$: Probabilité que le nombre de coups nécessaires soit égal à k.
r : Nombre de succès souhaités .
p : Probabilité de succès dans chaque essai .
k : Nombre total d'essais nécessaires (c'est ce que nous cherchons à déterminer).

Cependant, la modélisation stochastique, bien que puissante, ne résout pas entièrement le Puissance 4. Le défi central du jeu réside dans la compréhension des probabilités de victoire, une tâche complexe.

3.2 Analyse d'une Victoire

L'attribution de probabilités aux actions est capitale dans l'analyse probabiliste du Puissance 4, car elle permet non seulement d'évaluer les choix des joueurs en fonction de diverses stratégies, mais aussi de quantifier les probabilités de victoire à chaque étape du jeu :

$$P(action|état, victoire) \tag{2}$$

Où :

action : Représente une action spécifique, telle que le choix d'une colonne pour placer un jeton.

état : Désigne l'état actuel du jeu, c'est-à-dire la configuration de la grille à un moment donné.

victoire : Indique si cette action conduit à une victoire potentielle ou non.

Cette formulation permet d'évaluer la probabilité conditionnelle que le joueur opte pour une action donnée, compte tenu de l'état actuel du jeu et de la possibilité de remporter la partie grâce à cette action. Plus précisément, il s'agit de déterminer si une action particulière augmente les chances de gagner. Elle peut être influencée aussi par divers facteurs, notamment les stratégies de jeu des joueurs, les informations disponibles, et les objectifs à court et à long terme. Par exemple, une action qui contribue à la formation d'une combinaison gagnante aura une probabilité de victoire plus élevée que d'autres actions.

3.3 Les Stratégies Probabilistes pour un Équilibre Gagnant

Le dilemme entre exploitation et exploration dans le contexte probabiliste du Puissance 4 peut être exprimé mathématiquement en utilisant des probabilités. Supposons Pv la probabilité de victoire d'un joueur à un état de jeu donné v. Cette probabilité dépend des actions disponibles à partir de cet état et peut être décomposée en deux composante :

$$P_v = \alpha P_v^{\text{exploitation}} + (1 - \alpha) P_v^{\text{exploration}} \tag{3}$$

$P_v^{\text{exploitation}}$: Représente la probabilité de victoire en exploitant les choix les plus avantageux.

$P_v^{\text{exploration}}$: Représente la probabilité de victoire en explorant de nouvelles possibilités.

α : Paramètre de pondération .

Le dilemme se pose dans la recherche de la valeur optimale de α qui maximise les chances globales de victoire. Ce choix dépendra de la situation actuelle du jeu, des stratégies de l'adversaire, et des préférences du joueur en matière de risque. Une approche courante pour résoudre ce dilemme consiste à ajuster dynamiquement.

Le Puissance 4 est un jeu complexe qui peut être modélisé en tant que processus stochastique avec une distribution binomiale négative, mais la compréhension des probabilités de victoire, intégrant le dilemme exploration vs exploitation, reste le défi central.

Cette partie explore les fondements mathématiques et stratégiques du Puissance 4, montrant comment les décisions des joueurs et les probabilités s'entremêlent pour créer un environnement de jeu dynamique et complexe.

4 Stratégies pour Maximiser les Chances de Victoire dans le Puissance 4

Nous allons examiner comment les joueurs peuvent développer des stratégies gagnantes tout en explorant de nouvelles approches dans le Puissance 4. Le défi consiste à optimiser leurs décisions en intégrant à la fois l'exploitation de stratégies éprouvées et l'exploration pour découvrir des tactiques innovantes. Nous allons explorer plusieurs approches pour atteindre cet équilibre

4.1 Algorithmes axés sur l'Exploitation

Les algorithmes axés sur l'exploitation dans le contexte du Puissance 4 visent principalement à maximiser les gains immédiats en utilisant les informations disponibles. Leur objectif central est de choisir l'action jugée la plus favorable en fonction des données accumulées jusqu'à présent. Par exemple, l'algorithme Greedy privilégie systématiquement l'action qui offre la meilleure probabilité de victoire,

en se basant sur des critères tels que la configuration actuelle du plateau. Les joueurs Greedy cherchent ainsi à minimiser le risque de coups défavorables, tout en visant à accumuler des avantages de manière stable au fil du jeu :

$$A_{\text{greedy}} = \arg\max Q(a) \tag{4}$$

Où :

A_{greedy} : Représente l'action choisie par l'algorithme Greedy.
a : Représente une action possible.
$Q(a)$: Représente la valeur estimée d'une action possible.

Cependant, il est important de noter que la stratégie purement axée sur l'exploitation peut comporter des limites, car elle ne prend pas en compte la possibilité de découvrir de nouvelles tactiques gagnantes. Dans le Puissance 4, comme dans d'autres jeux, une dose d'exploration peut s'avérer cruciale pour rester compétitif et découvrir des stratégies non encore exploitées. Les joueurs doivent donc équilibrer judicieusement l'exploitation de leurs connaissances actuelles avec l'exploration de nouvelles opportunités pour maximiser leurs chances de victoire

4.2 Algorithmes axés sur l'Exploration

Les algorithmes axés sur l'exploration dans le contexte du Puissance 4 adoptent une approche différente de celle des algorithmes axés sur l'exploitation. Ils privilégient la collecte d'informations supplémentaires en explorant diverses possibilités, même si ces choix ne semblent pas immédiatement optimaux. L'objectif principal de ces algorithmes est d'acquérir une connaissance plus approfondie du jeu, ce qui peut conduire à des avantages à long terme. L'un des algorithmes populaires pour l'exploration est l'algorithme Epsilon-Greedy.

L'algorithme Epsilon-Greedy repose sur une stratégie d'exploration et d'exploitation équilibrée. Il choisit généralement l'action la plus prometteuse en fonction des informations disponibles, mais avec une probabilité epsilon ϵ, il opte pour une exploration aléatoire. Cela signifie qu'avec une petite probabilité , l'algorithme effectuera un mouvement aléatoire plutôt que de suivre la meilleure action connue :

$$\begin{cases} \text{Action optimale actuellement connue} & \text{avec probabilité } 1-\epsilon \\ \text{Action aléatoire} & \text{avec probabilité } \epsilon \end{cases} \tag{5}$$

Les algorithmes axés sur l'exploration, tels que l'algorithme Epsilon-Greedy, offrent un équilibre entre la découverte de nouvelles stratégies et l'exploitation des choix prometteurs. Dans le Puissance 4, ils permettent d'explorer des options innovantes tout en tirant parti des actions actuellement favorables, favorisant ainsi l'acquisition de connaissances à long terme. Il est important de noter qu'il existe également des algorithmes qui combinent à la fois l'exploration et l'exploitation pour trouver un équilibre subtil entre la découverte de nouvelles tactiques et l'optimisation des choix actuels.

4.3 Algorithmes Équilibrés : Maximiser les Gains tout en Restant Ouverts à l'Exploration

Les algorithmes d'équilibrage entre exploration et exploitation jouent un rôle crucial dans le Puissance 4, et deux d'entre eux, l'UCB (Upper Confidence Bound) et MCTS (Monte Carlo Tree Search), se distinguent par leur capacité à atteindre cet équilibre délicat.

4.3.1 Exploration Intelligente : L'Algorithnme UCB

L'algorithme UCB (Upper Confidence Bound) est une technique ingénieuse pour résoudre le dilemme entre exploration et exploitation. :

$$a_t = \text{argmax}_{i \in \{1,\ldots,N\}} \left(\hat{\mu}_i(t) + \sqrt{\frac{q \cdot \log(t)}{N_t(i)}} \right)$$

a_t : C'est l'action choisie à l'étape t.

N : C'est le nombre total d'actions possibles.

$\hat{\mu}_i(t)$: C'est l'estimation de la moyenne de récompense de l'action i jusqu'à l'étape t.

q : C'est une constante qui peut être utilisée pour ajuster l'équilibre entre exploration et exploitation. Elle dépend généralement de la plage de récompenses possibles et du niveau de confiance souhaité.

$\log(t)$: C'est le logarithme naturel du nombre d'étapes effectuées jusqu'à présent.

$N_t(i)$: C'est le nombre de fois que l'action i a été choisie jusqu'à l'étape t.

Le premier facteur, $\hat{\mu}_i(t)$, représente l'estimation actuelle de la récompense moyenne attendue de l'action i jusqu'à l'étape t. Cette estimation est basée sur les récompenses réelles reçues jusqu'à présent. En d'autres termes, c'est une mesure de la performance passée de chaque action. Ce facteur favorise naturellement les actions qui ont eu de bonnes performances dans le passé, ce qui est cohérent avec l'idée d'exploitation.

Le deuxième facteur, $\frac{q \cdot \log(t)}{N_t(i)}$, mesure l'exploration. Plus une action n'a pas été fréquemment choisie ($N_t(i)$ est faible), plus ce terme est élevé, ce qui signifie que cette action a plus de chances d'être explorée. Le logarithme naturel $\log(t)$ augmente lentement au fil du temps à mesure que t augmente, ce qui implique que l'agent devient progressivement plus orienté vers l'exploitation à mesure qu'il gagne en expérience. Cela reflète l'idée que, dans les premières étapes du jeu, il est important d'explorer différentes actions pour obtenir une estimation plus fiable de leurs récompenses potentielles. Cependant, à mesure que le temps passe, on commence à se fier davantage à ses estimations actuelles et à privilégier les actions qui ont montré des performances prometteuses.

En combinant ces deux facteurs, l'algorithme UCB parvient à équilibrer intelligemment l'exploration et l'exploitation.Alors qu'UCB se focalise sur les actions individuelles, MCTS adopte une approche plus globale en construisant un arbre de recherche pour évaluer toutes les séquences de coups possibles dans le Puissance 4.

4.3.2 Exploration Arborescente : L'Algorithme MCTS (Monte Carlo Tree Search) pour le Puissance 4

L'algorithme MCTS est une méthode séquentielle pour explorer des séquences de coups dans le Puissance 4 en construisant un arbre de recherche. Il équilibre l'exploration et l'exploitation.Tout commence par la sélection, où l'algorithme choisit un nœud à explorer à partir de la racine de l'arbre, qui représente l'état actuel du jeu. Pour cela, il utilise l'algorithme UCB (Upper Confidence Bound), qui équilibre la recherche entre l'exploration des nœuds moins explorés et l'exploitation des nœuds prometteurs :

Sélection du nœud par UCB

On choisit le nœud suivant en fonction de l'algorithme UCB parmi les enfants du nœud courant, jusqu'à tomber soit sur une feuile de l'arbre, soit sur un état où une des actions n'a jamais été explorée.

Expansion du nœud

Une fois le nœud candidat sélectionné, l'algorithme passe à l'étape d'expansion. Ici, il vérifie s'il existe des actions non encore explorées à ce nœud, et s'il en trouve, il crée un nouvel enfant pour représenter cette action.

Simulation à partir d'un nœud

Ensuite, l'algorithme procède à la simulation. À partir du nœud nouvellement créé (ou du nœud sélectionné s'il n'y avait pas d'actions non explorées), il simule le jeu en effectuant des coups aléatoires jusqu'à atteindre un état final du jeu, que ce soit une victoire, une défaite ou un match nul. Pendant cette phase de simulation, les états visités ne sont pas stockés dans l'arbre.

Rétro-propage des résultats

Enfin, vient l'étape de rétro-propagation. Une fois la simulation terminée, le résultat de la partie simulée est propagé vers le haut de l'arbre. Cela signifie que les nœuds sur le chemin de la racine à l'enfant simulé sont mis à jour en fonction du résultat de la simulation.

5 La Flexibilité des Algorithmes Face aux Stratégies Opposées et variations du Plateau de Jeu

MCTS (Monte Carlo Tree Search) équilibre judicieusement l'exploration de nouvelles actions avec l'exploitation d'actions prometteuses.Cependant, il est pertinent de se demander comment ces algorithmes parviennent à s'adapter aux variations externes, telles que les modifications du plateau de jeu ou les différentes stratégies auxquelles ils sont confrontés.

5.1 Confrontation des Stratégies

5.1.1 Exploration vs Exploitation

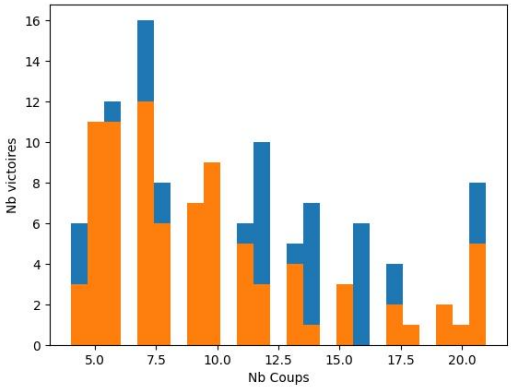

FIGURE 1 – Histogramme répresentant Greedy vs E-Greedy

Pour cette expérience, nous avons délibérément choisi une valeur élevée pour le paramètre epsilon (0,3) dans l'algorithme Epsilon-Greedy (représenté en orange) afin de favoriser l'exploration. Nous l'avons ensuite confronté à l'algorithme Greedy (représenté en bleu), qui privilégie davantage l'exploitation. Nous avons mené une série de 200 affrontements pour analyser la distribution du nombre de victoires en fonction du nombre de coups dans ces confrontations.

Nous pouvons constater que la proportion de victoires lors des premiers coups favorise clairement l'algorithme Epsilon-Greedy. Cependant, à mesure que le nombre de coups augmente, nous observons que l'algorithme Greedy devient plus avantageux que Epsilon-Greedy. En conséquence, nous pouvons conclure que lors des premiers coups, il est plus bénéfique d'explorer davantage les configurations du plateau de jeu. Cependant, à mesure que la partie progresse, il est préférable d'exploiter les données actuelles plutôt que de continuer à explorer de manière excessive.

Voici quelques chiffres relatifs à l'affrontement de ces deux algorithnmes :

	NB parties
Nb parties nulles	10
Nb parties gagnées joueur 1 (Greedy)	81
Nb parties gagnées joueur 2 (Epsilon-Greedy)	109

Epsilon-Greedy se montre plus efficace que Greedy. Cependant, il est essentiel de le confronter à un algorithme équilibré pour évaluer son comportement à long terme. Nous allons donc le tester en compétition avec un tel algorithme.

5.1.2 Exploitation vs Equilibré

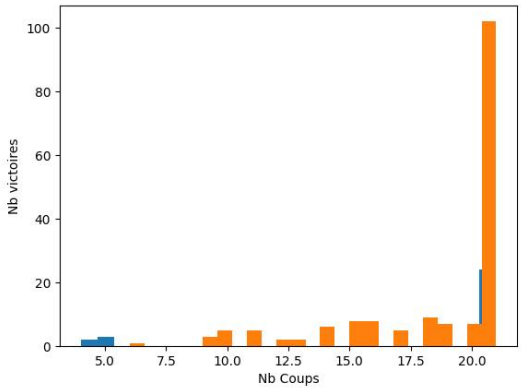

FIGURE 2 – Histogramme répresentant Espsilon-Greedy vs UCB

Comme précédement nous avons délibérément choisi une valeur élevée pour le paramètre epsilon (0,3) dans l'algorithme Epsilon-Greedy (représenté en bleu) afin de favoriser l'exploration. Nous l'avons ensuite confronté à l'algorithme UCB (représenté en orange), qui privilégie davantage l'exploitation. Nous avons mené une série de 200 affrontements pour analyser la distribution du nombre de victoires en fonction du nombre de coups dans ces confrontations.

Nous remarquons que dans les premiers coups, l'algorithme Epsilon-Greedy a légèrement l'avantage en termes de proportion de victoires. Cependant, à mesure que le nombre de coups augmente, l'algorithme UCB devient nettement plus performant que Epsilon-Greedy. Ainsi, nous pouvons conclure que, bien qu'ils soient à peu près à égalité au début, UCB prend largement le dessus sur Epsilon-Greedy au fur et à mesure de la progression de la partie.

Voici quelques chiffres relatifs à l'affrontement de ces deux algorithnmes :

	NB parties
Nb parties nulles	120
Nb parties gagnées joueur 1 (UCB)	75
Nb parties gagnées joueur 2 (Epsilon-Greedy)	5

Pour approfondir notre analyse, nous allons maintenant confronter deux algorithmes équilibrés dans une série de parties. Cette confrontation nous permettra de mieux comprendre leur comportement lorsqu'ils sont soumis à des stratégies similaires axées à la fois sur l'exploration et l'exploitation.

5.1.3 Confrontation de Deux Algorithmes Équilibrés

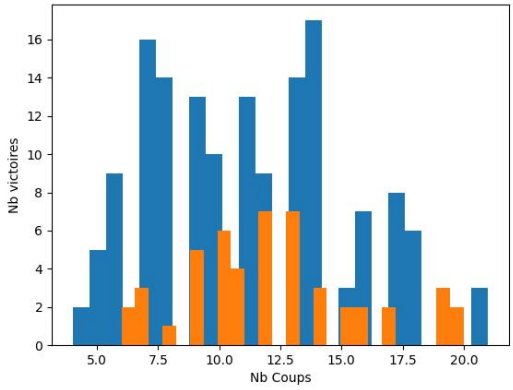

FIGURE 3 – Histogramme réprésentant MCTS vs UCB

Il ressort clairement de cet affrontement entre les algorithmes MCTS (en bleu) et UCB (en orange) que MCTS domine complètement UCB. Cette domination de MCTS pourrait s'expliquer par plusieurs facteurs, notamment son approche basée sur la simulation de parties complètes, qui lui permet d'explorer plus efficacement l'espace de recherche du jeu. Maintenant que nous avons examiné les performances de MCTS et d'autres algorithmes dans le contexte actuel, nous pouvons nous demander si ces relations resteraient les mêmes si nous modifiions le plateau de jeu

5.2 Variation du Plateau de Jeu

Dans cette section, nous allons explorer les conséquences de la modification de la configuration de notre plateau de jeu en doublant à la fois le nombre de lignes et de colonnes. Nous allons effectuer les mêmes analyses que précédemment pour évaluer l'impact de cette expansion sur les performances de nos algorithmes.

5.2.1 Exploration vs Exploitation avec changement de la configuration du Plateau de Jeu

Voici le tableau récapitulatif des résultats des matchs, comprenant les victoires, les défaites et les matchs nuls, avec la nouvelle configuration de jeu pour la confrontation entre Greedy et Epsilon-Greedy :

	NB parties
Nb parties nulles	0
Nb parties gagnées joueur 1 (Greedy)	128
Nb parties gagnées joueur 2 (Epsilon-Greedy)	78

Lorsque la taille du plateau de jeu augmente, Epsilon-Greedy peut perdre en performance en raison de la complexité croissante de l'espace d'exploration. Plus précisément, à mesure que le plateau devient plus grand, Epsilon-Greedy consacre une fraction constante de ses actions à l'exploration, ce qui peut rendre son exploitation des choix les plus prometteurs moins efficace. Par conséquent, ses performances peuvent diminuer par rapport à des algorithmes tels que Greedy, qui privilégient davantage l'exploitation des options connues

Dans cette configuration, il semble que l'approche axée sur l'exploitation, représentée par l'algorithme Greedy, soit plus efficace que l'approche axée sur l'exploration, telle qu'incarnée par Epsilon-Greedy. Par conséquent, nous allons poursuivre notre analyse en confrontant Greedy à un algorithme plus sophistiqué.

5.2.2 Exploitation vs UCB avec changement de la configuration du Plateau de Jeu

Voici le tableau récapitulatif des résultats des matchs, comprenant les victoires, les défaites et les matchs nuls, avec la nouvelle configuration de jeu pour la confrontation entre Greedy et UCB :

	NB parties
Nb parties nulles	0
Nb parties gagnées joueur 1 (Greedy)	1
Nb parties gagnées joueur 2 (UCB)	199

En poursuivant notre étude, nous avons constaté que l'augmentation de la taille du plateau de jeu n'a pas altéré la supériorité de l'algorithme UCB par rapport au Greedy. Cette constatation met en évidence la robustesse de l'UCB face aux variations de l'environnement de jeu. Quelle que soit la taille du plateau, l'UCB parvient à maintenir sa capacité à équilibrer l'exploration et l'exploitation de manière efficace. Il se montre capable de s'adapter à des environnements de jeu plus vastes sans sacrifier ses performances. À l'inverse, le Greedy semble souffrir davantage de cette augmentation de la complexité,

Fort de ces résultats, nous allons maintenant confronter l'UCB à un adversaire plus redoutable pour évaluer sa performance dans des conditions encore plus exigeantes

5.2.3 UCB vs MCTS avec changement de la configuration du Plateau de Jeu

Voici le tableau récapitulatif des résultats des matchs, comprenant les victoires, les défaites et les matchs nuls, avec la nouvelle configuration de jeu pour la confrontation entre MCTS et UCB :

	NB parties
Nb parties nulles	0
Nb parties gagnées joueur 1 (MCTS)	2435
Nb parties gagnées joueur 2 (UCB)	2180

Nos expériences confirment que même lorsque la configuration du plateau de jeu évolue, l'algorithme MCTS (Monte Carlo Tree Search) conserve son avantage sur l'UCB (Upper Confidence Bound). Cela souligne la robustesse et la cohérence des performances de MCTS face à des environnements de jeu changeants, ce qui renforce sa réputation en tant qu'approche solide pour aborder des problèmes complexes.

En revanche, l'algorithme UCB, bien qu'il ait fait ses preuves dans de nombreuses situations, montre des signes de faiblesse lorsque l'espace d'exploration devient considérablement plus vaste. Dans de telles circonstances, il peut avoir du mal à équilibrer efficacement l'exploration et l'exploitation, ce qui nuit à ses performances globales.

Ces observations mettent en lumière la polyvalence et l'efficacité constantes de l'algorithme MCTS, ce qui en fait un choix privilégié pour résoudre des problèmes complexes et dynamiques. Elles soulignent également l'importance de sélectionner l'algorithme le plus adapté en fonction de la nature spécifique du problème et de son contexte.

6 Conclusion

Cette étude a exploré différents aspects du jeu Puissance 4. Nous avons analysé sa complexité, examiné diverses stratégies, et confronté des algorithmes. On a pu voir que dans des situations complexes et changeantes, il est essentiel d'adapter sa stratégie en fonction de l'évolution du jeu. La recherche continue dans ce domaine contribue à développer des algorithmes plus intelligents et plus efficaces pour aborder ce dilemme, que ce soit dans le Puissance 4 ou d'autres domaines.