

UNIVERSITÉ DE MONTPELLIER

M1 IMAGINE - Algorithme d'exploration et de
mouvement
Compte Rendu - Morpion et minimax algorithme avec
chatGPT

Etudiant :
Guillaume Bataille
Daniel Blanchard

Encadrant :
Madalina CROITORU

Année 2022-2023



0.1 Résultat final

```
○ bambipc@bambipc-System-Product-Name:~/Bureau/M1S2$ /bin/python3 "/home/bambipc/Bureau/M1S2/Algoexp1.py"
Enter 1 for single player, 2 for multiplayer: 1
Enter grid size (e.g., 3 for 3x3, 4 for 4x4, etc.): 4
Enter search depth for the AI (e.g., 1 for a quick game, 5 for a stronger opponent): 5
Computer: O Vs. You: X
Enter to play 1(st) or 2(nd): 1

Current State Of Board:

- - - -
- - - -
- - - -
- - - -

Enter X's position from [1...16]: 5
AI took: 30.72 seconds to play
File size: 15.08 Mb
Nodes numbers : 396075 lines.
Writing flowrate: 0.49 Mb/s
Current State Of Board:

O - - -
X - - -
- - - -
- - - -

Enter X's position from [1...16]:
```

FIGURE 1 – Morpion contre IA minimax avec grille de taille 4x4 et profondeur max de l’algo 5. Une analyse sur la complexité d’espace et de temps est aussi réalisé (file contient les noeuds de l’arbre)

0.2 Contexte et Objectif

Pour ce petit projet, nous avons été choisi pour utiliser chatgpt pour travailler sur l’implémentation d’un algorithme minimax via un jeu du morpion.

Toutes nos étapes de développement consisteront donc de nous aider de ChatGPT le plus possible afin de nous alléger la tâche mais aussi pour apercevoir d’éventuelles limite de cet IA.

Pour spécifier le contexte :

- Nous parlons exclusivement en français avec chatGPT
- Sauf si ce n’est pas précisé, nous restons dans la même chatbox avec lui.
- Human = nous et AI = chatGPT

Sommaire

0.1	Résultat final	1
0.2	Contexte et Objectif	1
1	Le jeu du Morpion selon l'IA	3
2	L'algo minimax selon l'IA	4
3	Implémentation Python	5
3.1	Essai 1	5
3.2	Essai 2	6
3.3	Un question se pose.. Taille modifiable	7
3.4	Et après ? Profondeur max modifiable	8
4	Bonus : Implémentation Javascript	9
5	Conclusion	11
5.1	Annexe	12

1. Le jeu du Morpion selon l'IA

Pour commencer, nous demandons à chatGPT s'il connaît le jeu.

Human Tu connais le jeu du morpion ?

AI Oui, je connais le jeu du morpion ! C'est un jeu de société très populaire qui se joue à deux joueurs sur une grille de 3x3 cases. Le but du jeu est de placer trois symboles identiques (généralement des croix ou des ronds) en ligne horizontale, verticale ou diagonale.

Le joueur qui réussit à aligner trois symboles gagne la partie. Si toutes les cases de la grille sont remplies sans qu'aucun joueur ne parvienne à aligner trois symboles, la partie se termine par un match nul.

Le jeu du morpion est un jeu relativement simple et rapide à jouer, mais il peut tout de même nécessiter une certaine réflexion stratégique pour parvenir à gagner.

Conclusion Il semble qu'il a bien identifié ce dont on parle ici en évoquant les bonnes règles de ce jeu.

2. L’algo minimax selon l’IA

On s’intéresse à présent sur l’algorithme minimax vu en cours en lui demandant si il le sait mais surtout si il voit le lien avec le jeu du morpion.

Human Connais tu l’algorithme minimax ?

AI Oui, je connais l’algorithme Minimax ! Il s’agit d’un algorithme d’exploration de l’arbre de décision utilisé dans les jeux à deux joueurs, tels que le jeu du morpion, le jeu d’échecs, le jeu de Go, etc.

L’algorithme Minimax permet de déterminer le meilleur coup à jouer pour un joueur donné, en se basant sur une évaluation de l’état du jeu à chaque nœud de l’arbre de décision. L’objectif est de maximiser les chances de gagner si l’on est le joueur Max, et de minimiser les chances de gagner si l’on est le joueur Min.

L’algorithme Minimax fonctionne en explorant l’arbre de décision à partir de la racine, qui représente l’état actuel du jeu. À chaque nœud de l’arbre, l’algorithme calcule la valeur de l’état du jeu en utilisant une fonction d’évaluation. Cette fonction attribue une valeur numérique à l’état du jeu, en fonction de la qualité de la position pour le joueur Max ou Min.

Ensuite, l’algorithme explore récursivement les branches de l’arbre, en alternant entre les joueurs Max et Min, jusqu’à atteindre les feuilles de l’arbre (c’est-à-dire les positions finales possibles). En remontant ensuite l’arbre, l’algorithme détermine le meilleur coup à jouer pour le joueur Max.

Il existe des variantes de l’algorithme Minimax, comme l’élagage alpha-bêta, qui permettent de réduire le nombre de nœuds à explorer et ainsi d’optimiser les performances de l’algorithme.

Human Quel lien ont le morpion et cet algorithme ?

AI L’algorithme Minimax est souvent utilisé pour résoudre le jeu du morpion. Comme je l’ai mentionné précédemment, le morpion est un jeu à deux joueurs avec des informations complètes, ce qui signifie que les joueurs ont une connaissance complète de l’état actuel du jeu. En utilisant l’algorithme Minimax, il est possible de trouver le meilleur coup possible à jouer à chaque tour, en supposant que les joueurs jouent de manière optimale.

Conclusion Il nous a donné une définition satisfaisante de l’algorithme minimax et semble avoir identifié le lien entre morpion et minimax.

3. Implémentation Python

Pour l'implémentation, nous avons choisi python car dans la précédente expérience avec chatgpt, nous avons vu que le python faiblement typé et simple syntaxiquement réduit les erreurs de compilation du code qu'il nous donne (à l'inverse du c/c++).

A chaque essai avec chatGPT, nous avons demandé les questions précédente afin d'avoir un contexte ou il sait qu'on parle du minimax avec le morpion.

3.1 Essai 1

On lui demande simplement d'implémenter ce qui a été dit auparavant.

Human Implémente moi un morpion avec minimax en python

AI D'accord, voici le code *il commence à donner le code mais s'arrête brusquement après 2236 caractères*

Human Tu t'es arrêté à la fonction CheckWin(), continue

AI Désolé, voici l'implémentation d'un morpion avec l'algorithme minimax *Il recommence depuis le début et s'arrête à 2215 caractères*

Conclusion - ECHEC Cet essai est un échec.

Nous avons réessayer quelques fois avec cette méthode mais il semble que bien qu'il à l'air de savoir la réponse, il semble qu'il n'a pas la possibilité de fournir plus de 2300 caractères dans une même requête.

3.2 Essai 2

L'échec précédent nous a donné une idée, celle de séparer la quantité de caractère à lui demander pour toujours s'assurer qu'il ne s'interrompt pas au milieu de ses implémentations.

Human Donne nous une implémentation du morpion simple en python

AI Bien-sûr *Il nous donne entièrement l'implémentation en python*

Nous compilons et executons le code, et le jeu fonctionne.

C'est, comme nous l'avais spécifier l'IA, une version Joueur contre Joueur du morpion.

Nous essayons donc de le rendre jouable par une IA mais sans parler de minimax pour ne pas le surcharger.

Human Super! A partir de cette version on souhaite pouvoir jouer contre une IA qui joue de façon aléatoire. Peux-tu l'implémenter pour moi?

AI Voici une version ou le joueur joue contre une IA naïve qui joue aléatoirement *Il nous donne l'implémentation et la fini*

Cette nouvelle implémentation compile et fonctionne! On décide d'oser lui demander de modifier la fonction contrôlant l'IA pour qu'elle soit régie par l'algorithme minimax.

Human Maintenant, donne moi aiMove2() pour que l'IA joue selon un algorithme minimax.

AI Voici aiMove2 pour qu'une IA joue selon un algorithme minimax.*Donne la fonction et me donne aussi la fonction minimax*

Conclusion - SUCCES Cet essai est un succès, un code python d'une grille 3x3 avec minimax. Nous estimons la taille de l'arbre des possibilités a 9! 362 880 possibilités auquel il faut recouper les feuilles ou il y a une victoire avant la fin de la partie.

Cela prends au maximum quelques seconde pour un cpu moyen pour parcourir et concevoir cet arbre des possibilités. Voici ce que donne notre analyse en temps et en espace lorsqu'on écrit dans un ".txt" l'arbre minimax :

```
AI took: 3.49 seconds to play
File size: 11.09 Mb
Nodes numbers : 294777 lines.
Writing flowrate: 3.18 Mb/s
```

FIGURE 3.1 – Taille 3x3 et pas de profondeur max pour l'algo

3.3 Une question se pose.. Taille modifiable

Voici un élément qui, au vu du résultat précédent, nous a intrigué, combien de temps prendrait le parcours des possibilités d'une grille de taille 4x4 pour notre IA ?

Human Donne nous la possibilité de définir la taille de la grille de notre morpion.

AI Voici la version modifiée du morpion pour pouvoir spécifier la taille de la grille. *Il nous donne une version qui ne compile pas mais c'était à nous de modifier le main après sa modification de la grille*

Cependant, on lance avec une grille de taille 2 ou 3 cela fonctionne comme convenu et l'IA est imbattable. Mais avec une grille 4x4 l'algorithme semble **boucler à l'infini**. On a attendu que l'IA joue son coup mais 45min voilà ce qui s'est passé pendant que l'IA continuait d'écrire l'arbre minimax.

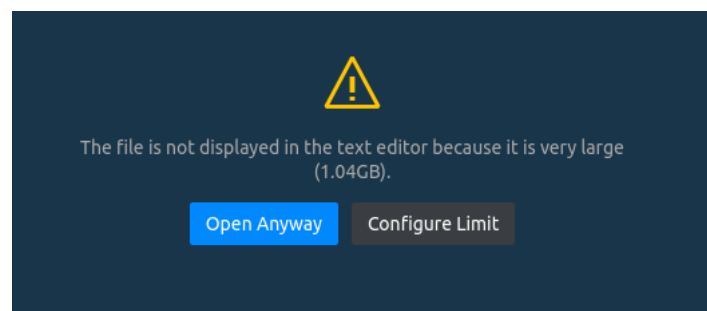


FIGURE 3.2 – Oh.. ça commence à faire beaucoup pour des nœuds d'un arbre

Nous avons donc demandé à chatGPT le nombre de nœuds présent dans cet arbre s'il devrait être construit

Human Si on est sur une grille 4x4, quel est la taille d'un arbre de possibilité minimax ?

AI Le nombre total de nœuds dans l'arbre de décision pour le morpion sur une grille de 4x4 cases sera donc plus grand que pour une grille de 3x3 cases, et sera de l'ordre de grandeur de $16! = 2.092279 \times 10^{13}$ nœuds. Soit 634 195 années si chaque nœud prends 1 seconde, on demande à chatgpt pour confirmer l'impossibilité de rendre cet arbre.

Human Combien de temps cela prendrait pour parcourir cet arbre ?

AI En pratique, il serait impossible de parcourir complètement cet arbre en un temps raisonnable, même avec un ordinateur très puissant. C'est pourquoi les techniques d'élagage telles que l'élagage alpha-bêta sont utilisées pour réduire le nombre de nœuds à explorer, ainsi que des heuristiques pour évaluer les positions plus rapidement et efficacement. Effectivement, cela prendrait sûrement les milliards d'années à être parcouru sans élagage.

3.4 Et après ? Profondeur max modifiable

On décide pour pouvoir quand même étudier minimax en taille 4x4 ou plus de définir une profondeur max, au delà de cette profondeur définie par l'utilisateur, on ne construira pas de noeud plus profond.

HUMAN Peut-on passer à une version de profondeur de l'algorithme minimax spécifiable par l'utilisateur ?

AI Bien-sûr *Il nous donne une version avec la profondeur à définir, elle ne compilait pas directement mais après lui avoir redonné le code éronné, il l'a corrigé*
Voici donc ce que l'on peut faire à présent :

```
o bambipc@bambipc-System-Product-Name:~/Bureau/M1S2$ /bin/python3 "/home/bambipc/Bureau/M1S2/Algoexp
Enter 1 for single player, 2 for multiplayer: 1
Enter grid size (e.g., 3 for 3x3, 4 for 4x4, etc.): 4
Enter search depth for the AI (e.g., 1 for a quick game, 5 for a stronger opponent): 5
Computer: 0 Vs. You: X
Enter to play 1(st) or 2(nd): 1

Current State Of Board:

- - - -
- - - -
- - - -
- - - -

Enter X's position from [1...16]: 5
AI took: 30.72 seconds to play
File size: 15.08 Mb
Nodes numbers : 396075 lines.
Writing flowrate: 0.49 Mb/s
Current State Of Board:

O - - -
X - - -
- - - -
- - - -

Enter X's position from [1...16]:
```

FIGURE 3.3 – Taille 4 profondeur max 5

Cela prends du temps mais nous pouvons jouer contre une IA qui anticipe au moins 5 coups à l'avance sur une grille 4x4!

On constate que la quantité de noeud est quand même impressionnante (+ de 300 000) ce qui donne un temps conséquent et un débit faible.

Conclusion - SUCCES Le code est fonctionnel et le travail en temps et en espace est cohérent avec ce à quoi on s'attendait.

Cela dit, une méthode d'élagage serait le bien venu car pour un 4x4, on attends presque la minute à chaque coup de l'IA..

4. Bonus : Implémentation Javascript

Après nos implémentation en python, nous avons essayer d'aller un peu plus loin en essayant à faire une version javascript dans une autre chatbox pour qu'elle soit à la fois plus visuelle mais aussi plus interactive (cliquer sur les cases pour jouer et non devoir donner les coordonnées au terminal python).

Human Donne nous une implémentation d'un morpion simple on ou peut choisir la taille de la grille

AI Pas de soucis! *Il nous donne une implémentation fonctionnelle d'un morpion simple avec la grille modifiable*

Human Modifie ce code pour pouvoir jouer soit contre une personne, soit contre une IA naïve.

AI Bien sûr, voici l'implémentation *Il nous donne une version fonctionnelle ou on peut jouer contre un joueur ou une IA*

Human Maintenant, modifie aiMove pour que l'ia puisse jouer selon un algorithme minimax.

AI Voici l'implémentation avec le nouvel aiMove *Il s'élance dans le code mais s'arrête encore une fois a 2232 caractères*

Et après plusieurs essais..

Conclusion - ECHEC Nous avons rencontré les mêmes soucis qu'avec C/C++ à savoir des erreurs de syntaxe mais surtout beaucoup d'arrêt au milieu du code.

Et impossible de lui dire de continuer sans qu'il perde le fil ou qu'il recommence depuis le début

Cela est sûrement dû au fait que javascript est est plus gourmand en caractère que python.

Nous avons essayé à de nombreuse reprise, mais voici ce qu'on à pu obtenir :

Tic Tac Toe

Choose the size of the board: Choose game mode: Choose the depth of the minimax algorithm:

	O	
X		

FIGURE 4.1 – Version javascript avec la taille de grille specifiable, jouable contre IA naïve(aleatoire) ou un humain

5. Conclusion

Pour conclure, chatGPT est très performant pour avoir une base de code "quickToStart" ou encore pour nous expliquer des concept/formule/algorithmes.

Même nous qui avons eu du mal avec les explications de nos collègues en cours, nous avons compris les subtilités de l'algorithme minimax rapidement avec chatGPT.

Cependant, pour finaliser une **Implémentation précise**, il a du mal à la réaliser.

Plus précisément, nous supposons qu'il sait exactement ce qu'il fait mais étant bridé au alentours de 2200 caractères, cela l'empêche d'aller droit au but. Et essayant de partitionner des interventions, il perds rapidement pied et se retrouve à faire appel à des fonctions qui n'existait pas, à oublier ce qu'il disait ou encore à recommencer au début en s'arrêtant au même endroit.

Malgré cela et certaines correction que nous avons dû manuellement apporté, nous avons pu avoir une version en python en moins de 2h, ce qui était sans doute impossible à faire from scratch dans ce délai imparti.

Cela montre donc que chatGPT est un outil puissant qui à la manière d'une bonne calculatrice facilite grandement le developpement mais il reste un outil qui assiste le développeur.

5.1 Annexe

Dans le CR, nous avons utilisé une analyse simple en se servant de chatGPT pour l'implémenter. Mais nous avons aussi de notre côté utilisé ce petit algo qui démontre ce qu'on a pu voir en Programmation Efficace d'Algorithme au semestre 1 du master Imagine.

```
def time_and_memory(self, f):
    def wrapper(*args, **kwargs):
        # Mesure de la mémoire utilisée
        tracemalloc.start()
        result = f(*args, **kwargs)
        current, peak = tracemalloc.get_traced_memory()
        tracemalloc.stop()

        # Affichage des résultats
        print(f"Temps d'exécution : {perf_counter()/1000000:.6f} secondes")
        print(f"Utilisation de la mémoire : {current / 10**6:.6f} Mo (pic de {peak / 10**6:.6f} Mo)")

    return result
    return wrapper
```

Voici ce qu'il peut donner pour une analyse plus poussée (notamment pour avoir ici la mémoire allouée et les pic de mémoire allouée) :

```
Utilisation de la mémoire : 0.004647 Mo (pic de 0.006111 Mo)
X | O |   |   |
-----
  |   |   |   |
-----
  |   |   |   |
-----
  |   |   |   |
-----
  |   |   |   |
-----
  |   |   |   |
Choisissez une case :
```