LU2IN013 - Projet de développement - Projet jeu à deux joueurs (Othello et Awele)

Sebastian SZEBRAT - 28706848
Baris KAFTANCIOGLU - 28711733

Description

Le projet consiste d'une plateforme de jeu modulaire avec des modules pour Awélé et Othello qui permet de tester des formes différentes de joueurs intelligents. Nous avons commencé par des algorithmes de recherche, puis de l'apprentissage.

Implémentation de la plateforme de jeu

Nous avons été fournis un fichier `game.py` avec les signatures des fonctions basiques pour le déroulement d'une partie.

La plateforme possède un répertoire par jeu (donc un répertoire `Awele` et un répertoire `Othello`). Dans ces répertoires, il y a un ficher python avec le nom du jeu qui possède les fonctions spécifiques à chaque jeu. Les sous-répertoires `Joueurs` contiennent tous les joueurs correspondant au jeu et le fichier `main.py` les fonctions de déroulement de partie et d'étude statistique.

Pour l'apprentissage, chaque jeu possède aussi un fichier `apprentissage_non_supervise.py` et `apprentissage_supervise.py` qui s'occupent du déroulement et de l'étude de l'apprentissage non supervisé et supervisé respectivement.

Joueurs

Tous les joueurs possèdent trois fonctions principales :

- `evaluation` : évalue la qualité d'un état de jeu à partir d'une série de paramètres (par exemple le nombre de cases vulnerables) qui composent un score en fonction de poids définis (ceci est utile pour l'apprentissage)
- `estimation` : une fonction qui évalue la qualité d'un coup, avec une série d'appels récursifs pour parcourir l'arbre du jeu, puis un appel d'`evaluation`
- `decision` : choisit le meilleur coup de la liste des coups possibles pour le joueur (fait appel à `estimation`)

Nous avons rendu disponible cinq types de joueurs pour chaque jeu :

Humain

Joueur qui permet à une personne de joueur contre un des joueurs intelligents ou une autre personne. Les coups sont sélectionnés dans la console.

Aléatoire

Joue aléatoirement un coup.

Horizon 1

Ce joueur a une profondeur limitée à 1, c'est-à-dire qu'il estime quel sera le meilleur coup en jouant le coup et regardant l'état prochain du jeu. Les joueurs avec un horizon plus élevé se réfèreront à l'état après plusieurs coups.

Minimax

L'algorithme minimax est utilisé pour passer à des profondeurs supérieures à 1, donc étudier l'état de jeu après 1 ou plusieurs coups. Il est censé choisir le meilleur coup pour lui-même et le pire coup pour son adversaire.

Alpha-bêta

Le même algorithme que minimax est utilisé, mais avec l'élagage alpha-bêta. On diminue le nombre d'états étudiés en supprimant les noeuds qui ont la probabilité la plus basse d'être "rentables".

Étude statistique

Pour faire des études statistiques sur nos joueurs et nos apprentissages, nous avons crée des fonctions qui fonctionnent de la manière suivante :

- une fonction "stats" génère des données à étudier et les sauvegarde dans le sous-répertoire `Donnees` à l'aide du module `pickle`, qui convertit les objets Python en flux d'octet (sérialisation)
- une fonction "graphique" génère un graphique à partir de ces données, qu'il sauvegarde dans le sous-répertoire `Graphiques`

Pour les fichiers `main`, la fonction stats est appelée par la fonction graphique. Il faut donc supprimer l'appel dans la fonction graphique si l'on ne veut pas générer des nouvelles données.

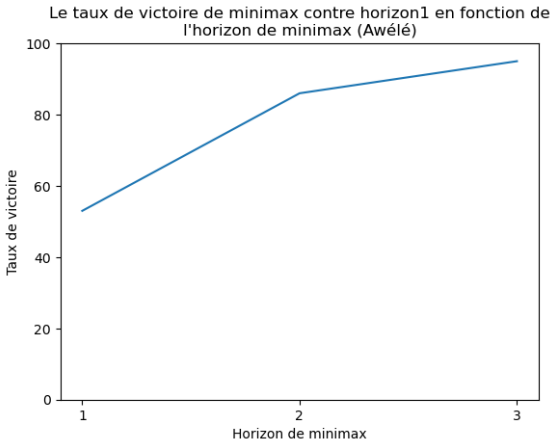
Pour les apprentissages, les deux fonctions sont appelées séparément.

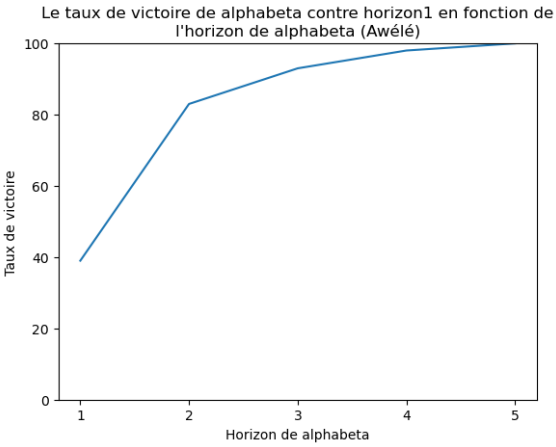
Nous avons gardé une copie des données utilisées pour générer les graphiques du rapport dans le sous-répertoire `Donnees` du répertoire `Rapport`. Elles sont également présents dans les répertoires des jeux, mais seront écrasées si nous lançons une nouvelle étude.

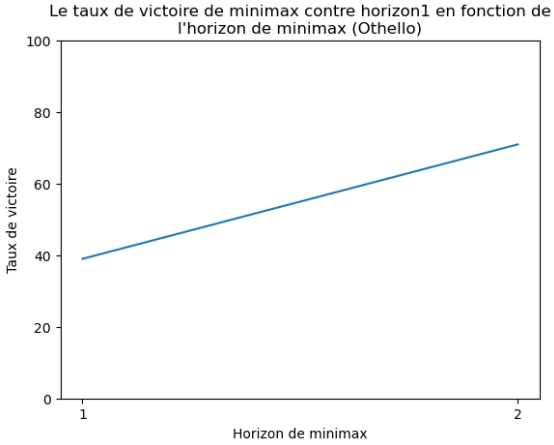
Étude de la performance des joueurs

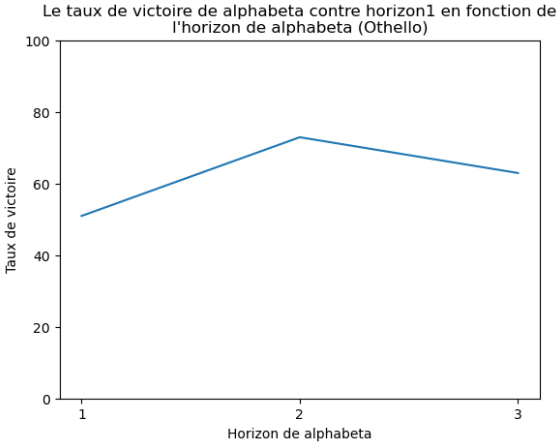
Nous avons d'abord testé les joueurs minimax et alphabeta contre le joueur horizon 1, en variant les horizons. Pour faire cela, nous avons crée une fonction qui fait la moyenne du taux de victoire sur deux tours de 50 parties : un tour où minimax/alphabeta est le premier joueur, et un tour où il est le deuxième.

Nous avons obtenu les courbes suivantes :



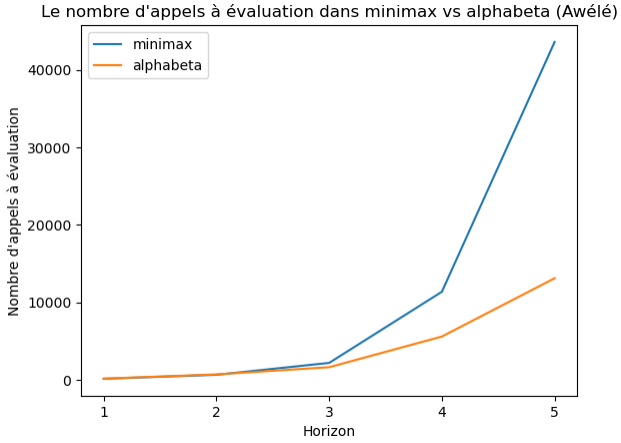


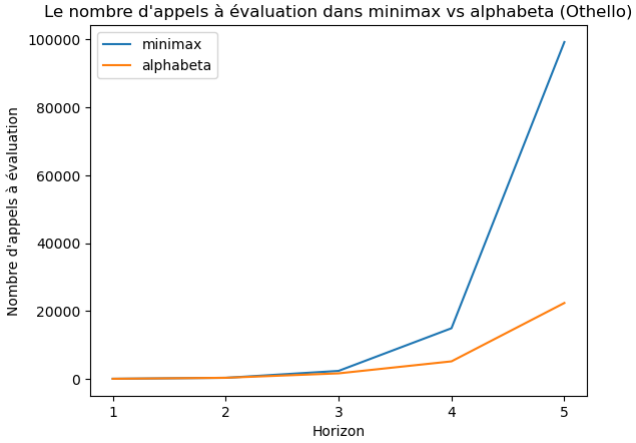




Nous constatons que pour Awélé, le taux de victoire, que ce soit minimax ou alphabeta contre horizon 1, est toujours croissant.
Pour Othello, le taux de victoire diminue légèrement à l'horizon 3. Ceci est dû au hazard, il n'y a pas une grande différence entre les horizons 2 et 3. Le taux de victoire augmenterait sûrement aux horizons suivants.

Nous avons ensuite voulu mesurer la différence de performance entre les joueurs minimax et alphabeta. Nous avons donc compté le nombre moyen d'appels à évaluation des deux joueurs sur le même ensemble de 100 parties.





Nous constations que la performance reste quasiment la même pour les deux jeux jusqu'à l'horizon 4, où l'on remarque une augmentation forte du nombre d'appels à évaluation dans minimax. Le nombre d'appels dans alphabeta double à chaque horizon alors que le nombre dans minimax augmente exponentiellement.

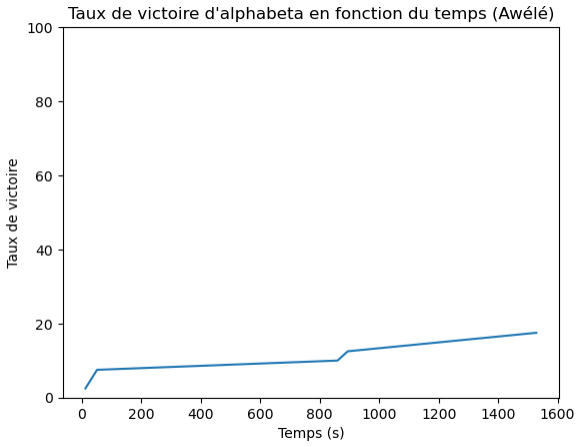
Apprentissage non supervisé

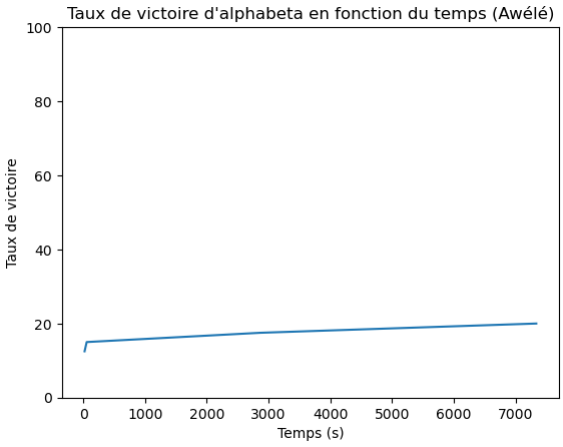
Pour l'apprentissage non supervisé, nous avons pris horizon 1 pour apprenti et alphabeta horizon 3 en tant qu'oracle.

Awélé Nous avons testé chaque poids sur 20 parties. Si la performance était meilleure que le précédent meilleur poids, on le remplace et on enregistre le nouveau poids dans le fichier `poids.txt`.

Notre étude a pour but de montrer l'évolution du taux de victoire d'horizon 1 contre alphabeta en fonction du temps (et non pas le taux de victoire d'alphabeta, il s'agit d'une erreur de notre part quand nous avons généré les graphiques).

Nous avons fait l'étude sur 4000 poids (image à droite). Nous avons également fait une étude plus petite pour voir en plus de précision la progression au début (image à gauche).

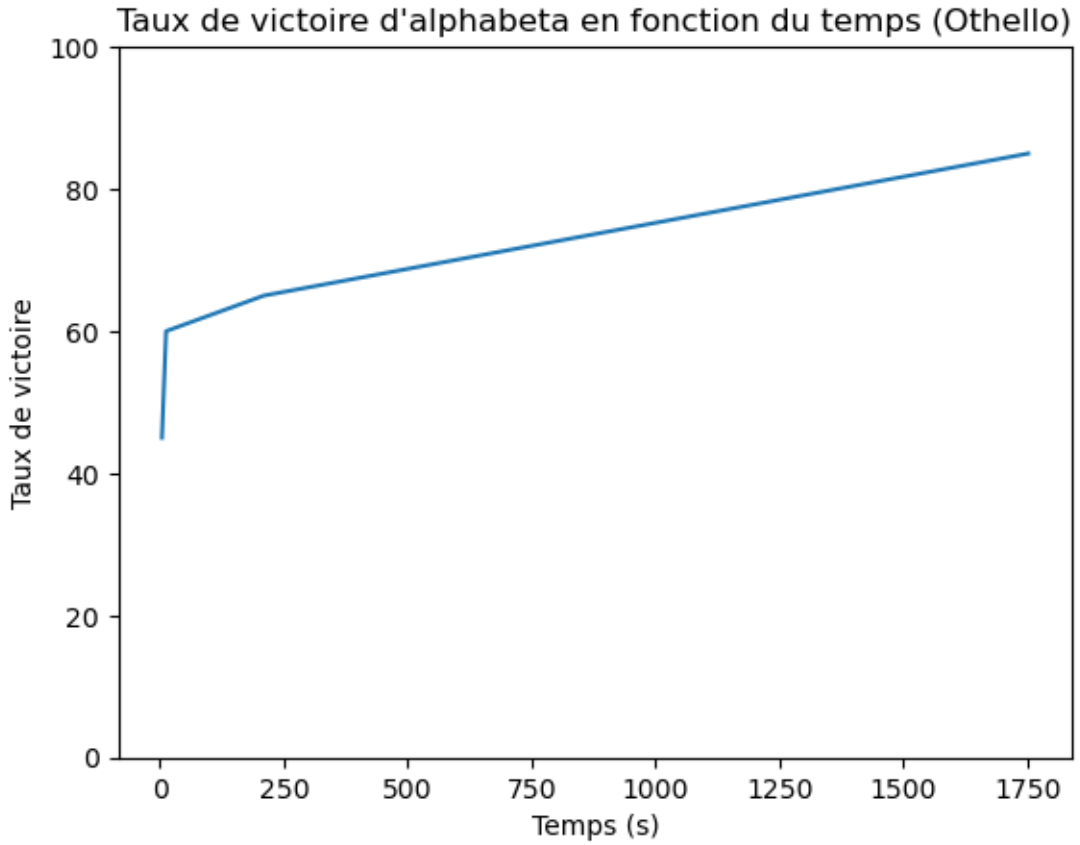




Nous constatons qu'il augment brusquement au début avant de se ralentir, on remarque pourtant que la courbe reste croissante jusqu'à la fin de l'étude. Il serait intéressant de faire des essais avec un nombre

d'itérations plus élévé.

Othello Ici, nous avons testé chaque poids sur 10 parties car le temps d'exécution est plus long pour Othello. Nous avons fait l'étude sur 1000 poids. Nous n'avons pas fait d'étude plus petite pour ce jeu. Il y a la même erreur qu'Awélé dans le titre du graphique.



On observe la même chose que pour Awélé.

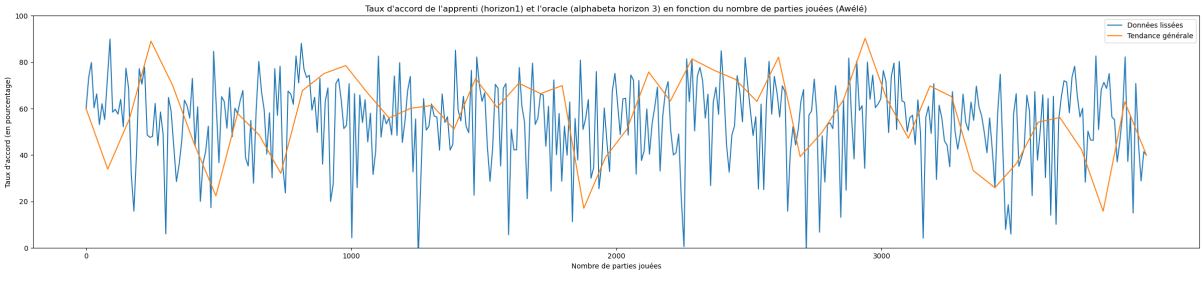
Apprentissage supervisé

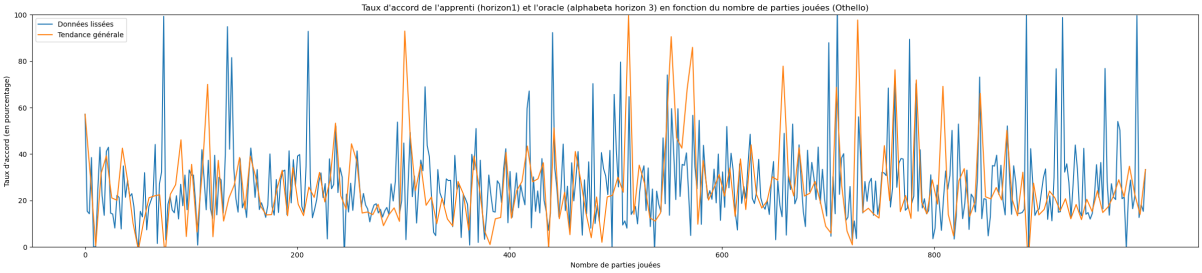
Pour l'apprentissage non supervisé, nous avons pris horizon 1 pour apprenti et alphabeta en tant qu'oracle.

alphabeta est d'horizon 5 pour Awélé et d'horizon 3 pour Othello. Il s'agit des horizons avec le meilleur rapport performance/temps d'exécution.

Nous mesurons la taux d'accord avec l'oracle en fonction du nombre de parties jouées.

Nous avons remarqué pour les deux jeux que le taux d'accord est très instable et atteint souvent des pics très élevés et des creux très bas. Nous avons donc rajouté une courbe qui montre la tendance générale du taux d'accord en fonction du nombre de parties.
Nous avons aussi lissé la courbe des données pour Othello pour qu'elle soit plus visible. Ceci a pour effet de montrer des pics à 100%, ce qui n'est pas exactement le cas, même si on s'en approche.





Cela nous permet de voir que nous avons un pic de taux d'accord toutes les 700-1000 parties environ pour Awélé (avec un pic initial autour de 100 itérations) et toutes les 200 parties environ pour Othello (à partir du pic initial à environ 100 itérations). Il n'est donc pas vraiment intéressant, avec notre méthode d'apprentissage, de faire plus que ~100 itérations, car les résultats ne s'améliorent plus trop à partir de ce nombre, même si nous avons parfois des pics après où le taux d'accord s'approche de 100%.