# Rapport du projet en 2I013

UNG Thierry

## I - Othello:

On remarque que les stratégies qu'on peut retrouver parmi les joueurs d'othello peuvent être implémentées en privilégiant certaines cases plutôt que d'autres. En effet, par exemple pour s'assurer d'avoir une mobilité importante, ou d'avoir des pions stables, c'est à dire des pions qui ne pourront pas par la suite être enlevés, il suffit par exemples de prendre les cases qui sont au coins. C'est ainsi donc qu'on à conçu la fonction composée des fonctions suivantes :

**nombres_de_cases_coins(jeu):**

Cette fonction retourne le nombre de pions qui se situent sur les coins après avoir jouer. Étant donné l'importance de s'accaparer des coins, cette fonction sera affectée d'un coefficient de **66**

**nombre_de_mauvaises_cases_X(jeu):**

Cette fonction retourne le nombre de pions qui se situent sur les cases qui sont adjacentes diagonalement aux coins, si ce coin est vide. Occuper ces cases, c'est prendre le risque de voir l'adversaire jouer aux coins et donc retourner l'ensemble des cases qui se trouvent dans toutes les directions à partir du coin. Il est donc logique de lui attribuer un coefficient négatif. On a choisi de lui attribuer **-63**

**nombre_de_mauvaises_cases_C(jeu):**

Cette fonction retourne le nombre de pions qui se situent sur les cases qui sont adjacentes horizontalement et verticalement aux coins. Comme la fonction précédemment définie, on voit tout de suite l'importance de ne pas occuper ces cases. D'où le coefficient de **-60** qu'on lui a attribué

Au delà de ces fonctions, la fonction d'évaluation vérifie d'abord si le coup joué ne nous amène pas dans une situation de fin jeu, si c'est le cas, en fonction de l'éventualité, renvoie un résultat qui prime sur les autres. Et si c'est pas le cas, elle fait appelle aux fonctions définies plus haut.

$$\text{Evaluation(jeu)} = \begin{cases} \text{jeufini(jeu)} & \text{if game.finJeu(jeu)} \\ \sum_i w_i e_i & \text{else} \end{cases}$$

Les coefficients qu'on a eu ont été choisis arbitrairement suite à plusieurs simulations.

**Résultats:**

On a donc testé la fonction d'évaluation avec une profondeur 1 et on a obtenu sur un ensemble de 50 parties contre un joueur aléatoire un pourcentage de victoires de 52**% .**

**Interprétation:**

Le pourcentage obtenu est assez raisonnable mais on a pensé qu'une des faiblesses de cette fonction d'évaluation, c'est le fait qu'elle traite les différents états de la même façon quelle que soit la phase de jeu dans laquelle on est. Par exemple une case du plateau n'a pas la même importance à travers toute la partie.

Pour pallier à ce problème, on a décidé de considérer l'environnement du jeu comme étant un système dynamique.

**Partitionnement des etats de jeu:**

On a remarqué que Othello est un jeu de maximum 60 tours. Cette caractéristique permet de répartir toutes les configurations qu'on peut avoir en 60 grands groupes dans lesquelles les configurations ont en commun le fait qu'elles sont à la même phases de jeu.

Pour obtenir le tour dans lequel on est, on a remarqué que pour un plateau donné, le tour dans lequel on se trouve correspond au **nombre de pions qui se trouvent sur le plateau - 4 .**

On se retrouve donc avec une nouvelle fonction d'évaluation que l'on espère plus performante

**Fonction d'évaluation dynamique:**

On définit cette fois-ci une variable global :
**tour = game.getScores(jeu)[0] + game.getScores(jeu)[1]**
dans la fonction saisieCoup(jeu).

On a finalement donc la fonction suivante:

$$\text{Evaluation(jeu)} = \begin{cases} \text{jeufini(jeu)} & \text{if game.finJeu(jeu)} \\ \sum_i w_i e_i & \text{else} \end{cases}$$

$$\begin{cases} e_1 = \text{Nombre de pions aux coins} \\ e_2 = \text{Nombre de pions aux cases X} \\ e_3 = \text{Nombre de pions aux cases C} \\ e_4 = \text{Score obtenu} \end{cases}$$

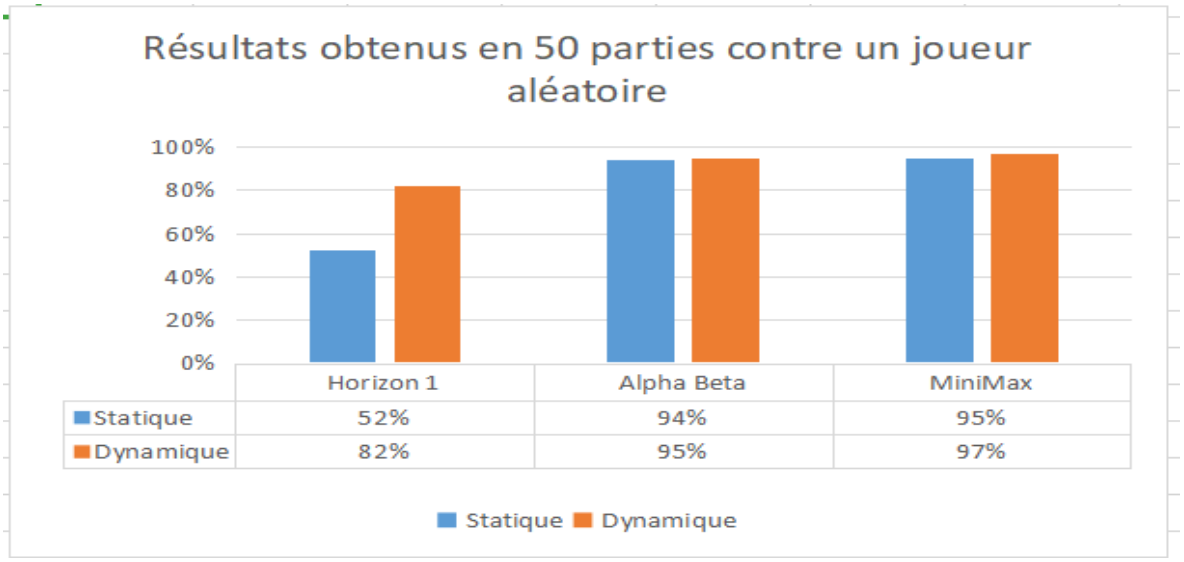
| Variables | Valeur statique | Valeur dynamique |
|---|---|---|
| $W_1$ | 66 | 66 - tour |
| $W_2$ | -63 | -63 + tour |

| | | |
|---|---|---|
| $W_3$ | -60 | -60 + tour |
| $W_4$ | 0 | if t>48 : 1 |

Après avoir effectuer un ensemble de 50 parties, le nouveau joueur avait un pourcentage de victoires de **82 %** comparé au 52**%** obtenu avec le joueur statique.

Résultats obtenus en 50 parties contre un joueur aléatoire

| **Score** | **Horizon 1** | **Alpha Beta** | **MiniMax** |
|---|---|---|---|
| **Statique** | **52%** | **94%** | **95%** |
| **Dynamique** | **82%** | **95%** | **97%** |




## II Awele

Sur le jeu Awele il existe différentes manières de distinguer ce qui est un mauvais coup de ce qui est un bon coup c'est pourquoi nous allons nous intéresser à des éléments différents du jeu afin d'établir le score résultant de la fonction d'évaluation. Pour cela nous avons établis les fonctions suivantes :

**getCasesDes(jeu,joueur):**

Cette fonction retourne le nombre maximale des cases successives ayant seulement 2 ou 3 graines. En effet comme posséder des cases avec seulement 2 ou 3 graines est désavantageux car ces graines peuvent être capturés par l'adversaire, le score que retourne cette fonction sera négatif.

**getCasesAv(jeu,joueur):**

Cette fonction retourne le nombre maximale des cases successives chez notre adversaire ayant seulement 2 ou 3 graines. En effet comme posséder des cases avec seulement 2 ou 3 graines est désavantageux pour notre adversaire car ces graines peuvent être capturés par nous, le score que retourne cette fonction sera positif.

**getNbGraines(jeu,joueur):**

Cette fonction retourne la différence de graine entre le joueur et son adversaire. En effet plus un joueur possède de graine moins il est susceptible de posséder des cases faibles et donc des cases ou les graines risquent d'être prises et inversement moins il a de graine plus il est susceptible de posséder des cases faibles ( cases avec 2 ou 3 graines)

**getFinDePartie(jeu,joueur):**

Cette fonction retourne la valeur de fin de partie, ainsi elle indique le joueur qui a gagné à l'issue de la partie.

**Résultats:**

On a donc testé la fonction d'évaluation avec une profondeur 1 et on a obtenu sur un ensemble de 100 parties contre un joueur aléatoire un pourcentage de victoires de **52%.**
Etant donné qu'il y avait encore de la marge de progression, on a décidé de concevoir un joueur utilisant l'algorithme minimax afin d'avoir une vision plus poussée lors de la prise des décisions.

## L’algorithme minimax

La fonction minimax peut être principalement divisées en 3:

- une partie décision qui renvoie le coup estimé le meilleur
- une partie d’évaluation constituée par l’heuristique définie plus haut
- une partie estimation qui à partir d’un état donné fait une simulation jusqu’à la profondeur désirée.

## Résultats:

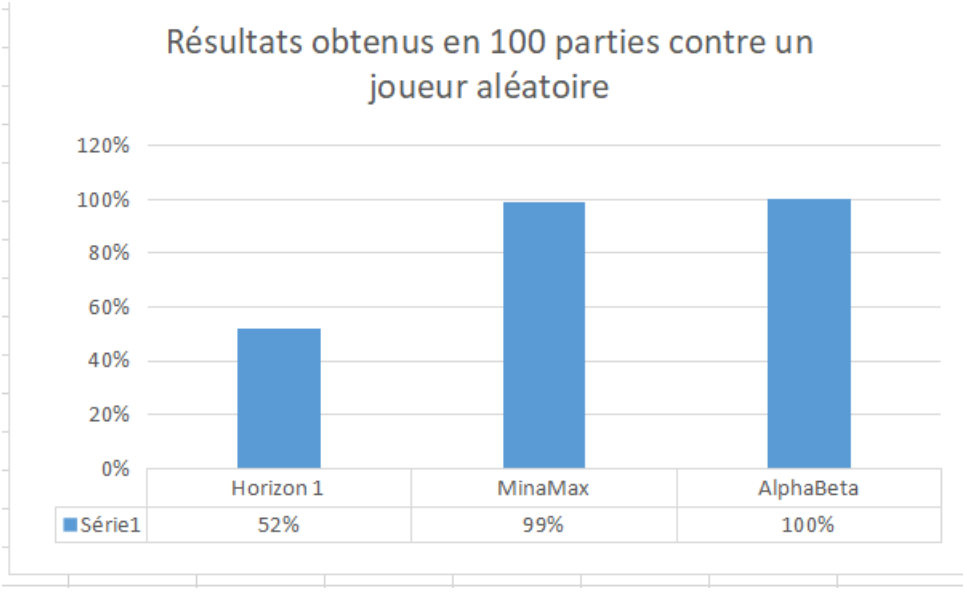
On a donc testé la fonction minimax et on a obtenu sur un ensemble de 100 parties contre un joueur aléatoire un pourcentage de victoire de **99%.**

## L’algorithme Alpha beta

Son objectif reste le même que celui de minimax, sauf que contrairement à minimax, alpha beta permet de faire un élagage permettant ainsi de n’exploiter que les noeuds qui nous seraient utiles

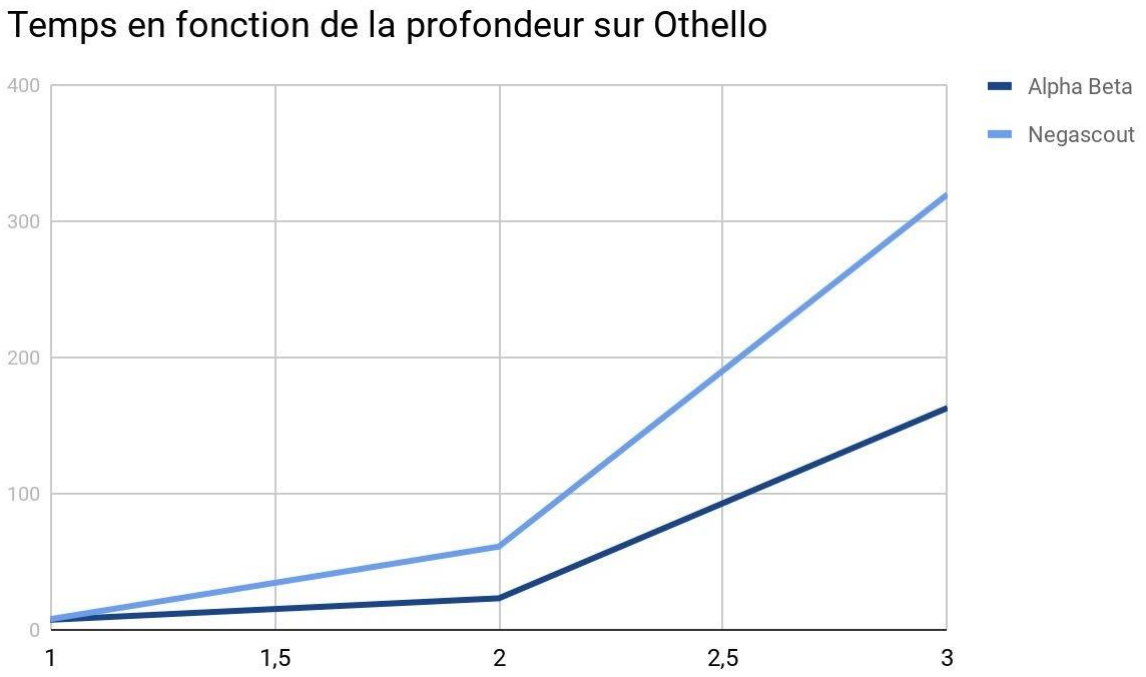
## Résultats:

On a donc testé la fonction alpha beta et on a obtenu sur un ensemble de 100 parties contre un joueur aléatoire un pourcentage de victoire de **100%.**



## III Negascout

Pour améliorer la profondeur de notre joueur pour othello, on a voulu essayé l'algorithme negascout, mais il se trouve ce dernier prend encore plus de temps.
Ce qui peut s'expliquer probablement par l'ordre des actions qui ne se trouve pas dans la configuration la plus optimale. On peut le constater avec les résultats suivants:



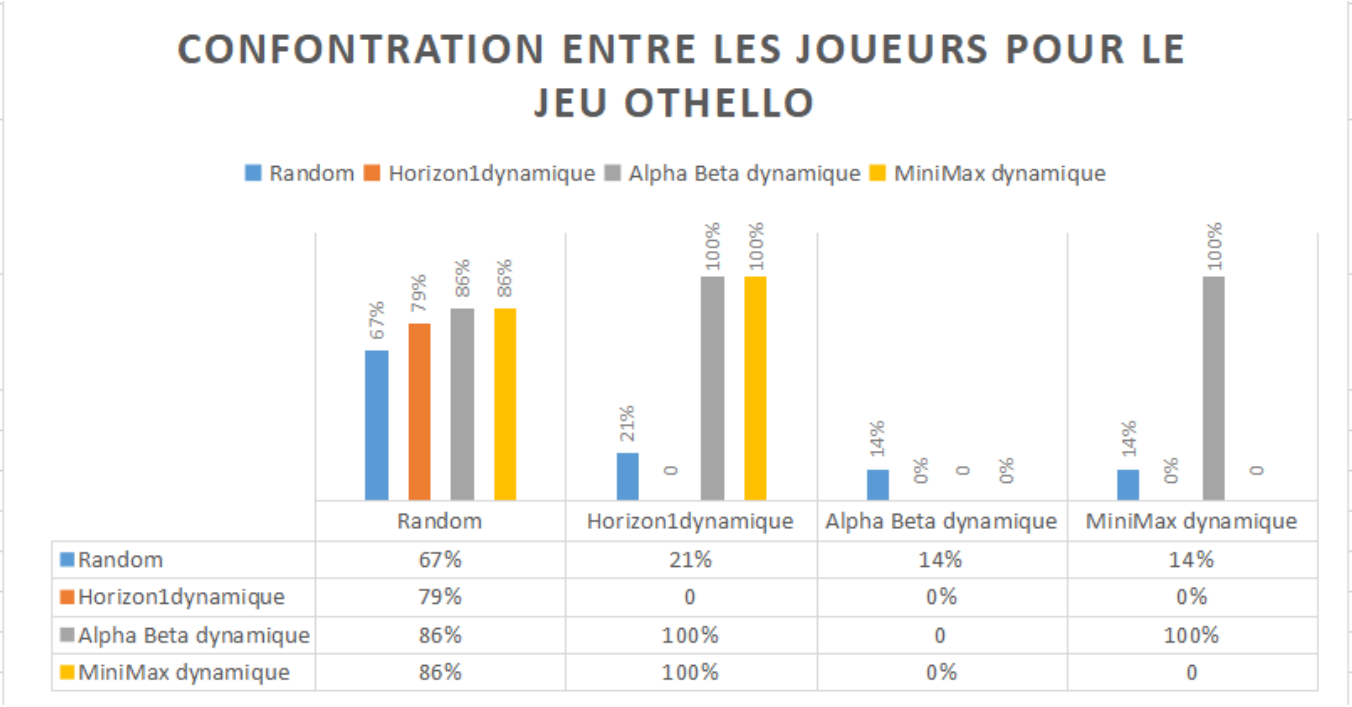

Temps d'exécution (en secondes) en fontion de la profondeur maximale

## IV Résultats finaux:

Les résultats finaux sont établis dans le tableau suivant, pour 100 parties :

Tableau de confontration entre les joueurs pour le jeu Othello

| | Random | Horizon1dynamique | Alpha Beta dynamique | MiniMax dynamique |
|---|---|---|---|---|
| Random | 67% | 21% | 14% | 14% |
| Horizon1dynamique | 79% | / | 0% | 0% |
| Alpha Beta dynamique | 86% | 100% | / | 100% |
| MiniMax dynamique | 86% | 100% | 0% | / |



Confrontation entre les joueurs pour le jeu Awélé

| | Random | Horizon 1 | Alpha Beta | MiniMax |
|---|---|---|---|---|
| Random | 46% | 46% | 0% | 0% |
| Horizon 1 | 54% | / | 0% | 0% |
| Alpha Beta | 100% | 100% | / | 0% |
| MiniMax | 100% | 100% | 100% | / |