Minimax

**Principe de l’algorithme :**

Le principe de cet algorithme est le suivant :

Une **fonction d’évaluation** permet de donner un score a une position donnée d’une partie d’Ultimate Tic Tac Toe. Ce score est établi suivant ce processus : pour le joueur maximisant, une sous grille 3x3 remportée rajoute 50 points au score. Pour les sous grilles non remportées, les cases centrales remportées rapportent 5 points, les coins 3 points et les côtés 2 points. Enfin, une grille 9x9 gagnée rapporte 1000 points. Pour le joueur minimisant, il suffit d’enlever les points au lieu de les rajouter.

A l’aide de cette fonction d’évaluation, l’algorithme va évaluer toutes les positions atteignables jusqu’à une certaine profondeur. En suite il considère comme « meilleur coup » le coup qui permet au joueur d’atteindre la position avec le meilleur score possible en considérant que le joueur adverse va lui aussi jouer le coup qui est le meilleur pour lui à chaque coup.

**Améliorations apportées :**

1. Après avoir joué plusieurs parties avec la première version de l’algorithme, nous nous sommes rendus compte que le fait de privilégier les cases centrales permettait au joueur adverse de facilement remporter la grille 3x3 centrale. Pour remédier à cela, nous avons rajouter la condition suivante : si jouer sur une case centrale permet au joueur adverse de remporter la grille 3x3 centrale au coup suivant, alors on retire 10 points au score de la position si le joueur est maximisant (respectivement ajoute 10 points s’il est minimisant).
2. La deuxième amélioration apportée, appelée **élagage alpha-bêta**, permet à l’algorithme de ne pas explorer certains sous arbres si cela n’est pas utile, c’est-à-dire si on sait que la position à l’origine du sous arbre aura un score plus mauvais qu’une position déjà évaluée. Cette amélioration à permis d’améliorer la rapidité de l’algorithme et donc d’augmenter la profondeur de recherche.

Monte Carlo

**Principe de l’algorithme :**

**AlgoRechercheM\_C:**

* **Meilleur Coup :** On renvoie le coup devant être joué par le joueur qui a lancé méthode d’algo de Recherche. On limite la recherche du meilleur coup grâce à un nombre d’itération et une limite temporelle qu’on fixe à 100ms.

**ArbreMonteCarlo :**

* **Selection** ​permet de sélectionner la feuille de l'arbre MonteCarlo dont on va simuler le reste de la partie. Si la partie est terminée lors de cette étape, l’expansion n’est pas nécessaire, on passe directement à l'étape 'backPropagation'.
* **Expansion** ​permet de créer un nouveau nœud à partir de la feuille choisie précédemment. Si la partie est terminée lors de cette étape, la simulation n’est pas nécessaire, on passe directement à l'étape 'backPropagation'.
* **Simulation** cette fonction simule le reste de la partie à partir d'une configuration (issue de l'étape de l'expansion dans notre cas). Les coups sont joués aléatoirement jusqu’à la fin de la partie. On détermine ensuite le joueur vainqueur.
* **BackPropagation** ​permet de modifier les paramètres des nœuds de la branche en fonction des résultats de la simulation. On modifie le paramètre : nbSimulation et nbVictoire afin d’actualiser les scores UCT.
* **Developper** correspond à un cycle des 4 actions précédemment définies.

**Commentaire :**

Après avoir fini l’implémentation de cet algorithme, nous nous somme aperçu que celui-ci ne fonctionnait pas comme prévu, car il actualisait mal les résultats des parties. Il ne choisissait donc pas le meilleur coup, mais souvent au coup bien inférieur. Nous n’avons malheureusement pas encore réussi à résoudre ce problème.