RAPPORT AFFRONTEMENT ENTRE IA POUR LE JEU « MORPION 4 »

Réalisé par Anthony – Tom – Pierre

ESILV
ENGINEERING SCHOOL
DE VINCI PARIS

ESILV
LÉONARD
DE VINCI

ÉCOLE
D'INGÉNIEURS
PARIS-LA DÉFENSE

Table des matières

Introduction

Dans le cadre d'un projet du second semestre au sein de l'ESILV, nous avons développé une IA qui joue au « Morpion 4 ». Le but est de la rendre suffisamment performante pour lui permettre d'affronter d'autres Intelligence Artificielle et de les vaincre.

Problématique

L'espace de recherche est énorme et l'algorithme peut prendre beaucoup de temps à prédire son prochain coup. Or dans ce genre compétition, la vitesse d'exécution fut le critère principal.

Première étape

```python
class Morpion():
    def __init__(self, numJoueur = 1,plateau=None, etat = None,N = 12, lcj=None):
        self.N = N
        if lcj == None:
            self.listeCoupJoue=[]
        else:
            self.listeCoupJoue = lcj
        if( plateau != None):
            self.plateau = plateau
        else:
            self.plateau = [['.' for i in range(N)] for j in range(N)]
        if(etat != None):
            self.etat = etat
        else:
            self.etat = 1
        self.numJoueur = numJoueur
```

Pour démarrer, nous avons créé une classe « Morpion » qui prend en paramètres un joueur, un plateau de jeu et un état correspondant à l'état de la partie. Ont suivi toutes les fonctions essentielles au bon déroulement de la partie comme la fonction listant tous les coups possibles depuis une situation donnée, une fonction réalisant l'application du coup dans la partie ainsi qu'une fonction pour actualiser le déroulement de la partie.

Solution retenue

Une fois la base de l'algorithme de base **Minimax** créée, nous avons réalisé un algorithme d'élagage **Alpha Beta**, ce qui permet d'éliminer plusieurs milliers de possibilités dont l'exploration est totalement superflue.

```python
def Min_Value(morpion,a, profondeur = 0, alpha = -1000000, beta=1000000):
    global count
    count += 1
    if(morpion.Terminal_Test() or profondeur == 0):
        etatBase = morpion.etat
        morpion.Result(a)
        val = morpion.Utility()
        morpion.UnResult(a, etatBase)
        return val
    score_min = 100000
    etatBase = morpion.etat
    profondeur -= 1
    for a in morpion.Actions():
        score_min = min(score_min,Max_Value(morpion.Result(a),a,profondeur,alpha,beta))
        morpion.UnResult(a, etatBase)
        beta = min(score_min,beta)
        if beta <= alpha:
            break
    return score_min * ((0.8)**profondeur)
```

```python
def Max_Value(morpion, a, profondeur = 0, alpha = -1000000, beta = 1000000):
    global count
    count += 1
    if(morpion.Terminal_Test() or profondeur == 0):
        etatBase = morpion.etat
        morpion.Result(a)
        val = morpion.Utility()
        morpion.UnResult(a, etatBase)
        return val
    score_max = -100000
    etatBase = morpion.etat
    profondeur -= 1
    for a in morpion.Actions():
        score_max = max(score_max,Min_Value(morpion.Result(a),a,profondeur,alpha,beta))
        morpion.UnResult(a, etatBase)
        alpha = max(alpha,score_max)
        if beta <= alpha:
            break
    return score_max * ((0.8)**profondeur)
```

Les Poids

Nous avons identifié 17 possibilités pour la modification de notre poids. Les premiers et les plus importants sont ceux de la victoire ou de la défaite (10000 points). Il faut de plus bloquer les trios ou les duos de l'adversaire tout en créant les propres duos et trios de notre IA. Un poids plus élevé a été attribué à l'action de création d'un trio (XXX ou OOO) tout comme le blocage d'un trio adverse.

Des coefficients ont également été créés pour prendre en compte la profondeur de notre arbre ainsi que pour peaufiner le poids des coups selon que notre IA jouait la partie en premier ou en deuxième.

Les critères qui ont servi à réaliser la version finale du projet sont rassemblés dans le tableau Excel ci-dessous. Ce sont ceux qui nous ont paru les plus pertinents pour améliorer au maximum le raffinage de nos poids en fonction des situations de jeu.

	A	B	C	D	E	F	G	H
1	Coup	Poids Actuel	Refonte				coefAgressive	coef Profondeur
2	Nb Duo bloqué	N	N * 1.5				J1 = 1.2	1
3	Nb trio bloqué	2 * N	2.5 * N				J2 = 0.9	0.8^prodondeur
4	Création Trio	60 - NbPions	80 - nbPions					
5	Création Duo	20 ou 10 - nbPions	30 - nbPions					
6	Bloqué Trio	60 - NbPions	100 - nbPions					
7	Bloqué Duo	20 ou 10 - nbPions	65 - nbPions					
8	Nb voisins	N	N * 1.2					
9	_XXX	500	500					
10	_XX	50	100					
11	X_X	60	150					
12	_X	N	0.5 * N					
13	OXX_	N	0.9 * N					
14	OX-XO	0	0.3 * N					
15	OX-X.	N	N					
16	WIN	10000	10000					
17	LOOSE	-10000	-9500					

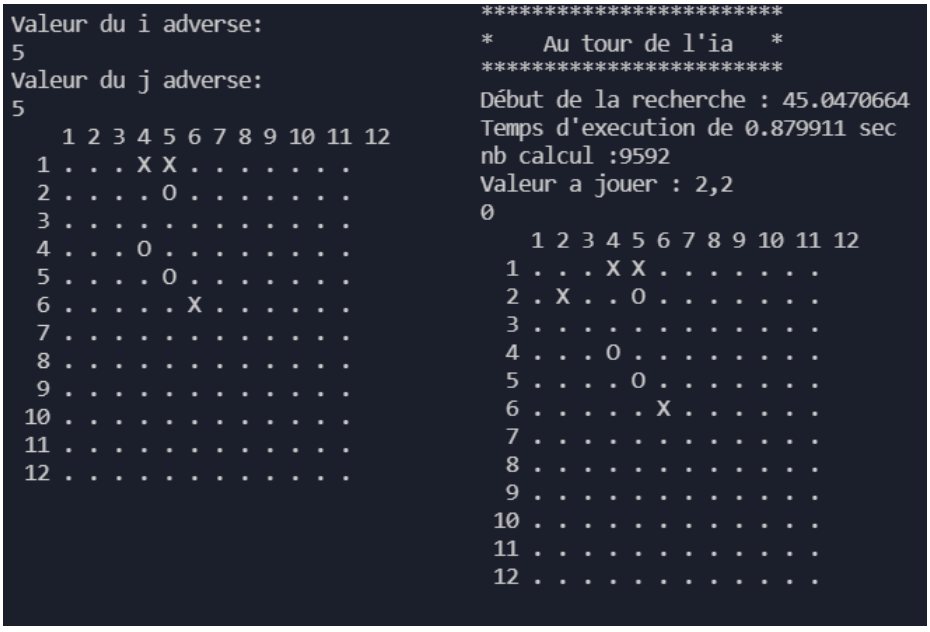
Heuristique

Il y a bien évidemment autant d'heuristiques que de programmeurs. Il fallait donc trouver la stratégie la plus performante. Afin de gagner en vitesse, nous avons décidé de limiter les calculs comme suit : à une action attribuée « a » dans la position (i , j), l'algorithme traite la sous matrice de ± 2 de chaque côté (gauche, droite, haut, bas) au lieu de faire la totalité des calculs sur la matrice 12x12.

Exécution

Notre algorithme a eu une performance tout à fait satisfaisante aussi bien en terme de rapidité que dans le choix des coups lors de la compétition. En effet, notre IA a fini 3ème lors de l'affrontement en classe et s'est montrée la plus rapide quel que soit l'adversaire.

Néanmoins, nous avons constaté lors de la phase finale que notre algorithme privilégiait un peu trop les coups défensifs et ne saisissait pas assez les opportunités de victoire. Cela est dû à un choix de poids trop faible sur le poids permettant de terminer la partie, comme dans l'exemple ci-dessous réalisé peu avant la finale avec le groupe A.



Exécution de l'algorithme en temps réel.

Conclusion

Ce projet fut très enrichissant aussi bien sur le plan humain que sur le plan technique. Il fut l'occasion de débats passionnants dans l'élaboration d'une stratégie gagnante et nous sommes plutôt satisfaits du résultat. Nous effectuerons également pour le plaisir une nouvelle confrontation avec l'IA du groupe A une fois la modification sur notre poids « victoire » effectué afin de prendre notre revanche.