Rapport IA – Puissance 4

Rémi GUILLON BONY

Armand LANGLE

Vincent POUPET

Clément VALOT

Table des matières

Travail sur le Plateau

Le plateau de jeu étant de dimensions 12x6 (12 colonnes et 6 lignes) nous avons opté pour une initialisation du plateau à travers l'utilisation d'une classe (sobrement nommée « Plateau »).

Dans cette classe 4 méthodes sont développées afin de pouvoir échanger des données avec le plateau. Il s'agit de :

- La création : utilisation de la librairie Numpy afin de représenter le tableau plus simplement et proprement,
- L'affichage : surcharge de la méthode __str__() avec différentiation des jetons par leur forme et leur couleur
- Obtention et écriture sur une case du plateau avec la surcharge des méthodes __setitem__() et __getitem__()

Travail sur la fonction Utility

En reprenant le travail que nous avions fait pour l'IA de Morpion nous nous sommes aperçus que si des parties pouvaient être conservées presque telles quelles, d'autres telles l'Utility devaient être intégralement modifiées.

En effet l'enjeu de cette Utility était de pouvoir déterminer la valeur d'un état pour nous, qu'il soit terminal ou non.

Dans un premier temps nous avons créé une Utility simpliste qui nous a permis de nous assurer que le code fonctionnait dans son intégralité. L'objectif n'était pas nécessairement de gagner des parties, mais au moins de réussir à les jouer et ne pas les perdre trop facilement.

La première Utility

Cette Utility était composée de 3 parties. La première était une matrice de la taille du plateau dans laquelle chaque case se voyait attribué une valeur. Cette valeur correspondait au nombre de combinaisons gagnantes faisables avec cette case.

Par exemple la case en bas à gauche avait une valeur de 3 car elle pouvait être utilisée pour faire 1 ligne, 1 colonne et 1 diagonale (vers la droite en haut).

Pour déterminer la valeur d'un état pour nous, nous faisions appel à la deuxième partie de la fonction. Pour chacune des cases alliées sur le plateau, la valeur de sa case était ajoutée à une somme générale. Pour chacune des cases ennemies, cette valeur était retranchée.

On arrivait ainsi à un total qui résumait la valeur de l'état pour nous.

Toutefois cette valeur était purement statistique et ne prenait pas en compte l'état réel du plateau, les valeurs attribuées aux cases étaient fixes. Cela posait problème notamment dans le système de défense de l'IA qui ne se rendait pas compte qu'elle était sur le point de perdre.

Nous avions donc ajouté une troisième partie qui déterminait si l'état était terminal ou non. Si l'état observé faisait gagner notre IA, une valeur infiniment grande était ajoutée à la somme générale. Dans le cas contraire, c'est son équivalent négatif qui était ajouté.

En résumé cette Utility permettait une défense anticipée à 1 tour et une attaque en fonction des valeurs statistiques des cases. Comme nous l'avons dit, une telle IA n'était pas suffisante et pouvait être battue par un humain. Néanmoins elle nous a permis de procéder à des tests notamment vérifier le bon fonctionnement de l'élagage alpha-bêta ainsi que des autres fonctions usuelles de notre système.

Après avoir observé que ces tests étaient concluants, nous nous sommes lancés sur le développement d'une nouvelle fonction Utility, plus performante et plus agressive.

La seconde Utility

Le principal point faible de notre Utility précédente était la nature statique des valeurs de sa matrice d'évaluation. Nous avons donc concentré nos efforts sur ce point là en voulant créer une IA qui adapterait ses plans de jeu en fonction de l'état réel du plateau.

Afin de développer cette Utility, nous avons nous-même joué contre une IA sur une Internet et noté à la main les différentes réflexions que nous avions avant de jouer un coup. C'est à l'aide de ces réflexions que nous avons dégagé des règles générales que nous avons établi en algorithme.

Ces règles sont les suivantes :

- La valeur d'une case est définie par la somme des possibilités qu'elle m'offre en fonction du plateau actuel
- Une possibilité est définie comme un alignement de 4 pions alliés potentiellement réalisable (pas de pion ennemi dans cet alignement à l'heure actuelle)
- Chaque possibilité d'alignement est pondérée par le nombre de pions alliés déjà présents dans cet alignement. Cette pondération n'est pas linéaire afin de prioriser l'importance d'un alignement de 4.

Par exemple une possibilité d'alignement de 4 avec un seul pion allié bien placé vaudra moins qu'une possibilité d'alignement de 4 où deux pions alliés sont déjà présents. En effet on considère que ce deuxième cas est plus facile à transformer en alignement gagnant.

- Un alignement de 3 pions où il suffit de placer le 4e pour gagner immédiatement et un alignement de 3 pions où il faut combler un trou de 4 de profondeur pour remporter la partie ne doivent pas avoir une valeur équivalente. Il faut donc pondérer la valeur d'une possibilité par le nombre de cases à combler avant de pouvoir la réaliser.

La formule finale pour calculer la valeur d'une case est la suivante :

$$\text{Valeur case} = \sum^{chaque-alignement-réalisable} \frac{nb-pion-déjà-alignés^2}{nb-cases-à-combler+1}$$

La seconde partie reste la même : pour chaque case du plateau occupée par un pion, on mesure la valeur de la case et on l'ajoute (ou la retranche) à la somme générale en fonction de son propriétaire. Notre capacité défensive est ainsi équivalente à notre capacité offensive. Avec une profondeur de 6, nous attaquons et nous défendons avec 3 tours d'avance.

La troisième partie que nous utilisions dans la première Utility est devenue obsolète car avec la nouvelle version la défense se fait de la même manière que l'attaque.

Afin d'évaluer cette nouvelle Utility, nous avons fait jouer notre IA face à son ancienne version d'elle-même. Jusqu'à présent la nouvelle version n'a pas perdu un seul match et en a même gagné la majorité.

Travail sur l'élagage en amplitude et recherche des paramètres

Afin de réduire le temps de réflexion de notre IA à chaque tour, nous utilisions évidemment un système de limite de profondeur afin d'éviter qu'elle descende trop bas dans l'arbre. Mais afin de gagner encore plus de temps nous avons aussi mis en place un élagage en amplitude. Plutôt que d'explorer chacune des 12 possibilités à chaque niveau, nous utilisons un paramètre qui limite ce nombre à 50%, 60%, …

Son fonctionnement est simple : plutôt que s'enfoncer d'un niveau supplémentaire pour chaque action déterminée dans nos fonctions Min et Max, l'Utility du résultat de chacune de ses actions est évalué. Pour les x% ayant la valeur d'Utility la plus élevée du lot, on s'enfonce effectivement (x étant un paramètre). Les autres actions sont abandonnées.

L'économie de temps fournie par cet ajout semble exponentielle. Pour un temps de recherche de 30s à x = 100%, on passe à 15s pour x = 90%.

Ce filtrage des actions inintéressantes est basé sur l'assomption suivante : On considère ainsi que les états permis par une action inintéressante à la profondeur Y seront en moyenne moins bons que ceux d'une action intéressante à la profondeur Y.

Nous nous sommes ensuite demandé dans quelle mesure nous pouvions diminuer la valeur de x afin de gagner plus de temps tout en ne perdant pas en efficacité. Pour cela nous avons effectué un certain nombre de matchs en faisant varier la profondeur et l'amplitude.

Détails des matchs

Une fois notre utility définie ainsi que l'élagage sur la largeur implémenté, nous avions 2 paramètres à notre disposition pouvant faire varier le comportement de notre IA : la profondeur maximale pouvant être atteinte pour la recherche d'un coup et le pourcentage d'élagage sur la largeur pour chaque ensemble de coups disponibles.

Nous avons utilisé le fichier Optimisation.py afin de procéder à des tests.

Nous devions donc faire varier tous ces paramètres afin d'en dégager la meilleure combinaison. Nous avons choisi de faire varier la profondeur de 4 à 8 ainsi que l'élagage de 40% à 70% avec un pas de 5%. Ce qui donne un total de 210 matchs. Nous avons fait jouer notre IA contre elle-même sur ces 210 matchs et avons noté pour chaque match qui avait commencé, qui avait gagné et si l'une d'elles avait été trop lente (temps de réflexion supérieur à 8 secondes).

Enfin, une fois tous ces matchs terminés des champions par profondeur se sont dégagés. Nous avons donc effectué des combats entre ces champions (42 matchs supplémentaires) afin de trouver notre champion absolu qui nous donnerait ses paramètres comme paramètres de références pour les affrontements contre les autres IA de la promo.

Il en est ressorti que les paramètres : profondeur = 6 && largeur = 0.45% étaient les plus prometteurs.

NB : Pour voir le résultat de chaque match, se référer au document :
« Optimisation-Puissance4_Projet-IA_A3.csv »

Cependant nous avons proposé à d'autres équipes d'effectuer des matchs amicaux afin de tester notre IA et nos paramètres contres d'autres IA n'ayant pas la même heuristique. Après certains matchs (gagnés et perdus) nous avons réajusté nos paramètres afin de respecter le temps imparti à chaque coup (environ 10 secondes par coup) tout en garantissant une IA solide. Il est en ressorti que les meilleurs paramètres étaient :

Profondeur = 5 && Largeur = 60%

Remarques sur l'amplitude

Après avoir effectué ces différents matchs, nous avons trouvé des couples de paramètres qui nous semblaient intéressants.

Toutefois après avoir joué face à une autre équipe, nous nous sommes rendu compte que trop diminuer la valeur de x réduisait drastiquement la qualité de notre IA.

Après avoir joué la même partie en boucle face à l'IA adverse en changeant à chaque fois la valeur de x, nous avons remarqué que les plays recommandé par notre IA en x=0.4 étaient rapidement différents de ceux recommandés en x = 0.5. Nous en avons conclu que la valeur x=0.4 ne permettait pas à notre IA de voir les meilleurs plays (le filtre était trop large).

Nous avons répété l'expérience avec x=0.5 et x=0.6, x=0.6 et x=0.7, etc.

Nous en avons conclu que la qualité des plays semblait se stabiliser à partir de x=0.6

C'est pourquoi nous avons fixé la valeur de x à 60% dans le reste de nos évaluations et de nos matchs.