Rapport de Projet

INTELLIGENCE ARTICIELLE

Le jeu de l'Awalé et de son Bot Hawaleluja

Réalisé par
KREUTNER—BARATEAU Gwenaël

2021-2022

Table des matières

Introduction

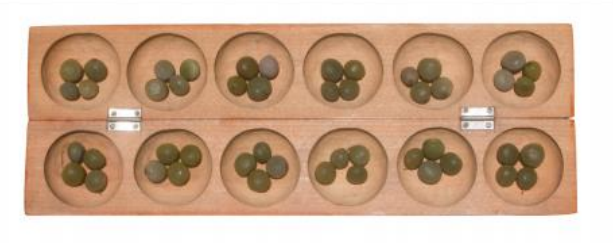
Pour ce projet d'intelligence d'artificielle, il nous a été demandé de coder un bot capable de jouer une partie d'Awalé, c'est-à-dire de prendre une décision sur le prochain coup à jouer. Appartenant à la grande famille Mancala, l'Awalé est un jeu de société combinatoire abstrait d'origine africaine. Le principe de ce jeu est l'affrontement de deux joueurs avec l'objectif de ramasser un plus grand nombre de graines que son adversaire.

1 Le Projet

1.1 Les règles de l'Awalé

Avant toute chose, j'ai tout d'abord pris connaissance des règles du jeu d'Awalé pour pouvoir commencer à travailler sur ce projet.

L'Awalé est un jeu pour 2 joueurs avec des règles simples mais de nombreuses stratégies. C'est l'un des plus anciens jeux de société au monde et il est connu sous de nombreux noms différents, l'un d'eux étant Kalaha.



Comme nous pouvons le constater ci-dessus, le plateau de jeu se compose de 2 rangées, avec 6 trous dans chaque rangé. Au début du jeu, chaque trou contient 4 pierres.

Le joueur à tour de rôle prend toutes les pierres d'un de ses trous et les distribue dans le sens antihoraire, une dans chaque trou, quel que soit le propriétaire du trou. Si plus de 11 pierres (Krou) doivent être distribuées, plus d'une boucle autour du plateau est nécessaire, mais le trou d'où les pierres ont été prises reste vide.

Si la dernière pierre distribuée tombe dans le trou d'un adversaire, qui contient alors 2 ou 3 pierres, le joueur à son tour réclame ces pierres. Lorsque cela se produit, le trou précédent est également examiné de la même manière. Cette procédure est répétée jusqu'à ce que le propre trou d'un joueur doive être examiné ou jusqu'à ce que le trou d'un adversaire contienne moins de 2 ou plus de 3 pierres, selon la première éventualité. Dans les deux cas, les pierres restent dans ce trou.

Il est interdit d'affamer son adversaire c'est-à-dire qu'un joueur ne peut pas se retrouver avec 0 graines de son côté. Un coup permettant de prendre toutes les graines chez l'adversaire est possible mais les points qu'ils devraient rapporter ne sont pas pris en compte.

L'objectif de ce jeu est d'obtenir 25 points ou plus ce qui met fin à la partie. Une égalité est possible si les 2 joueurs ont 24 points à la fin du jeu.

1.2 Objectif et méthodologie

Il nous a été demandé de mettre en place un modèle de prise de décision permettant de choisir de manière optimal le meilleur coup à jouer en fonction d'une situation donnée. Des contraintes ont été également imposés.

Après avoir analyser le sujet et pris en main les règles du jeu d'Awalé. Dans un premier temps, j'ai lu les rapports de projet des précédentes années fournis par le professeur qui m'ont guidé dans mes recherches et permis de progresser assez rapidement dans le développement de mon Bot. Ensuite, après implémentation de mon modèle, j'ai fait mes propres recherches sur des heuristiques qui pourraient être ajoutées afin d'améliorer mon algorithme.

D'après les rapports précédents, l'algorithme du MiniMax semblait être le plus efficace, je me suis donc lancer sur son implémentation en me basant sur celui fourni par le professeur.

2 Minimax

2.1 Principe général

L'algorithme MiniMax ou MinMax est un algorithme d'exploration d'arbre en profondeur d'abord. Il est bien adapté au jeu d'Awalé car c'est jeu à deux joueurs à somme nulle et à information complète. L'objectif de cette algorithme est de minimiser la perte maximale. Il utilise une structure d'arbre avec des nœuds qui représente des maximums et des minimums. Le but est de calculer tous les coups possibles. Chaque joueur cherche à maximiser son score, ce qui revient à minimiser celui de l'adversaire.

De base, la profondeur est limité à 6 car le temps de calcul pour parcourir tous les nœuds est exponentiel en fonction de la profondeur. Ainsi, pour explorer plus de solutions dans l'ensembles des coups et obtenir une solution optimal, on doit augmenter la profondeur de la recherche. C'est pour cela, que j'ai dû modifié l'algorithme (MinMaxNode) fourni par le professeur en mettant en place un élagage.

2.2 Élagage Alpha-Beta

L'algorithme Alpha-Beta est une variante du MiniMax. Elle permet de couper des branches de l'arbre lors de l'exploration sans changer le résultat de l'évaluation. En effet, il n'est pas nécessaire de parcourir les nœuds fils qui conduisent à des nœuds de score inférieur ou déjà évalué.

On va donc ajouter 2 nouvelles variables, α et β qui représentent respectivement le score minimum que peut avoir le joueur Max et le score maximum que peut avoir le joueur Min.

L'élagage Alpha-Beta réduit considérablement la taille de l'arbre donc le temps de calcul également. Mais dans le pire des cas, on peut parcourir l'arbre dans son entièreté. C'est pourquoi, j'ai ajouté une optimisation supplémentaire pour l'exploration de l'ordre des nœuds fils.

2.3 Optimisation

Pour rester dans les temps impartis (200ms par coup) pour une décision. Avec l'appui du rapport de 2019, j'ai ajouté un système de tri des coups ce qui va me permettre d'agrandir la profondeur d'exploration lors d'une décision pour un coup d'un duel.

Pour ce faire, on va trier les coups dans un certains ordre afin d'éviter un maximum de nœuds à parcourir ce qui rendra l'algorithme Alpha-beta plus performant.

Dans un TP(Tic-Tac-Toe) vu avec mon professeur, nous avons utilisé une liste des nœuds déjà visités car plusieurs chemins peuvent mener vers des nœuds identiques (identique à une symétrie près). J'ai donc, pour chaque coup du jeu, calculé sa valeur selon l'état du jeu(voir formule ci-dessous) et ainsi la ranger dans une liste contenant une note correspondante au type de coup. Et lorsqu'un coup similaire se représentera, on évitera de parcourir entièrement la suite de la branche.

*Categorie(c) = TrouDepart(c) + 6 * GraineTrouDepart(c) + 6 * 48 * GraineTrouArrive(c)*

Avec c = coup

Après le calcul des catégories, on trie dans l'ordre décroissant l'index et on va parcourir les nœuds fils selon le nombre de fois que l'on est déjà tomber sur le coup. C'est dans l'intérêt d'explorer plus rapidement l'arbre et ainsi augmenter la profondeur.

Suite à ces améliorations d'élagages sur Alpha-Beta, j'ai cherché d'autres techniques pour améliorer mon algorithme. Je suis donc parti sur l'ajout d'heuristiques pour que mon algorithme prenne une meilleur prise de décision en fonction de l'état du jeu.

2.4 Heuristiques

Les heuristiques ont pour but d'améliorer l'évaluation de la situation du plateau du jeu grâce à un ensemble de coefficients. Une bonne connaissance des règles du jeu et de recherche de techniques du jeu est nécessaire pour développer une heuristique. J'ai donc recherché des heuristiques qui pourraient améliorer mon bot, j'en ai retenu 4.

- Heuristique Score Individu :
  Dans un premier temps, j'ai implémenté un coefficient pour le score du joueur ainsi que de l'adversaire fixé respectivement à -10 et 10.
- Heuristique Krou :
  Une technique revenant souvent est celle du Krou. Elle consiste à l'accumulation dans un trou d'un nombre de graines suffisant pour faire un tour complet, soit au moins 12 graines. Il permet de marquer beaucoup de points en un coup. C'est dans le but de pouvoir jouer au moins un tour complet et tenter de faire des captures multiples chez l'adversaire.
- Heuristique 1 ou 2 Graines dans une case :
  Les cases de 1 ou 2 graines permettent de marquer des points directement.
- Heuristique Trou Vide :
  Les cases vides permettent d'avoir une information sur le nombre de points pouvant être marqué dans un futur proche

En faisant la somme pondéré des heuristiques avec un poids propre à chaque valeur, l'algorithme prend de meilleurs décisions. Mais trouver la bonne valeur des coefficients est difficile pour l'être humain. C'est pourquoi, trouver des coefficients les plus justes possibles pour la prise de décision est important, j'ai donc appliqué un algorithme évolutionnaire pour l'apprentissage de mes coefficients.

3 Apprentissage

3.1 Algorithme évolutionnaire

Grâce à l'option MetaHeuristique que j'ai choisi pour ce semestre, j'ai pu apprendre différents algorithmes d'optimisation visant à résoudre des problèmes d'optimisation difficile pour lesquels on ne connait pas de méthode classique plus efficace. Pour que mon algorithme puisse trouver par lui-même les coefficients les plus justes, j'ai fait le choix de faire mon apprentissage sur un algorithme génétique.

3.1.1 Description

Les algorithmes génétiques sont des algorithmes s'inspirant des théories de l'évolution. Dans ces algorithmes, une population d'individus évolue afin d'optimiser un problème. Le génome de chaque individus sert à représenter une solution au problème. Des opérateurs de sélections, de croisement et de mutation permettent de renouveler cette population avec de nouveaux individus ayant en théorie un meilleur génome.

3.1.2 Configuration

Ma population est constituée de 40 bots. D'après mes tests, ce nombre d'individus m'a globalement renvoyé les meilleurs résultats, même si les différences étaient plutôt faibles.

3.1.2.1 Initialisation des individus

Pour avoir un algorithme génétique efficace, j'ai initialisé un individu dans ma population avec des coefficients fixé et j'ai complété ma population de bots avec des coefficients générés aléatoirement.

3.1.2.2 Multi-Élitisme

J'ai créé une méthode qui permet de garder un nombre N de bots de la génération actuelle vers la population suivante. Ces bots correspondent aux meilleurs bots de notre population. J'ai fixé N à la moitié de l'ensemble c'est-à-dire 40/2 = 20.

3.1.2.3 Mutations

L'opérateur de mutation va faire muter un bot c'est-à-dire qu'un de ces gènes (coefficients dans notre cas) est changé de manière aléatoire.

3.1.2.4 Croisement

Le croisement est très important dans un algorithme génétique. C'est cet opérateur qui créé la descendance qui va remplacer la population actuelle. Il créer un enfant ou plus à partir de 2 parents. Dans mon cas, il créé 1 enfant.

3.1.2.5 Sélection par tournoi

Afin de choisir quels bots de la population j'allais faire muter et croiser, j'ai implémenté une sélection par tournoi. Je sélectionne un certain nombre d'individus, dont le meilleur, qui seront les individus à battre. Le reste de la populations va donc se battre contre eux

3.2 Résultats

Pour l'algorithme définitif, j'ai mis en place l'algorithme Hawaleluja. C'est un algorithme Alpha-Beta avec des amélioration et d'une profondeur d'exploration à 8.
Pour l'apprentissage je passe à une profondeur de 6 pour réduire le temps d'une génération et donc augmenter son nombre.

Conclusion

J'ai trouvé que ce projet était très intéressant pour se familiariser avec l'intelligence artificielle. En effet, c'est un domaine qui me porte très à cœur et j'ai pu vraiment progresser durant ce projet. J'ai pu constater la robustesse d'un algorithme bien configurer pour l'apprentissage d'un bot. Ainsi que la force des algorithmes d'exploration comme Minimax avec l'élagage Alpha-Beta qui le rend encore plus performant. Je me suis également rendu compte de l'importance de bien structurer son code ainsi qu'optimiser mes algorithmes afin d'améliorer le plus possible la performance de mon algorithme.

Bibliographie

1. SAILLOT Mathis, J'ai Awalé de Travers, 2020

2. LELLEMENT Théo & MASKIO Alexis, Teluob Bagarre, 2021

3. Stratégies de jeu pour l'Awalé : https://www.myriad-online.com/resources/docs/awale/francais/strategy.htm