Compte-Rendu AI-Game Programming
Awalé

Réalisé par:
LEVESQUE Mathis, DOGAN Yunus Emre

1. Introduction

Dans ce projet, nous avons implémenté une intelligence artificielle pour le jeu Awalé (variante à 16 trous avec graines rouges, bleues et transparentes).
L'objectif était de développer une IA capable de choisir de bons coups à l'aide de l'algorithme **Minimax avec élagage alpha-bêta**, et surtout de concevoir une **fonction d'évaluation (heuristique)** pertinente pour estimer la qualité d'un état du jeu.

Le cœur du travail a donc porté sur la comparaison entre une heuristique simple et une heuristique plus avancée, ainsi que sur l'analyse de leur impact sur le comportement de l'IA.

2. Heuristique simple : évaluation basée uniquement sur le score

Dans un premier temps, nous avons utilisé une heuristique très simple :

scoreDiff = scoreIA - scoreAdversaire;

Cette heuristique consiste uniquement à comparer le nombre de graines capturées par l'IA et par l'adversaire.

2.1 Avantages

- Simple à implémenter
- Facile à comprendre
- Permet de vérifier que l'algorithme Minimax fonctionne correctement

2.2 Inconvénients

- Ne prend pas en compte la position des graines sur le plateau
- Ne prévoit pas les captures futures
- Ne tient pas compte du risque de starvation
- Donne un comportement souvent trop passif ou naïf

Cette heuristique est donc utile comme référence de base, mais insuffisante pour jouer de manière stratégique.

3. Heuristique avancée : évaluation positionnelle et stratégique

Afin d'améliorer la qualité de jeu de l'IA, nous avons développé une heuristique plus complète, combinant plusieurs critères :

3.1 Différence de score

La différence de score reste le critère principal, car l'objectif final du jeu est de capturer plus de graines que l'adversaire.

*120 * scoreDiff*

Un coefficient élevé est utilisé pour donner une importance prioritaire au score réel.

3.2 Potentiel de capture

Nous avons ajouté une estimation du potentiel de capture :

- **captureIA :** potentiel de capture pour l'IA
- **captureOpp :** potentiel de capture pour l'adversaire

Cela permet à l'IA :

- d'anticiper des gains futurs
- d'éviter de laisser de grosses captures à l'adversaire

*+ 35 * captureIA*

*- 45 * captureOpp*

3.3 Starvation (blocage de l'adversaire)

Le règlement autorisant le starvation, nous avons introduit un bonus lorsque l'adversaire a peu ou pas de coups possibles :

if (oppMoves == 0) starvation += 250;

else if (oppMoves == 1) starvation += 120;

else if (oppMoves == 2) starvation += 60;

L'idée est d'encourager l'IA à mettre l'adversaire sous pression sans pour autant sacrifier trop de points.

3.4 Tempo

Le joueur dont c'est le tour possède souvent un léger avantage tactique.

Nous avons donc ajouté un petit bonus/malus :

tempo = (joueurActif == ia) ? +30 : -30;

3.5 Cases vulnérables

Certaines configurations de trous vides peuvent faciliter des captures adverses.

Nous pénalisons donc les positions où l'IA possède des cases vulnérables :

*- 25 * vulnerable*

3.6 Fin de partie (endgame)

Lorsque le nombre de graines restantes diminue, le jeu se rapproche de la fin.

Nous avons donc introduit un bonus progressif :

*endgame = (100 - grainesRestantes) * 3;*

4. Tests et comparaison des paramètres de starvation

Nous avons testé plusieurs configurations pour le poids du critère de starvation.

4.1 Cas 1 : Starvation modéré (250 / 120 / 60)

Dans cette configuration :

- Joueur 2 (nouvelle heuristique) a battu Joueur 1 (ancienne heuristique) 10–6
- Le jeu était plus équilibré
- Les écarts de score restaient raisonnables
- L'IA privilégiait à la fois la capture et le contrôle du plateau

4.2 Cas 2 : Starvation très élevé (500 / 250 / 120)

Avec ces valeurs :

- Joueur 1 a remporté la série 10–9
- L'IA jouant avec la nouvelle heuristique cherchait trop souvent à affamer l'adversaire
- Certaines parties présentaient des écarts très importants
- Le comportement devenait parfois trop agressif ou risqué

5. Profondeur dynamique

Afin d'améliorer le compromis entre la qualité des décisions de l'IA et le temps de calcul, nous avons choisi d'adapter dynamiquement la profondeur de recherche du Minimax en fonction du nombre de coups possibles dans la position courante.

Dans le jeu Awalé, le nombre de coups légaux peut varier fortement selon la phase de la partie :

- En début et milieu de partie, le plateau contient beaucoup de graines et le nombre de coups possibles est élevé.
- En fin de partie, le nombre de graines diminue, ce qui réduit fortement le nombre de coups légaux.

Nous avons donc utilisé la règle suivante :

- Si le nombre de coups possibles est supérieur à 5, la profondeur est fixée à 5.
- Sinon, lorsque le nombre de coups possibles est inférieur ou égal à 15, la profondeur est augmentée à 7.

Cette stratégie permet :

- de limiter le coût de calcul lorsque le facteur de branchement est élevé

- d’explorer plus profondément l’arbre de recherche dans les positions critiques de fin de partie ;
- d’améliorer la précision des décisions lorsque les captures et le starvation deviennent déterminants.

Ce choix s’est avéré pertinent lors des tests, car il renforce la qualité du jeu de l’IA en fin de partie sans pénaliser excessivement les performances en début de partie.

6.Conclusion

Ce projet montre que la qualité d’une IA ne dépend pas uniquement de l’algorithme Minimax, mais surtout de la fonction d’évaluation utilisée.

- Une heuristique simple est utile pour valider le fonctionnement de l’IA
- Une heuristique avancée permet un jeu plus stratégique et plus réaliste
- Le réglage des coefficients est crucial et influence fortement le comportement de l’IA
- Les tests comparatifs sont indispensables pour évaluer la pertinence d’une heuristique
- Ce travail nous a permis de mieux comprendre l’importance de la modélisation stratégique dans les jeux à information parfaite et le rôle central des heuristiques dans les algorithmes de recherche.