Université de Lorraine

Rapport de projet d'IA : Awélé

L'AwélibertéGuidantLePeuple

Robin LAGLER | Thomas FELKER
21/03/2023

Table des matières

Introduction

L'Awélé

L'Awélé est un jeu de société combinatoire abstrait d'origine africaine faisant partie de la famille des jeux mancala et se jouant sur un plateau en bois ou en métal, composé de deux rangées de six trous, avec des graines, des pierres ou des coquillages dans chaque trou.

Figure 1 : Plateau de jeu d'Awélé, source : www.afrik.com

Le but du jeu est de capturer les graines de l'adversaire en les déplaçant à tour de rôle autour du plateau, généralement dans le sens inverse des aiguilles d'une montre. Le joueur qui réussit à capturer le plus de graines à la fin de la partie remporte la partie.

L'Awélé est un jeu de stratégie simple mais très captivant, pouvant être joué par des joueurs de tous niveaux et nécessitant de la réflexion ainsi que de l'observation afin de réussir à prévoir les coups de son adversaire et à prendre les meilleures décisions pour optimiser ses gains. C'est un jeu très populaire en Afrique de l'Ouest et en Afrique centrale, ainsi que dans certaines parties d'Asie et des Caraïbes.

Le projet

Le sujet.

L'objectif de ce projet est de créer un moteur d'intelligence artificielle capable de jouer à l'Awélé en utilisant un ou plusieurs algorithmes d'intelligence artificielle. Cet agent devra prendre des décisions efficaces en fonction de la situation actuelle de jeu, en se basant sur une base de données de coups joués, sans que les décisions ne soient directement codées.

Différences avec le jeu officiel.

Les règles du jeu employées dans le cadre de ce projet diffèrent légèrement des règles "officielles" du jeu sur la détermination d'une fin de partie ; en effet, dans les compétitions européennes, la fin d'une partie n'est pas déterminée par le nombre de graines mais est réglementée selon 2 cas possibles :

1. S'il y a une situation de **famine** : si l'adversaire ne possède plus aucune graine et que le joueur actuel se trouve dans l'impossibilité de jouer un coup qui redonne des graines à l'adversaire, alors la partie s'arrête et le joueur conserve les graines de son côté ;
2. S'il y a une situation de **boucle** : si les positions se répètent en boucle, alors les joueurs peuvent décider d'arrêter la partie et se répartissent les graines.

Recherches

Préambule.

Xavier Blanvillain et 48stones.

Xavier Blanvillain, vice-champion d'Europe d'Awélé en 2000 et fondateur du site 48stones, est un acteur important du monde de l'Awélé européen de par son investissement dans ce milieu. Il œuvre afin de réunir le monde scientifique et le monde du jeu de l'Awélé, ainsi que pour permettre aux meilleurs joueurs de s'entraîner dans les meilleures conditions.

48stones est également le nom de la base de données de 1,7Tb créée par Xavier Blanvillain et contenant toutes les positions possibles, soit 827 240 309 058. Grâce à ses recherches et à cette base de données, remplie avec un seul ordinateur et un algorithme multithread de force brute, Xavier Blanvillain a réussi à prouver qu'une partie se termine par un match nul si aucun des deux joueurs ne commet la moindre erreur.

Joan Sala et Aualé.

Joan Sala Soler est un ingénieur logiciel et développeur web full-stack espagnol. Il est également le concepteur et unique développeur derrière l'interface graphique Aualé ainsi que son moteur d'intelligence artificielle Aalina, détentrice actuelle du record du meilleur score effectué par une IA contre la base de données 48stones, à savoir une victoire de 23-25 pour 48stones, soit le même score que celui effectué par le champion international d'Awélé Trevor Simon.

Actuellement, il prévoit de construire une version améliorée d'Aualé utilisant le moteur de jeu Unity et de la rendre disponible gratuitement sur diverses plateformes. Il dispose d'un prototype fonctionnel proposant de nouvelles fonctionnalités telles que de nouveaux niveaux de difficulté, des modes de jeu, un mode analyse et des intégrations de réseaux sociaux, et il prévoit pour le futur d'inclure un éventuel mode de jeu en ligne.

Les conseils reçus.

Nos recherches nous ont menées à la découverte de ces deux personnes influentes dans le monde de l'Awélé informatique, et c'est dans le cadre de celles-ci que Robin a décidé de les contacter tous les deux afin de leur demander des informations et des conseils.

Xavier Blanvillain.

Dans cette démarche, Robin a pu s'entretenir avec Xavier Blanvillain dans un premier temps par mail puis via une visio-conférence. Voici un bref résumé des conseils reçus par ce dernier lors de ces échanges :

- Savoir gérer nos "**Move in hands"** : également appelés "coups tranquilles", il s'agit des actions lors desquelles le joueur n'envoie aucune graine chez son adversaire. Il existe une seule position fournissant 26 coups avant de pouvoir donner une seule graine à l'adversaire. ;
- **Toujours** commencer la partie par un coup à droite : avec 48stones, Xavier Blanvillain a réussi à prouver que, si aucun des joueurs ne commet d'erreur lors de la partie, alors si le joueur qui a commencé la partie a commencé à droite la partie se soldera par un match nul, sinon elle se conclura sur une défaite du premier joueur (voir figure 1) ;
- **Doubles cases dangereuses** : il s'agit là du nom que Xavier a donné à la règle permettant de récupérer des graines pour les ajouter à son score. Xavier Blanvillain nous a ainsi conseillé de les prendre en compte dans notre algorithme afin d'essayer de trouver une optimisation en ajoutant que, dans sa base de données, il constate rarement un enchaînement de plus de deux trous récupérés d'affilée.

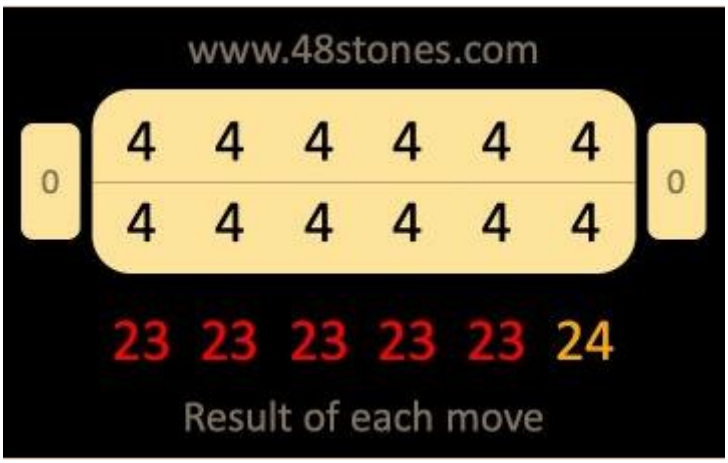


Figure 2, source : 48stones

Enfin, Xavier Blanvillain nous a également dirigé vers d'autres pistes de réflexion en nous fournissant les liens suivants :

- *Minimax: https://www.myriad-online.com/en/products/awale.htm*
- *IA: http://www.joansala.com/auale/en/*
- *Avec son heuristique: http://www.joansala.com/auale/strategy/en/*
- *Résultat de parties contre IA et Minimax: https://48stones.com/research-games-benchmark/*

Joan Sala.

Suite à la consultation des liens fournis par Xavier Blanvillain, Robin a pris la décision de contacter Joan Sala par mail afin de discuter de son moteur de jeu Aualé et de son moteur d'intelligence artificielle Aalina. Voici les éléments principaux que nous avons pu tirer de cet échange :

- Tout d'abord, Joan Sala nous a redirigé vers son repository GitHub contenant son framework "Samurai" ainsi que diverses implémentations pour différents jeux ;
- Deuxièmement, il nous également redirigé vers son repository contenant son implémentation de la logique du jeu de l'Awélé utilisant par défaut le search engine Negamax, ainsi que vers son repository contenant son interface graphique Aualé, utilisant le protocole UCI (Universal Chess Interface) afin de nous permettre de tester notre moteur d'intelligence artificielle ;

- Ensuite, Joan Sala nous a dit que suite à son travail sur son moteur, il avait tiré deux conclusions :
    - Pour l'Awélé, une simple méthode statique d'évaluation heuristique semble très bien fonctionner ;
    - De plus, toujours par l'Awélé, il semble qu'il vaille mieux avoir une méthode d'évaluation rapide que complexe, afin de pouvoir augmenter la profondeur de recherche et ainsi augmenter le nombre de coups prévus ;
- Il nous a également conseillé d'implémenter et d'améliorer au maximum un élagage alpha-beta afin de pouvoir agrandir notre profondeur au maximum et ainsi prédire le plus de coups possibles ;
- Enfin, il nous a proposé de modifier le résultat de la fonction getDecision afin de retourner uniquement le meilleur coup trouvé au lieu d'un tableau de réel, dans l'objectif d'améliorer la vitesse de notre moteur. Bien sûr, cette proposition ne respectant pas une des contraintes du sujet, nous l'avons immédiatement mise de côté.

Nos conclusions.

Une fois tous ces précieux conseils reçus, nous les avons analysés, testés pour certains, et voici les conclusions que nous avons tiré :

- Premièrement, l'obligation de commencer par un coup à droite étant simple et visiblement inévitable à implémenter afin de ne pas perdre la partie en cas de partie parfaite, nous avons décidé de l'implémenter au plus vite ;
- Deuxièmement, nous avons décidé de réfléchir à une implémentation viable d'une optimisation de la gestion des "Move in hands" et des "doubles cases dangereuses", avec notamment l'objectif de réussir à les prendre en compte dans notre heuristique. Après plusieurs essais infructueux, nous avons constaté que cela baissait la profondeur de notre algorithme et qu'il serait plus judicieux de les intégrer dans un apprentissage via un algorithme génétique, à la condition d'en avoir le temps ;
- Enfin, troisièmement, après analyse des repository envoyés par Joan Sala nous avons conclu qu'il était trop complexe et donc coûteux en temps pour nous de nous servir de son framework ou de son interface graphique. En revanche, nous avons décidé de suivre ses conseils en visant une heuristique simple et rapide afin de privilégier la profondeur ;

Implémentation.

Avant-propos.

Pour le choix de notre implémentation, la forme fut pour nous évidente suite à une discussion avec nos professeurs qui nous ont confirmé que, sur les précédentes années, les résultats les plus satisfaisants furent obtenus via un algorithme MinMax. De plus, nos discussions avec Joan Sala nous ont conforté dans cette voie, puisqu'il nous a confié que selon lui aussi une implémentation simple de MinMax alpha-bêta était probablement la meilleure piste, à la condition d'avoir une fonction d'évaluation rapide et un élagage suffisamment efficace.

Première version : MinMax alpha-bêta avec l'heuristique fournie.

Nous avons ainsi débuté ce projet en recréant un MinMax avec élagage alpha-bêta différent de celui fourni avec le projet, puisqu'il n'utilise qu'une seule classe, et calculant la profondeur utilisée dynamiquement en fonction du nombre de graines selon le calcul suivant :

12 - (NbGrainesRestantesSurLePlateau / 10)

Nous l'avons ensuite fait affronter l'implémentation MinMax fournie dans le projet, afin de nous situer dans un premier temps par rapport à un moteur dont nous supposions qu'il serait amélioré par la suite par chacun de nos camarades, et que nous avons donc supposé plus "faible" que nos véritables futurs adversaires.

```
Scores finaux :
IAWELE - minmax : 2.91
MinMax : 0.09

Rangs :
1. IAWELE - minmax : 2.91
2. MinMax : 0.09
```

Comme on peut le constater, le résultat fut sans appel : notre MinMax surpassait celui fourni. Bien que satisfaits de ces premiers résultats, il nous fallait alors continuer à l'améliorer, afin d'en avoir d'autres encore plus satisfaisants.

Deuxième version : MinMax alpha-bêta avec heuristique des Krous.

Le krou est une stratégie définie comme étant l'accumulation dans un trou d'un nombre de graines suffisant pour faire un tour complet, soit au moins 12 graines. Le Krou peut faire des ravages lorsque le territoire de l'adversaire est presque vide, car, lors du semis, le premier passage ensemence, et le deuxième se termine par une récolte *(cf. "Stratégies")*.
Nous avons donc décidé de tenter une amélioration de notre algorithme MinMax, en gardant notre calcul de profondeur mais changeant l'heuristique pour la suivante :

(0.81 * scoreDifférence) + ((0.42 * krouPossible) - (0.11 * trouDangereux))

Ces trois variables, scoreDifférence, krouPossible et trouDangereux, sont toutes trois instanciées à l'aide de méthodes codées dans la classe. Cette heuristique permet de ne pas uniquement prioriser les krous en prenant en compte le score actuel de la partie ainsi qu'en ajoutant une notion de danger : un trou est considéré comme dangereux lorsqu'il n'est rempli qu'avec 1 ou 2 graines.

Nous avons ensuite fait s'affronter cette nouvelle implémentation contre l'implémentation de MinMax fournie dans le projet, afin de nous faire une première idée :

```
Winner : 1 IAWele - Heuristique des Krou avec :56.5
Winner : 1 IAWele - Heuristique des Krou avec :91.5
Score : 0.03 - 2.94
IAWele - Heuristique des Krou a gagné
Nombre de coups joués : 72.745 par match
Durée : 00:00:02.726 par match
Mémoire utilisée : 2,8 MiB
```

Puis, dans un second temps, nous l'avons fait affronter notre première implémentation :

```
Score : 2.19 - 0.75
IAWele - Heuristique des Krou a gagné
Nombre de coups joués : 93.695 par match
Durée : 00:00:03.297 par match
Mémoire utilisée : 2,9 MiB
```

L'amélioration semblait efficace, les matchs semblaient plus complexes car nécessitant plus de coups et les consommations de temps et de mémoire semblaient tout à fait convenables.

Troisième version : amélioration du MinMax alpha-bêta avec heuristique des Krous.

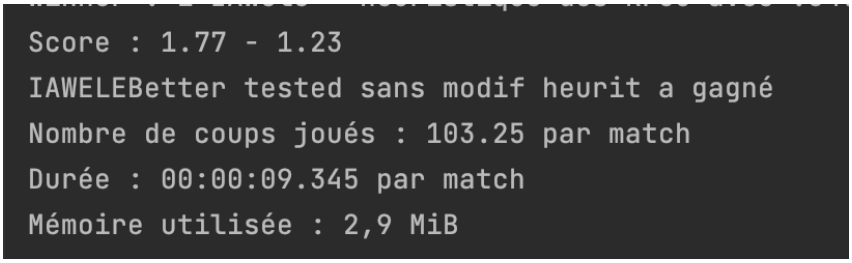
L'étape suivante consistait en une amélioration de la profondeur utilisée, afin de rendre notre algorithme encore plus performant puisque notre premier calcul nous limitait à une profondeur strictement inférieure à 12 en théorie, et 11 en pratique. Cette amélioration s'est décomposée en 2 étapes :

- **Recherche de la profondeur optimale selon les capacités de la machine :** lors de l'appel à la fonction learn, le moteur va instancier un bot avec une base de profondeur égale à 3. Il va ensuite effectuer des tests de durée de prise de décision de la même manière que dans le *main*, c'est-à-dire en instanciant un bot Random pour jouer contre lui et faire la moyenne des durées de prises de décisions, et ce jusqu'à ce que la durée dépasse la valeur souhaitée. C'est à ce moment que le moteur récupère la dernière profondeur valide, calculée avec une marge égale à 0.6*200, sachant qu'une marge max égale à 200ms est définie auparavant ;
- **Augmentation de la profondeur lors de l'exécution :** parallèlement, nous avons mis en place une condition permettant d'aller un tout petit plus loin que ce qui est prévu initialement en agrandissant la profondeur si possible. Par exemple, si la profondeur est de 0, qu'il y a moins de 20 graines sur le plateau et qu'il y a plus de 3 coup invalides alors on remet la profondeur à 1, nous permettant ainsi d'évaluer les prochaines situations très rapidement ;

C'est également à ce moment que nous avons pu échanger avec Xavier Blanvillain, nous permettant de profiter de cette nouvelle version de notre moteur pour implémenter la règle du **premier coup à droite**. Pour rappel, Xavier a prouvé que, pour obtenir une égalité ou une victoire en cas de partie parfaite, il faut impérativement commencer la partie en jouant le coup le plus à droite, ce qui permet de donner ces 4 graines à l'adversaire. Nous avons ainsi ajouté cette stratégie à toutes les versions de notre moteur en utilisant un tableau avec des poids codés en dur dans le cas où il s'agit du premier

tour de notre moteur, et ce peu importe que ce soit lui ou l'adversaire qui ait commencé la partie. Nous avons également mis à jour le calcul de la profondeur sur toutes les versions de notre moteur. C'est à ce moment que nous avons décidé que nous n'aurions pas de méthode d'apprentissage, et que nous n'utiliserions la méthode learn que pour calculer la profondeur maximale de notre MinMax.

Nous avons ensuite fait s'affronter cette nouvelle version contre la précédente, et voici les résultats :



L'amélioration est moins impressionnante que la précédente en termes d'écart de victoire, mais elle reste non-négligeable et nous avons donc conclu que nous étions, une fois de plus, sur la bonne voie.

Version finale : MinMax alpha-bêta avec amélioration de l'heuristique et profondeur améliorée.

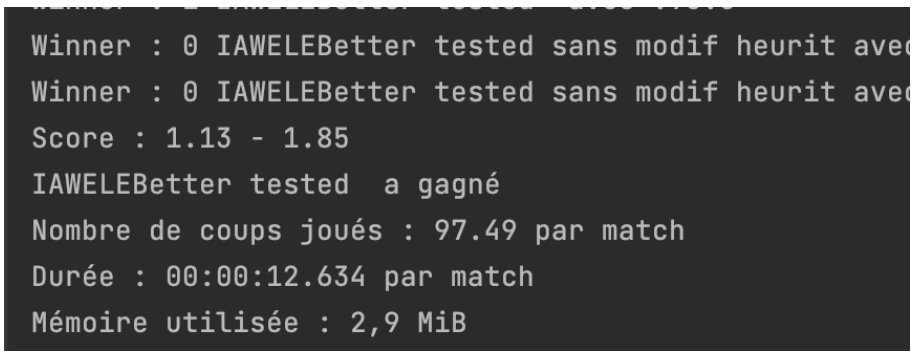
La dernière étape après avoir amélioré la profondeur de notre algorithme était de rendre notre heuristique plus efficace. C'est à ce moment, par chance, que nous avons réussi à entrer en contact avec Joan.

Lors de nos échanges avec lui, il nous a fait part de son avis expliqué précédemment : plus l'heuristique est simple, mieux c'est afin de continuer à rendre notre moteur plus rapide et efficace. Ainsi, nous avons décidé reprendre l'heuristique que Joan a utilisé pour son moteur Aalina et qui est disponible librement sur son GitHub. Son heuristique est la suivante :

Pour le joueur : Multiplier la différence de score par 30, ajouter 28 au score par krou possible, retirer 36 si un trou est inutile et retirer 54 si c'est un trou dangereux ;

Pour l'adversaire : retirer 28 par krou possible pour lui, ajouter 36 pour un trou inutile et ajouter 54 pour un trou dangereux.

Suite à cette amélioration, nous avons une fois de plus fait s'affronter notre dernière version avec la précédente, et voici les résultats :



Comme nous pouvons le voir, cette dernière version avec l'heuristique de Joan bat une fois de plus notre précédente version.

Conclusion.

Arrivés à notre quatrième version, la date de rendu était malheureusement proche et nous n'avons pas pu continuer à améliorer notre moteur. Voici, au dernier jour, les résultats de nos essais lorsque nous avons fait s'affronter nos différents moteurs ainsi que le MinMax fourni dans le projet :

```
Rangs :
1. IAWELEBetter tested : 6.98
2. IAWele - Heuristique des Krou : 6.22
3. IAWELE - minmax : 4.25
4. MinMax : 0.45
```

Comme nous pouvons le constater, notre dernière version semble bien meilleure que toutes nos autres versions, et nous pouvons dire que nous sommes très satisfaits de l'évolution des performances de notre moteur. Nous n'avons donc pas de méthode d'apprentissage, puisque nous n'utilisons la méthode learn que pour calculer la profondeur maximale.

Cependant, nous aurions aimé si nous en avions eu le temps tenter d'implémenter pour l'apprentissage un algorithme génétique pour la phase d'apprentissage afin d'améliorer encore son efficacité.

Nous pouvons également nous interroger sur la pertinence des heuristiques dans ce contexte et de leur efficacité l'une par rapport à l'autre, puisqu'il semble presque impossible de réussir à prévoir et gérer parfaitement chaque cas possible à l'aide d'une seule heuristique. Ainsi, si certains prétendent avoir résolu l'Awélé d'un point de vue théorique, nous pouvons affirmer qu'à titre personnel nous sommes loin d'avoir réussi à produire un moteur capable de le résoudre en ne commettant aucune erreur.

En mot de la fin, nous aimerions remercier particulièrement messieurs Xavier Blanvillain et Joan Sala pour leur soutien et leur intérêt dans notre travail, ainsi que notre professeur monsieur Alexandre Blansche pour l'originalité de ce sujet, qui aura su nous passionner et nous stimuler jusqu'à la dernière minute et même au-delà, puisque nous avons prévu de continuer à améliorer notre moteur afin de le rendre le plus performant possible et, qui sait, peut-être finalement réussir à arracher une égalité à 48 stones.

Références.

"**Awélé sur Wikipédia**." *Wikipédia*, https://fr.wikipedia.org/wiki/Awal%C3%A9.

Blanvillain, Xavier. "**48stones.**" *48stones*, https://48stones.com/home-fr/.

"**Play Awélé on PlayOK.**" *PlayOK*, https://www.playok.com/en/oware/.

Sala, Joan. "**Aualé.**" *Aualé*, http://www.joansala.com/auale/en/.

Sala, Joan. "**Joan Sala on GitHub.**" *GitHub*, https://github.com/joansalasoler/.

"**Stratégies.**" *Myriad Online*, http://www.myriad-online.com/resources/docs/awale/francais/strategy.htm.