INF8175 – Intelligence artificielle: méthodes et algorithmes

Hiver 2024

Rapport du projet Abalone

Equipe MR

2075512 – Mark Ibrahim

2061606 – Ryan Lahbabi

Remise : Le 18 Avril 2024

Table des matières

1 - Introduction

Abalone est un jeu de stratégie où deux joueurs s'affrontent en se relayant pour pousser les billes de l'adversaire hors du plateau. Plusieurs études ont essayé de créer un programme informatique capable de maîtriser ce jeu. Dans notre cours sur les méthodes et algorithmes en intelligence artificielle, nous avons eu la chance de développer un tel programme et d'explorer différentes approches dans le but de créer un agent intelligent capable de gagner un maximum de parties contre divers adversaires. Le but étant d'explorer les différentes notions d'algorithmes que nous avons acquises pendant le cours. C'est une occasion pour nous de les appliquer de façon ludique dans le cadre du jeu Abalone avec des heuristiques propres à ce jeu.

2 - Méthodologie

Pour développer notre agent, il nous a été fourni une version du jeu en Python. Nous pouvions utiliser toutes les bibliothèques et algorithmes disponibles. Pour mener à bien notre projet, nous avons suivi une méthodologie en deux phases faite en parallèle durant la session : Recherche adversarielle et la revue bibliographique pour améliorer notre algorithme.

Phase 1: Recherche Adversarielle:

Pour la recherche adversarielle, nous avons utilisé la stratégie vue en cours de MiniMax de l'algorithme alpha-beta prunning . Pour cela, nous avons conçu deux méthodes intitulé `mini_max_alpha_beta_max_value et mini_max_alpha_beta_min value` Dans le contexte du jeu Abalone, ils permettent de prendre des décisions plus stratégiques et plus rapides en se concentrant uniquement sur les branches de l'arbre de décision qui sont pertinentes pour obtenir le meilleur résultat possible. Chaque mouvement potentiel est évalué non seulement sur la base de ses conséquences immédiates mais aussi de ses implications futures. En éliminant les branches inutiles, le joueur peut prendre des décisions dans les temps limités alloués, ce qui est crucial surtout dans les versions compétitives du jeu où le temps est un facteur limitant. L'approche adaptative face au temps restant, choisissant parfois un mouvement aléatoire si le temps est presque écoulé, montre une stratégie de repli utile dans des situations de haute pression.

Les deux méthodes se complètent. La première est utilisée par le joueur qui essaie de maximiser son score. Cela signifie qu'elle cherche à maximiser les avantages obtenus par les mouvements possibles. Elle évalue le score optimal qu'un joueur peut assurer contre la meilleure défense de l'adversaire. Ensuite, à l'inverse, `mini_max_alpha_beta_min` est employée par le joueur qui cherche à minimiser le

score maximum que l'adversaire peut atteindre. C'est le point de vue de l'adversaire qui tente de réduire les pertes, simulant le meilleur coup possible pour l'adversaire dans le cadre d'une stratégie défensive. Ces valeurs sont mises à jour à travers les appels récursifs et utilisées pour décider si une branche doit être explorée ou non. Si une branche ne peut pas améliorer le résultat pour le joueur actuel par rapport à la meilleure option déjà trouvée par l'adversaire, elle est coupée.

Afin de concevoir l'heuristique propre au jeu Abalone, il nous a fallu étudier le jeu et se renseigner sur les différentes stratégies pour gagner une partie. La première heuristique, qui est la plus triviale, est de faire en sorte que l'agent possède plus de pièces que l'adversaire et de l'encourager à capturer les pièces adverses.

La deuxième heuristique est de programmer l'agent afin qu'il essaie d'avoir le plus de pièces au centre du jeu car c'est à cet endroit que les pièces seront les plus éloignées du bord du plateau et qu'elles risquent moins de se faire éjecter. Pour implémenter cela, nous avons calculé les distances euclidiennes entre chaque pièce et les positions voulues au centre qui sont de coordonnées (4, 4).

Ensuite, notre troisième heuristique est la différence de score. Elle permet d'évaluer la différence de score entre le joueur actuel et son adversaire, et ce, de manière à influencer le choix de l'action à effectuer. Si un joueur est significativement en avance ou en retard, cela devrait influencer ses décisions stratégiques, incitant à un style de jeu plus défensif ou agressif selon la situation.

La quatrième et dernière heuristique est d'avoir des pièces adjacentes afin qu'il soit plus difficile pour l'adversaire de bouger nos pièces par supériorité numérique et plus facile pour nous d'attaquer les pièces de l'adversaire, cet heuristique améliore alors la performance offensive et défensive à la fois de notre agent. C'est la méthode `adjacent_positions` qui nous permet de calculer cette heuristique

Avec toutes ces heuristiques, nous avons alors calculé un score avec un coefficient plus élevé pour la différence au score qui a avantagé notre agent plus que les autres heuristiques pendant le déroulement des parties. Les coefficients ont été calculés par de simples tests essaie-erreur dont je vous parlerais davantage dans la partie résultats et évolution de l'agent.

<u>Phase 2: Recherche bibliographique:</u>
Pendant la réalisation de notre projet, nous avons pensé que si nous voulions aller au-delà des limites de l'algorithme alpha Beta initialement implémenté, il serait intéressant d'étudier davantage les ressources littéraires en ligne.

En regardant les travaux de Toumpas et al. (2012) sur l'optimisation des stratégies dans le cadre de la recherche Alpha-Beta pour Abalone, nous avons compris qu'une amélioration de l'algorithme Alpha-Bêta nécessite l'intégration de techniques d'estimation performantes. Leur étude montre également que notre hypothèse initiale selon laquelle posséder le plus de pièces au centre du plateau est stratégique et souligne l'importance de regrouper les pièces pour planifier des attaques tout en minimisant les risques de contre-attaques adverses. En conséquence, nous avons intégré un critère de regroupement à notre heuristique, favorisant la formation de grands groupes. Cette approche implique une recherche en largeur (breadth-first search) pour chaque groupe de pièces afin de déterminer leur nombre. Notre fonction d'évaluation prend donc en compte ces éléments pour évaluer l'avantage sur l'adversaire dans une situation donnée.

Comme discuté par Miikkulainen (2007), l'utilisation de techniques d'apprentissage automatique comme les réseaux de neurones et l'apprentissage par renforcement peut être adaptée pour améliorer les capacités stratégiques de notre agent. Ces méthodes peuvent aider à développer un agent qui apprend de ses erreurs et s'adapte aux stratégies de l'adversaire au fil du temps. Malheureusement nous n'avions pas finalement eu le temps de tester cette approche mais nous les garderons en tête dans le cadre du prochain d'agent intelligent que nous programmerons.

3 - Résultats et évolution de l'agent

Afin de tester notre agent, nous l'avons fait affronter les différents agents fournis dans le devoir. Nous avons commencé par le plus facile à battre à savoir le *random_player_abalone* ensuite le *greedy_player_abalone*.

Pendant le développement de notre agent, avant de penser à ces quatre heuristiques, nous avions tout d'abord commencé avec deux heuristiques seulement qui étaient la préférence d'avoir le plus de pièces et d'être le plus proche du centre du plateau. Dès le début notre agent battait systématiquement le random_player et nous avons pensé qu'il n'était plus pertinent de tester notre agent contre ce dernier. En passant au niveau suivant, nous avons testé dix fois notre agent mais cette fois-ci contre le greedy_player.py. Les résultats étaient moins saitisfaisant, d'où notre besoin de penser à d'autres heuristiques. Vous trouverez ci-dessous les résultats de nos tests contre le greedy_player.py par nombre d'heuristiques, avec trois heuristiques c'était la position adjacente que nous avions ajouté et enfin pour le quatrième, nous avons implémenté la différence de score. A noter que nous considérions qu'un match nul comme une

défaite, afin d'être exigeant et d'avoir un algorithme encore plus robuste contre ces adversaires.

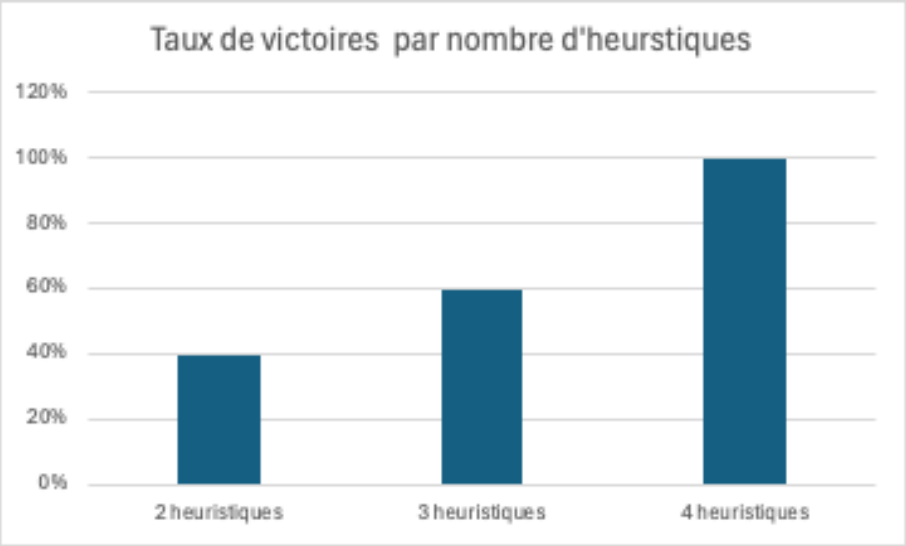

Figure 1 : Taux de victoire en fonction du nombre d'heuristiques

Après avoir initié notre agent, nous avons décidé de le tester en le confrontant aux agents développés par d'autres équipes du cours, pour évaluer ses performances. Les résultats initiaux ont été variés. Lors des quatre premières parties contre l'agent du premier binôme, notre agent a remporté deux victoires et concédé deux matchs nuls. En revanche, face à un second binôme, il a perdu trois des quatre parties, ce qui a soulevé des questions sur son efficacité.

Suite à ces résultats mitigés, nous avons entrepris une révision détaillée des poids attribués aux différentes heuristiques de notre modèle. Après avoir ajusté les paramètres, notamment en augmentant à trois le coefficient pondérant la différence de score, nous avons observé une nette amélioration. En effet, lors d'une nouvelle série de quatre parties contre l'agent du deuxième binôme, notre agent a réussi à remporter trois victoires. Ces ajustements semblent donc avoir eu un impact significatif sur la compétitivité de notre agent et justifient les coefficient qu'on a attribué par essai-erreurs.

4 - Discussion

Notre solution offre plusieurs avantages qui expliquent ses performances satisfaisantes lors des tests. Premièrement, elle inclut deux éléments clés favorisant la capture des pièces adverses : une heuristique qui attribue un score

supérieur lorsque notre agent possède plus de pièces que l'adversaire. Deuxièmement, notre stratégie encourage les pièces à se positionner vers le centre du plateau, une tactique reconnue pour améliorer la défense contre les attaques, tout en facilitant l'avancée vers l'adversaire. En outre, notre heuristique favorise les formations offensives en incitant l'agent à adopter des configurations avec plusieurs pièces adjacentes, ce qui renforce le contrôle du plateau et la défense contre les attaques adverses. La combinaison de ces éléments offre une approche équilibrée de la prise de décision, en considérant différents aspects de l'état du jeu pour évaluer la qualité des états intermédiaires pendant le déroulement de la partie.

Cependant, notre agent présente des faiblesses notables. L'une des principales est l'utilisation d'un heuristique statique qui ne s'adapte pas aux changements dynamiques du jeu ni à l'évolution de l'état du plateau. Nous avons envisagé d'améliorer cet aspect avec notre deuxième agent, mais le manque de temps a empêché son implémentation. De plus, notre système ne bénéficie d'aucun mécanisme d'apprentissage tiré des expériences précédentes ou des résultats des parties. Une implémentation de type apprentissage profond aurait pû améliorer les performances de notre agent. Une autre limitation majeure réside dans la sensibilité de notre agent aux poids attribués aux différentes composantes de l'heuristique, choisis initialement par tâtonnement, ce qui pose problème. Enfin, notre heuristique ne prend pas en compte la stratégie de l'adversaire.

5 - Conclusion

Tout au long de ce projet, nous avons eu l'occasion de valoriser nos compétences en intelligence artificielle, en améliorant constamment notre agent. Nous reconnaissons ses limites et il serait judicieux de nous pencher sur l'application de techniques telles que les heuristiques adaptatives, l'apprentissage par renforcement et l’apprentissage automatique afin de l'améliorer davantage. Au-delà des performances de notre agent, ce projet a enrichi notre compréhension des concepts l’intelligence artificielle et pose les fondations pour élaborer un modèle encore plus performant à l'avenir que nous pourrons appliquer dans un jeu de société ou dans d’autres domaines encore plus communs du monde professionnel tel que l’éducation, les finances, les services, la santé etc.

6 - Références

1. Miikkulainen, R. (2007). Creating intelligent agents in games. In National Academy of Engineering (Ed.), Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2006 Symposium (pp. 16-23). National Academies Press. Doi: 10.17226/1182

2. Konstantinos Toumpas, Athanasios Papadopoulos et al. 'Exploring optimization strategies in board game Abalone for Alpha-Beta search'. In: *2012 IEEE Conference on Computational Intelligence and Games, CIG 2012*. 2012. DOI: 10.1109/CIG.2012.6374139