INF8175 - Intelligence artificielle : Méthodes et algorithmes

Automne 2024

Projet Final

Équipe 91

Huy Viet Nguyen, 2136374

Alaa Eddine Chenak, 1976567

Soumis à : M. Quentin Cappart

7 décembre 2024

Introduction

Le jeu de plateau *Divercité* se distingue par sa complexité stratégique, où deux joueurs s'affrontent pour maximiser leurs scores en plaçant des cités et des ressources sur un plateau. Chaque décision prise par les joueurs influence non seulement leur propre score, mais également celui de leur adversaire, rendant le jeu à somme nulle particulièrement compétitif. Ce projet, réalisé dans le cadre du cours INF8175, a pour objectif de concevoir et de développer un agent intelligent capable de rivaliser efficacement dans cet environnement complexe et d'optimiser ses décisions selon les règles et contraintes du jeu.

Pour atteindre cet objectif, nous avons exploré les techniques classiques de recherche adversariale, en nous concentrant sur l'algorithme Minimax, largement reconnu pour son efficacité dans les jeux de stratégie. Cet algorithme a été renforcé par l'élagage Alpha-Bêta, permettant de réduire les calculs redondants en éliminant les branches non pertinentes de l'arbre de recherche. De plus, un mécanisme d'ajustement dynamique de la profondeur a été intégré pour adapter l'agent à différentes phases du jeu, en tenant compte de la complexité croissante des décisions et du temps computationnel disponible.

1. Méthodologie

La méthodologie adoptée pour développer notre agent intelligent pour le jeu **Divercité** repose sur une combinaison de techniques de recherche adversariale, d'évaluations heuristiques, et d'optimisations algorithmiques. L'objectif principal était de concevoir un agent capable de prendre des décisions stratégiques optimales dans les limites du temps computationnel imparti, tout en exploitant au mieux les différentes phases du jeu.

1.1 Architecture de l'agent

L'agent repose sur l'algorithme **Minimax**, qui explore l'arbre des états de jeu en simulant les coups possibles des deux joueurs pour identifier la meilleure action à entreprendre face à un agent rationnel. Cette approche est renforcée par l'**élagage Alpha-Bêta**, qui optimise la recherche en réduisant le nombre de nœuds à explorer. L'élagage permet d'exclure efficacement les branches qui ne peuvent pas influencer la décision finale, diminuant ainsi la complexité computationnelle sans compromettre la qualité des choix.

1.2 Mécanisme d'ajustement dynamique de la profondeur

Pour équilibrer efficacité et rapidité, un mécanisme d'ajustement dynamique de la profondeur a été mis en place, divisant le jeu en trois phases clés :

- **Début de partie** : Une profondeur de recherche limitée (3 niveaux) est utilisée pour accélérer les décisions lorsque peu de pièces sont sur le plateau.
- **Milieu de partie** : Une profondeur moyenne (4 niveaux) est adoptée pour maintenir un équilibre entre la qualité des décisions et la rapidité des calculs.
- **Fin de partie** : Une profondeur maximale (5 niveaux) est atteinte pour explorer en détail les configurations critiques, où chaque décision peut avoir un impact déterminant sur l'issue du jeu.

Le choix de la profondeur est ajusté dynamiquement en fonction du temps restant et du nombre d'actions possibles. Cette stratégie garantit une adaptation optimale aux exigences spécifiques de chaque phase du jeu.

1.3. Fonction d'évaluation des états

Une **fonction d'évaluation** sophistiquée guide les décisions de l'agent en évaluant les états du plateau selon deux composantes principales :

- **Différentiel de score** : Cette composante mesure l'écart entre le score de l'agent et celui de son adversaire, en intégrant les changements récents pour anticiper à la fois les opportunités et les menaces.

- **Force positionnelle** : Une heuristique évalue l'avantage stratégique des pièces, basé sur deux critères spécifiques :
    - **Centralité** : Les pièces situées au centre du plateau reçoivent un bonus, en raison de leur flexibilité et de leur potentiel de contrôle.
    - **Interactions avec les voisins** : Les pièces adjacentes, qu'elles soient alliées ou adverses, influencent positivement ou négativement l'évaluation selon leur type et leur propriétaire.

Cette **fonction** permet à l'agent de prioriser les actions maximisant son score tout en minimisant les opportunités de son adversaire.

1.4 Recherche de quiescence dans la phase finale

Pour éviter les erreurs dues à l'effet de l'horizon, la **recherche de quiescence** est intégrée dans la phase finale du jeu. Elle affine l'évaluation des positions instables, où des événements critiques (captures, menaces immédiates) peuvent fausser les décisions.

- **Principe** : Lorsque l'évaluation standard d'un état du plateau détecte une instabilité (par exemple, des captures possibles), l'exploration est prolongée pour inclure uniquement les actions "bruyantes" (captures ou mouvements décisifs).
- **Critères d'activation** : Cette recherche est déclenchée dans des situations où des changements significatifs du différentiel de score ou des positions stratégiques sont imminents.
- **Avantages** : Elle améliore la précision des évaluations, réduit les erreurs d'horizon, et garantit des décisions robustes dans les configurations critiques.

La recherche de quiescence est directement intégrée au mécanisme d'ajustement dynamique de la profondeur, intervenant lorsque l'algorithme atteint la profondeur maximale.

1.5 Optimisation par hachage Zobrist

Pour éviter les calculs redondants, une **table de transpositions** basée sur le hachage Zobrist a été implémentée. Chaque état du jeu est associé à une clé unique, permettant de stocker et réutiliser les évaluations des états déjà explorés. Cette optimisation réduit considérablement les besoins computationnels, notamment dans les scénarios où différentes séquences d'actions mènent à des états identiques.

2. Résultats et évaluation

La phase d'évaluation de notre projet se décompose en deux volets distincts : la **précision**, mesurée en termes de taux de victoire et de qualité des décisions stratégiques, et la **performance**, évaluée en termes de temps d'exécution et de nombre de nœuds explorés. Les résultats obtenus permettent de comparer deux algorithmes (Algorithme 1 et Algorithme 2) développés pour le jeu *Divercité*. Ces algorithmes reposent sur des approches similaires (Minimax avec élagage Alpha-Bêta), mais diffèrent dans leur mécanisme de gestion de profondeur et leur utilisation des tables de transpositions.

Les expérimentations ont été menées en simulant plusieurs séries de parties entre :

- **Algorithme 1** : Un Minimax classique avec élagage Alpha-Bêta et une profondeur fixe de 4 et une table de transpositions.
- **Algorithme 2** : Une version améliorée avec profondeur dynamique, table de transpositions, et heuristique adaptative.

2.1 Précision

Pour évaluer la précision, nous avons mesuré le taux de victoire des deux algorithmes dans un environnement compétitif. Chaque algorithme a été confronté à lui-même et à l'autre dans des simulations comprenant 10 parties

pour chaque scénario. L'objectif était d'évaluer leur capacité à maximiser les scores tout en minimisant les opportunités adverses.

Résultats des confrontations

	Algorithme 1	Algorithme 2
Algorithme 1	Égalité	Algorithme 2 Vainqueur
Algorithme 2	Algorithme 2 Vainqueur	Égalité

Table 1.Comparaison des performances des algorithmes en termes de taux de victoire

Analyse :

- Lorsque confrontés à eux-mêmes, les deux algorithmes affichent une égalité, démontrant la cohérence et la fiabilité de leur implémentation respective.
- L'Algorithme 2 s'impose systématiquement face à l'Algorithme 1, grâce à sa profondeur dynamique et à son exploitation des tables de transpositions, qui permettent une meilleure anticipation des scénarios adverses.

2.2 Performance

La performance de notre agent a été évaluée en termes de **temps d'exécution par tour** et de **temps moyen par coup**, afin de mettre en lumière son efficacité computationnelle et sa capacité à s'adapter aux différentes phases du jeu. Les deux algorithmes testés, à savoir **l'Algorithme 1** (Minimax classique avec profondeur fixe) et **l'Algorithme 2** (version améliorée avec profondeur dynamique et heuristique adaptative), ont été comparés pour identifier leurs forces et limitations.

Temps moyen par tour

La **Figure 1** montre une comparaison des temps moyens par tour pour chaque algorithme :

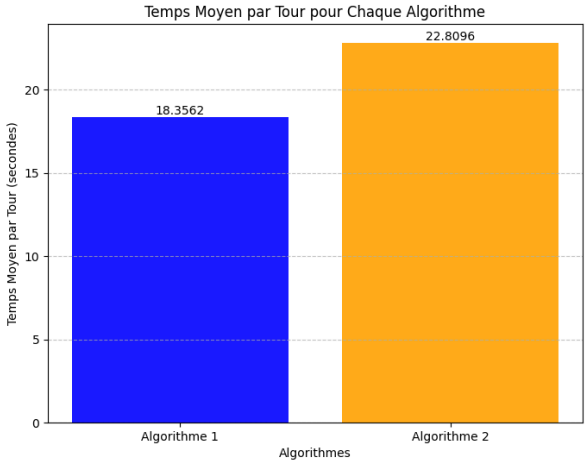

Figure 1 : Temps Moyen par Tour pour chaque Algorithme

Algorithme 1 : Il atteint un temps moyen de **18,36 secondes par tour**, démontrant une stabilité relative grâce à sa profondeur fixe de 4 niveaux et à l'optimisation par table de transpositions.

Algorithme 2 : Avec un temps moyen légèrement supérieur de **22,81 secondes par tour**, cet algorithme investit davantage de temps dans des décisions stratégiques critiques, notamment en utilisant des mécanismes comme la profondeur dynamique et la recherche de quiescence.

Ces résultats montrent que l'Algorithme 2 est plus coûteux en termes de temps moyen, mais cette augmentation s'explique par sa capacité à effectuer des recherches plus approfondies dans des configurations complexes.

Analyse détaillée des temps par tour

La **Figure 2** illustre les variations du temps d'exécution par tour entre les deux algorithmes, avec des observations marquantes selon les phases du jeu :

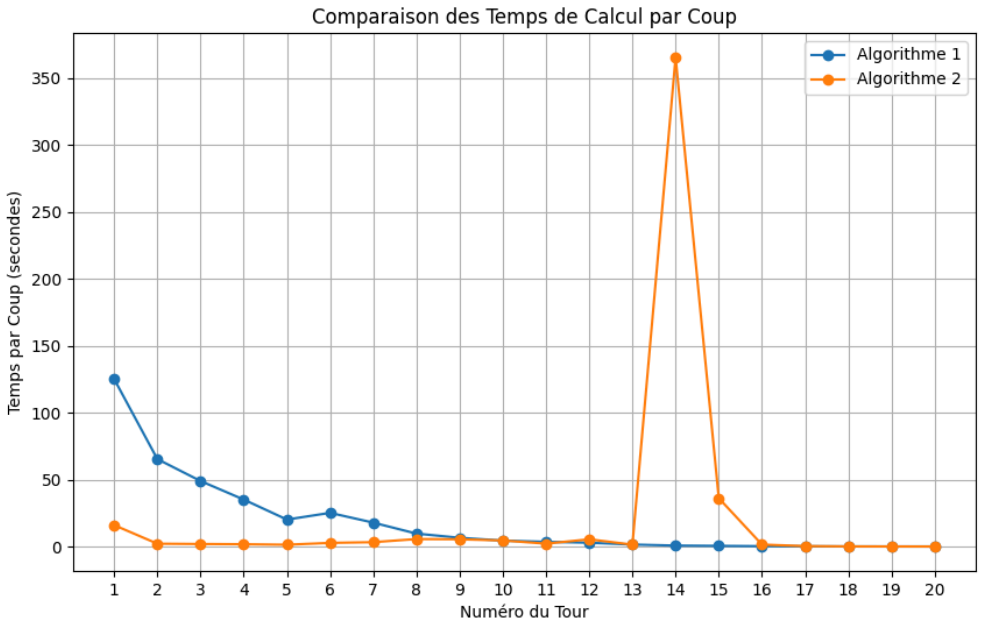

Figure 2 : Comparaison des Temps de Calcul par Coup pour chaque Tour

1. **Début de partie (tours 1 à 8)** :
   - L'Algorithme 2 se montre plus rapide grâce à sa gestion dynamique de la profondeur, qui limite les calculs dans des positions simples.
   - L'Algorithme 1 présente une décroissance progressive de son temps de calcul, atteignant une stabilisation dès le 8 tour.
2. **Milieu de partie (tours 8 à 13)** :
   - Les deux algorithmes affichent des performances similaires, bien que l'Algorithme 2 parvienne à maintenir une légère avance grâce à son adaptabilité.
   - L'utilisation de tables de transpositions dans les deux cas contribue à réduire les calculs redondants, stabilisant les temps de réponse.
3. **Fin de partie (tours 14 à 20)** :
   - Un **pic critique** est observé pour l'Algorithme 2 au **tour 14**, où son temps de calcul dépasse les 350 secondes. Ce comportement est lié à l'activation de la **recherche de quiescence**, qui prolonge l'exploration dans des positions instables afin de garantir des décisions optimales.
   - En dehors de ce pic, l'Algorithme 2 retrouve une performance supérieure, clôturant les derniers tours plus rapidement que l'Algorithme 1.

2.3 Comparaison globale

Malgré un temps moyen plus élevé, l'Algorithme 2 démontre une meilleure gestion des positions critiques grâce à des mécanismes avancés comme la profondeur dynamique et la recherche de quiescence. En particulier, l'Algorithme 2 concentre une part significative de ses ressources sur les états de la fin de jeu, où les décisions stratégiques ont un impact crucial sur l'issue de la partie. Cette focalisation permet d'explorer plus en profondeur les configurations critiques et d'anticiper les scénarios adverses. L'Algorithme 1, bien qu'efficace et stable, montre ses limites dans des configurations plus complexes, où une exploration adaptative et ciblée, comme celle réalisée par l'Algorithme 2, devient essentielle.

3. Évolution de l'agent

Le développement de notre agent a suivi une approche itérative, passant par plusieurs versions successives, chacune visant à résoudre les faiblesses identifiées dans les versions précédentes. D'abord, nous avons commencé avec un agent Minimax à profondeur 2, puis nous avons augmenté la profondeur de recherche à 4 en introduisant un élagage Alpha-Bêta. Ensuite, nous avons introduit une table de transposition basée sur le hachage de Zobrist pour accélérer la recherche et introduit une profondeur de recherche dynamique allant de 3 jusqu'à 5. Puis, nous avons ajouté une heuristique qui évalue la qualité du placement des villes pour améliorer la qualité de l'évaluation des états.

Discussion

Notre agent repose sur une recherche Minimax avec une certaine profondeur, intégrant des mécanismes d'accélération tels que l'élagage Alpha-Bêta et l'utilisation d'une table de transposition. Ces mécanismes lui permettent d'atteindre des profondeurs de recherche jusqu'à 5, tout en adoptant une évaluation des états calibrée pour le contexte spécifique du jeu. Cependant, les résultats obtenus montrent que les performances de l'agent restent limitées par des faiblesses inhérentes à l'approche Minimax, notamment en ce qui concerne la profondeur de recherche et la qualité de l'heuristique utilisée pour évaluer les états.

Lorsqu'il atteint une profondeur de recherche supérieure à 4, l'agent commence à déployer des tactiques plus sophistiquées, comme le placement stratégique de villes au centre du plateau pour maximiser son score. Cela contraste avec une stratégie plus naïve de remplissage du plateau en partant du coin supérieur droit. Toutefois, ces performances demeurent insuffisantes face à des agents utilisant des profondeurs supérieures à 5, qui sont capables de développer des stratégies à plus long terme, notamment la préparation de diversifications stratégiques. Cette limitation met en lumière la dépendance de notre agent à des ressources informatiques accrues ou à des techniques avancées comme l'intégration de la recherche en arbre Monte Carlo pour prolonger l'analyse des états prometteurs.

De plus, l'évaluation des états, bien que basée sur la position des villes, manque de complexité. Elle ne prend pas en compte des éléments clés tels que le nombre de villes restantes, les ressources disponibles ou leur emplacement sur le plateau. Ces aspects sont pourtant cruciaux dans la gestion des ressources, une composante essentielle du jeu que notre agent ignore. Pour surmonter cette limite, une approche basée sur l'apprentissage machine pourrait être envisagée, permettant de développer une fonction d'évaluation plus robuste et intégrant toutes les variables pertinentes pour le jeu.

Conclusion

En résumé, notre agent représente une amélioration par rapport aux versions précédentes, grâce à l'utilisation de mécanismes d'accélération de la recherche et d'une évaluation contextuelle des états. Toutefois, ses performances sont limitées par une profondeur de recherche bornée et une évaluation des états encore insuffisamment sophistiquée. Pour améliorer ces performances, deux pistes principales se dégagent : l'augmentation de la puissance de calcul pour permettre une recherche plus profonde et l'intégration d'une fonction d'évaluation plus avancée, éventuellement basée sur des techniques d'apprentissage machine. Ces améliorations pourraient permettre à l'agent de mieux gérer les subtilités stratégiques du jeu, en exploitant pleinement les ressources et en anticipant des tactiques à long terme.