Université Paris Cité

Rapport Final
Projet d'intelligence artificielle

Année : 2024-2025

Groupe N°3 :

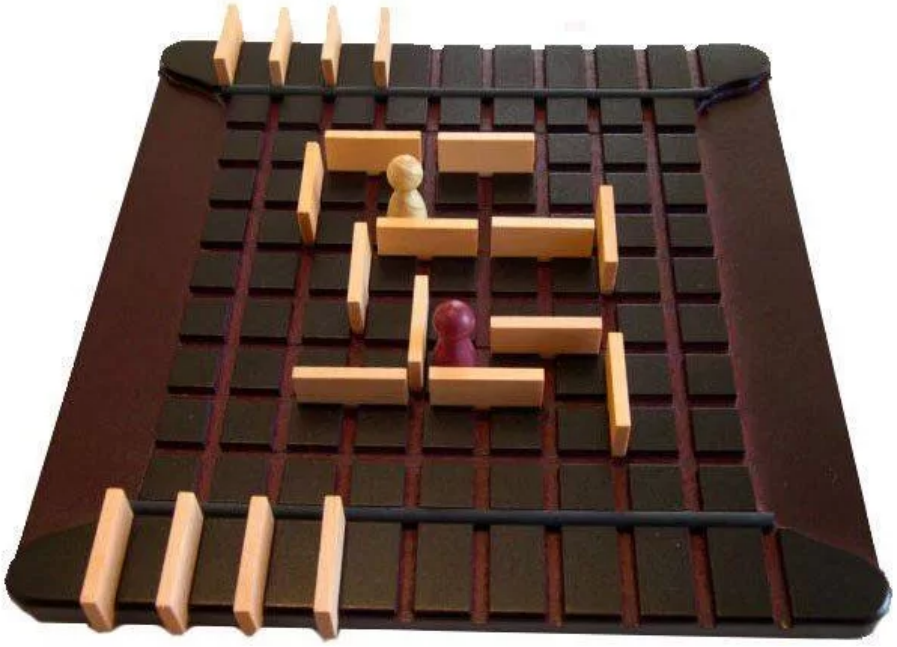
GOURMELEN Thomas, PONNOUSSAMY Valentin

Enseignante : Elise Bonzon

1) Introduction.

Dans le cadre de notre projet en intelligence artificielle, nous avons pu développer le jeu Quoridor ainsi que des intelligences artificielles pour ce dernier. Afin d'être en accord avec les consignes du projet, notre choix s'est porté sur ce jeu à 2 joueurs et à connaissance parfaite. (Ce jeu fut également un cadeau d'anniversaire très apprécié…)

Quoridor est un jeu de stratégie combinatoire abstrait à 2 ou 4 joueurs. Dans notre projet, nous nous sommes concentrés sur la version à 2 joueurs.



Voici une représentation du jeu au cours d'une partie à 2 joueurs, sur cette image, nous pouvons observer :

- Les pions des deux joueurs
- Les murs placés par les joueurs.
- Les murs en stock pour chaque joueur.

2) Description du jeu.

2.1) Règle du jeu.

2.1.1) But du jeu.

Le but du jeu est d'atteindre le premier la ligne opposée à sa ligne de départ. Le problème est que le joueur adverse peut poser des barrières pour ralentir l'avancée, et inversement. Toutefois, il est obligatoire de laisser au moins un passage libre.

2.1.2) Comment jouer.

Les joueurs se partagent les barrières, et chacun place son pion au centre de sa ligne de départ. À tour de rôle, ils déplacent leur pion d'une case ou posent une barrière afin de ralentir l'adversaire. Les pions doivent contourner les barrières, qui créent un labyrinthe dont il faut sortir.

2.1.3) Règles.

Début de la partie :

Les barrières sont remisées dans leur zone de stockage (10 par joueur). Chaque joueur place son pion au centre de sa ligne de départ (la première devant lui). Les deux joueurs se font face. Un tirage au sort détermine qui commence.
À tour de rôle, chaque joueur choisit de déplacer son pion d'une case ou de poser l'une de ses barrières.

Déplacement des pions :

Les pions se déplacent d'une seule case, horizontalement ou verticalement, vers l'avant ou vers l'arrière, mais jamais en diagonale. Les barrières doivent être contournées et ne peuvent pas être sautées.

Face à face :

Quand les deux pions se retrouvent en vis-à-vis sur deux cases voisines non séparées par une barrière, le joueur dont c'est le tour peut sauter par-dessus son adversaire. Si une barrière se trouve derrière le pion sauté, alors le pion sauteur doit être déplacé sur l'une des cases adjacentes au pion sauté.

Pose des barrières :

Les barrières doivent être posées exactement entre deux blocs de deux cases. Elles ne peuvent pas se chevaucher. Une fois posées, elles ne peuvent plus être retirées ni déplacées pendant la partie.

La pose des barrières a pour but de se créer son propre chemin ou de ralentir l'adversaire, mais il est interdit de lui bloquer totalement l'accès à sa ligne d'arrivée : une solution doit toujours lui être laissée.

Lorsqu'un joueur n'a plus de barrières, il est obligé de déplacer son pion.

Fin de la partie :

Le premier joueur qui atteint l'une des neuf cases de la ligne opposée à sa ligne de départ remporte la partie.

2.2) Programme.

2.2.1) Mode de jeux :

Plusieurs modes de jeu sont possibles :

- Humain *VS* Humain.
- Humain *VS* IA.
- IA *VS* IA.

Pour un joueur IA, il existe trois niveaux de difficulté, correspondant à des valeurs de paramétrage spécifiques :

- Niveau facile :
  - Profondeur : 1
  - Epsilon : 0,4
  - Poids avancer : 1
  - Poids bloquer : 0,5
  - Poids murs : 0,2

- Niveau moyen :
  - Profondeur : 2
  - Epsilon : 0,2
  - Poids avancer : 1,2
  - Poids bloquer : 0,8
  - Poids murs : 0,3

- Niveau difficile :
  - Profondeur : 3
  - Epsilon : 0,1
  - Poids avancer : 1,5
  - Poids bloquer : 1
  - Poids murs : 0,4

Nous avons choisi ce paramétrage à la suite de nombreux tests de performance de l'IA avec différents paramètres.

2.2.2) Déroulement d'une partie :

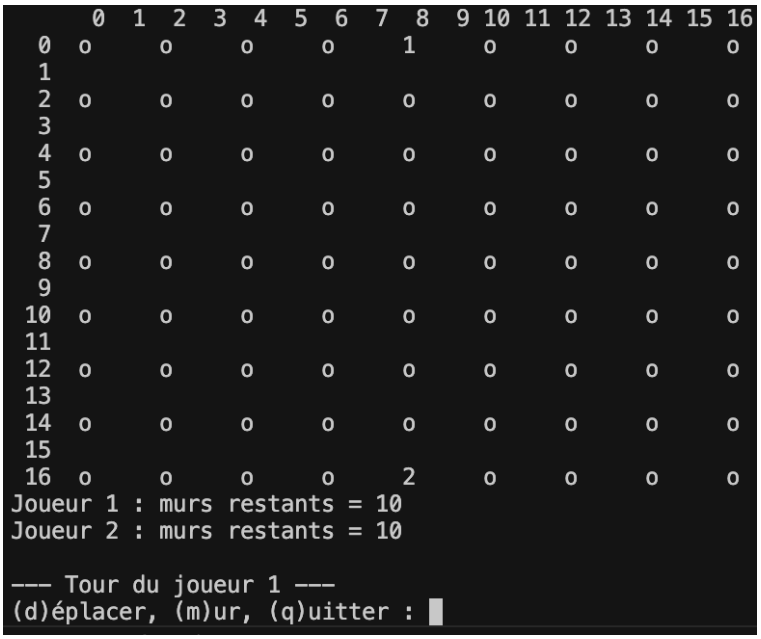
Avant le début de partie :

1. Au lancement du programme, l'utilisateur choisit son mode de jeu :

2. Si l'utilisateur choisit le mode de jeu 2 ou 3, il devra sélectionner le niveau de difficulté de l'IA ou des IA :

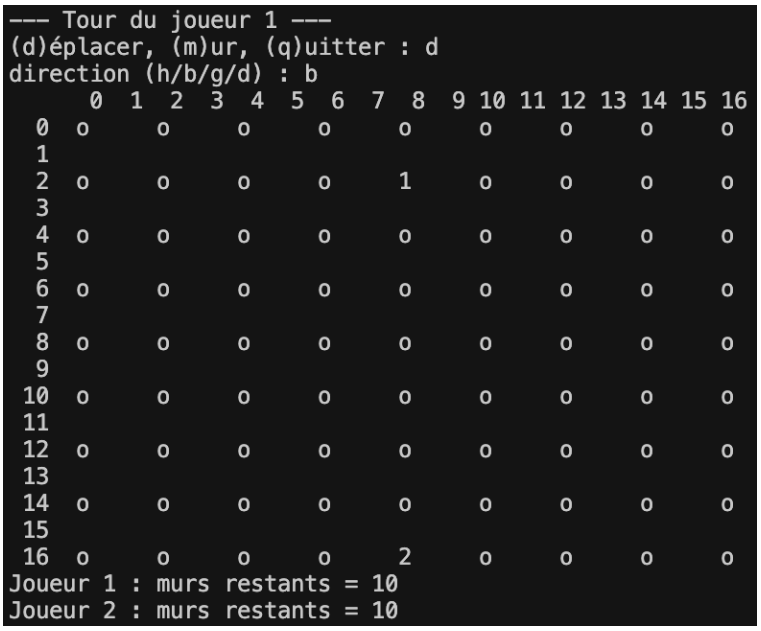
<u>Début de partie :</u> (Ici, nous prendrons pour exemple le déroulement d'une partie Humain contre IA.)

1. Affichage du plateau de jeu :



Quand la partie commence, les joueurs peuvent visualiser le plateau de jeu, la position initiale de leur pion, ainsi que le nombre de murs en leur possession.

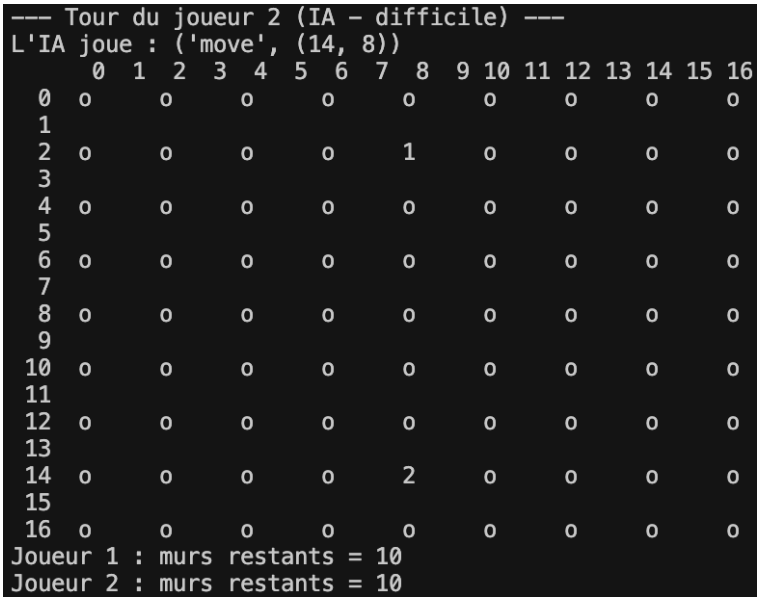
2. Action (déplacement) :



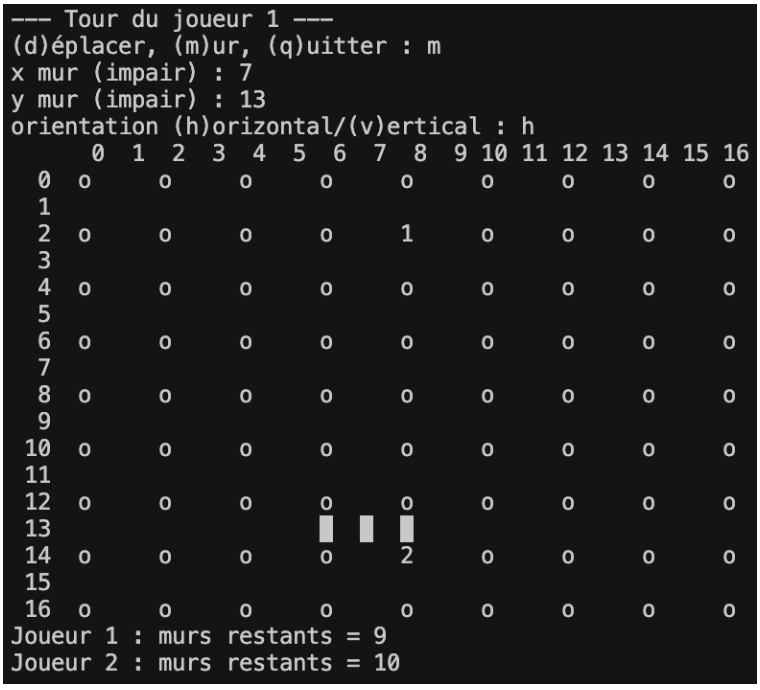
En choisissant l'action « Se déplacer », l'utilisateur doit indiquer la direction de son déplacement (haut, bas, gauche ou droite).

(Dans notre exemple, le joueur a choisi d'effectuer un déplacement vers le bas.)

Le joueur IA a joué !!! Il faut absolument bloquer sa progression pour l'empêcher de gagner.



3. Action (placement d'un mur) :



En choisissant l'action « Placement d'un mur », le joueur doit renseigner les coordonnées x et y à lesquelles il souhaite placer son mur. Il doit également préciser l'orientation du mur (horizontale ou verticale).

(Ici, le joueur a décidé de placer un mur à la position x = 7 et y = 13. Nous pouvons remarquer que son nombre de murs restants a diminué.)

4. Action (déplacement au-dessus d’un joueur) :

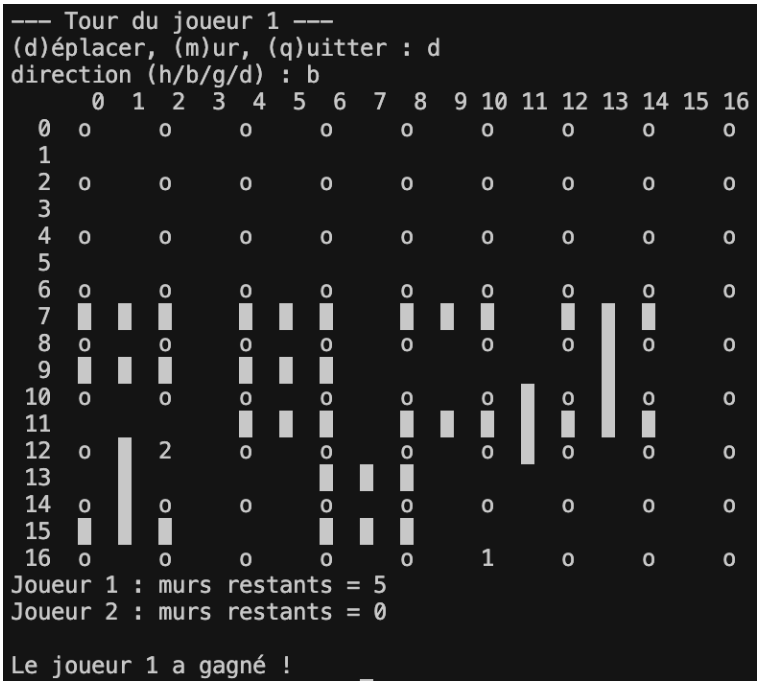
```
--- Tour du joueur 2 (IA - difficile) ---
L'IA joue : ('move', (8, 10))
      0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
  0  o     o     o     o     o     o     o     o     o
  1
  2  o     o     o     o     o     o     o     o     o
  3
  4  o     o     o     o     o     o     o     o     o
  5
  6  o     o     o     o     o     1     o     o     o
  7
  8  o     o     o     o     o     2     o     o     o
  9
 10  o     o     o     o     o     o     o     o     o
 11
 12  o     o     o     o     o     o     o     o     o
 13                  ▮  ▮  ▮
 14  o     o     o     o     o     o     o     o     o
 15
 16  o     o     o     o     o     o     o     o     o
Joueur 1 : murs restants = 9
Joueur 2 : murs restants = 10

--- Tour du joueur 1 ---
(d)éplacer, (m)ur, (q)uitter : d
direction (h/b/g/d) : b
      0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
  0  o     o     o     o     o     o     o     o     o
  1
  2  o     o     o     o     o     o     o     o     o
  3
  4  o     o     o     o     o     o     o     o     o
  5
  6  o     o     o     o     o     o     o     o     o
  7
  8  o     o     o     o     o     2     o     o     o
  9
 10  o     o     o     o     o     1     o     o     o
 11
 12  o     o     o     o     o     o     o     o     o
 13                  ▮  ▮  ▮
 14  o     o     o     o     o     o     o     o     o
 15
 16  o     o     o     o     o     o     o     o     o
Joueur 1 : murs restants = 9
Joueur 2 : murs restants = 10
```

Dans cette situation, le joueur 1 peut décider, en effectuant un déplacement vers le bas, de sauter par-dessus le joueur adverse.

Le joueur 1 (humain) a sauté au-dessus du joueur adverse, ce qui lui a fait gagner deux cases.

Fin de partie :



Victoire le joueur 1 a gagné !!!

3) Description des IAs implémentées.

3.1) Architecture Générale.

Notre implémentation du jeu Quoridor intègre plusieurs niveaux d'intelligence artificielle basés sur l'algorithme Minimax avec élagage Alpha-Bêta. Chaque IA utilise une fonction d'évaluation paramétrable ainsi qu'une table de transposition pour optimiser les performances.

3.2) Algorithme Minimax avec élagage Alpha-Beta.

L'algorithme Minimax est celui que nous avons utilisé pour l'IA. Il permet d'explorer l'arbre des possibilités de jeu jusqu'à une certaine profondeur et d'évaluer les positions correspondantes.
Nous avons optimisé Minimax grâce à l'élagage Alpha-Bêta, qui réduit le nombre de nœuds explorés en évitant l'évaluation des branches ne pouvant pas influencer le résultat final.

Notre implémentation présente plusieurs caractéristiques notables :

- Profondeur de recherche variable selon le niveau de difficulté
- Élagage Alpha-Beta pour optimiser les performances
- Table de transposition pour mémoriser les états déjà évalués et éviter les calculs redondants
- Paramètres ajustables pour personnaliser le comportement de l'IA

3.3) Fonction d’évaluation.

La fonction d’évaluation permet de déterminer la qualité d’une position.
Notre heuristique prend en compte plusieurs facteurs :

1. **Distance à l'objectif** (poids_avancer) : Calcule la distance entre le joueur et la ligne d'arrivée. Plus la distance est faible, meilleur est le score.

2. **Blocage de l'adversaire** (poids_bloquer) : Valorise les positions où l'adversaire est loin de son objectif.

3. **Gestion des murs** (poids_murs) : Considère l'avantage relatif en termes de murs restants.

4. **Avance relative** (poids_avance) : Bonus accordé lorsque le joueur est plus proche de son objectif que l'adversaire de son propre objectif.

3.4) Niveaux de Difficulté.

Dans le cadre de ce projet, il nous a été demandé de créer trois niveaux de difficulté, que voici :

Niveau Facile :

- Profondeur de recherche limitée à 1.
- Epsilon élevé (0.4) favorisant l'exploration aléatoire.
- Poids modérés mettant en valeur le déplacement direct vers l’objectif.
- Faible considération pour le blocage de l’adversaire.

Niveau Moyen :

- Profondeur de recherche de 2.
- Epsilon modéré (0.2) permettant de trouvé un équilibre entre l’aléatoire et la recherche.
- Poids équilibrés entre progression et blocage.
- Meilleure utilisation de la stratégique des murs.

Niveau Difficile :

- Profondeur de recherche de 3.
- Faible epsilon (0.1) mettant en avant les coups optimaux.
- Valeur forte pour bloquer l'adversaire et progresser.
- Utilisation très stratégique des murs.

3.4) Optimisations Techniques.

Plusieurs optimisations ont été implémentées pour améliorer les performances :

1. **Table de transposition** : Stocke les états déjà évalués pour éviter les calculs redondants
2. **Génération stratégique des coups** : Priorise les positions de murs les plus prometteuses
3. **Détection précoce des victoires** : Arrête la recherche si une victoire immédiate est possible
4. **Algorithme BFS optimisé** : Calcule efficacement les plus courts chemins vers l’objectif

3.4) Comportement Adaptatif.

L’IA intègre un paramètre epsilon, qui introduit une part d’aléatoire dans ses décisions. Cela permet d’explorer différentes stratégies et d’éviter un comportement trop prévisible. Ce paramètre diminue avec le niveau de difficulté, rendant l’IA plus déterministe à mesure que la difficulté augmente.

4) Résultat des tournois.

Afin d'analyser la performance des différents niveaux d'IA, nous avons organisé un tournoi de 50 matchs entre tout les niveaux de difficulté dont voici le résultat :

Joueur 1	Joueur 2	Victoire J1	Victoire J2	Coups moyen
Facile	**Facile**	27	23	84,06
Facile	**Moyen**	6	44	81,80
Facile	**Difficile**	0	50	39,28
Moyen	**Moyen**	25	25	89,42
Moyen	**Difficile**	1	49	47,24
Difficile	**Difficile**	22	28	96,26

4.1) Analyse par tournois

1. Facile vs Facile :

Ce tournoi montre un équilibre entre les deux IA de niveau facile, avec un léger avantage pour J1. Cette quasi-parité s'explique par la faible profondeur d'exploration (c'est-à-dire 1) et par un facteur d'aléatoire élevé (epsilon = 0,4), qui introduit une forte variabilité dans les décisions. Le nombre moyen de coups (84,06) indique des parties relativement longues.

2. Facile vs Moyen :

La domination de l'IA Moyen démontre clairement la supériorité d'une plus grande profondeur d'exploration (2 contre 1). L'IA Moyen peut anticiper un coup de plus que l'IA Facile, ce qui lui permet d'éviter des pièges simples et de construire des stratégies plus efficaces. Le poids plus élevé accordé au blocage de l'adversaire (0,8 contre 0,5) lui permet également de mieux exploiter les faiblesses de l'IA Facile.

3. Facile vs Difficile :

La victoire écrasante de l'IA Difficile démontre l'importance de la profondeur d'exploration dans les jeux de stratégie. Avec une profondeur de 3, elle peut anticiper les conséquences de ses actions bien plus loin que l'IA Facile. Le nombre moyen de coups, très faible (39,28), indique que l'IA Difficile termine les parties rapidement, signe d'une stratégie efficace et bien structurée.

4. Moyen vs Moyen :

Ce tournoi présente un équilibre parfait, ce qui est cohérent puisque les deux IA utilisent des paramètres identiques. Le nombre élevé de coups par partie (89,42) indique des affrontements tactiques où les deux IA sont capables de contrer les stratégies l'une de l'autre.

5. Moyen vs Difficile :

Nous observons ici un résultat similaire à celui du tournoi opposant l'IA Facile à l'IA Difficile, ce qui confirme l'avantage d'une plus grande profondeur d'exploration. L'IA Difficile est capable d'anticiper un coup de plus que l'IA Moyen. Le nombre moyen de coups (47,24) montre que l'IA Difficile domine, mais met plus de temps à vaincre l'IA Moyen que l'IA Facile, ce qui est logique puisque l'IA Moyen commet moins d'erreurs.

6. Difficile vs Difficile :

Ce tournoi entre IA de niveau difficile montre un léger déséquilibre, possiblement dû à l'avantage du premier joueur dans le jeu Quoridor. Le nombre moyen de coups, très élevé (96,26), indique des parties particulièrement disputées, où chaque IA analyse et contre les stratégies de l'autre.

4.2) Analyse globale

Les résultats de ces tournois mettent en évidence plusieurs aspects clés de l'intelligence artificielle appliquée aux jeux de stratégie :

1. **Grande importante de la profondeur d'exploration** : L'écart de niveau entre les différentes IA s'explique principalement par leur capacité d'anticipation. Une IA capable de prévoir ne serait-ce qu'un coup de plus domine systématiquement son adversaire.

2. **Impact des paramètres d'évaluation** : Les poids attribués aux différents aspects du jeu (avancer, bloquer, utiliser les murs) influencent fortement la performance des IA. Celles de niveau supérieur utilisent des valeurs mieux calibrées, ce qui améliore la pertinence de leurs décisions.

3. **Avantage du premier joueur** : Le léger déséquilibre observé dans le tournoi Difficile contre Difficile met en évidence un avantage pour le premier joueur.

5) Bilan du projet.

Ce projet d'IA pour le jeu Quoridor a permis de mettre en œuvre l'algorithme Minimax avec élagage Alpha-Bêta et d'en évaluer l'efficacité à travers différents niveaux de difficulté.
Les analyses réalisées ont clairement démontré que :

1. **La profondeur d'exploration est déterminante** : Une IA capable d'anticiper un coup de plus domine ses adversaires, comme le démontrent les résultats de la partie 4.

2. **La valeur des paramètres d'évaluation influence la stratégie** : Les poids attribués aux différentes actions (avancer, bloquer, utiliser les murs) permettent de créer des comportements différents selon le niveau.

3. **Le rapport temps/performance** : L'augmentation de la profondeur d'exploration améliore les performances de l'IA, mais accroît significativement le temps de calcul.

Pour améliorer ce projet, nous pourrions envisager d'implémenter un apprentissage automatique des paramètres, d'optimiser davantage l'élagage, ou encore d'explorer d'autres approches, telles que les réseaux de neurones.

En conclusion, ce projet a permis de mettre en pratique les concepts abordés ce semestre dans le contexte des jeux de stratégie, et de constater l'efficacité des algorithmes lorsqu'ils sont correctement paramétrés.

6) Références.

Rapport :

- ***ChatGPT 4o*** : Pour la rédaction de ce rapport, nous avons utilisé ChatGPT afin de nous aider dans la rédaction, la reformulation et la correction des fautes d'orthographe.
- **Lien**: Nous nous sommes également appuyés sur les règles du jeu Quoridor. https://jeux-casse-tete.com/blog/regles-de-jeux/regle-du-jeu-quoridor

Développement :

- ***ChatGPT 4o*** : Pour le développement de notre projet, nous avons utilisé à certains moments un modèle de langage (LLM) afin de nous aider dans le développement :
  - Optimisation : Le LLM nous a aidés à implémenter une table de transposition, permettant de mémoriser les états déjà évalués afin d'éviter de recalculer plusieurs fois les mêmes positions. Il a également contribué à la mise en cache du hachage des états, afin d'éviter de le recalculer à chaque appel.
  - Correction de bugs : Le LLM nous a été d'une grande aide pour comprendre de nombreux bugs rencontrés au cours du développement.
  - Compréhension : Le LLM nous a également aidés à comprendre certains comportements des IA que nous avons développées.