Rapport de Projet : Jeu IA Hanabi

SOW Ismael & DIALLO Fatoumata Mbalou
Nom de l'Encadrant : Alexandre Niveau

21 mai 2024

Table des matières

1 Introduction

Ce projet vise à développer des IA pour le jeu Hanabi, où les joueurs collaborent sans connaître leurs propres cartes. L'objectif est d'explorer comment l'IA peut améliorer les stratégies de jeu et la coopération entre les joueurs.

1.1 Description du Jeu Hanabi

Hanabi, un jeu reconnu pour son originalité, un défi pour les amateurs de jeux de société. Conçu pour être joué de manière coopérative, il inverse un fondamental de nombreux jeux de cartes : les joueurs voient les cartes de tout le monde sauf les leurs. Chaque participant contribue à un objectif commun : **la création de spectacles de feux d'artifice** en plaçant les cartes dans le bon ordre sans connaître sa propre main. Ce principe crée une dynamique de jeu unique et un besoin de communication entre les joueurs.

Fin de jeu la partie se termine : si le troisième jeton rouge est utilisé, si tous les feux d'artifice sont complétés avec un score maximal, ou si la dernière carte de la pile est piochée.
Liste des composants : Hanabi contient 50 cartes réparties en 5 couleurs, avec des valeurs allant de 1 à 5, 8 jetons bleus pour donner les indices, 3 jetons rouges pour les erreur(une mauvaise carte jouée).

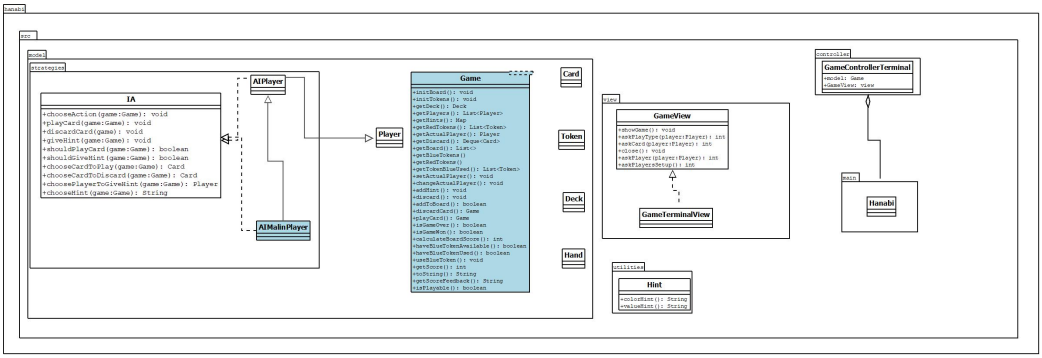
1.2 Objectifs du Projet

L'objectif du projet est de développer une IA avancée pour le jeu Hanabi qui peut non seulement jouer de manière autonome mais aussi prendre des décisions stratégiques basées sur les actions des autres joueurs. Ces objectifs spécifiques comprennent :

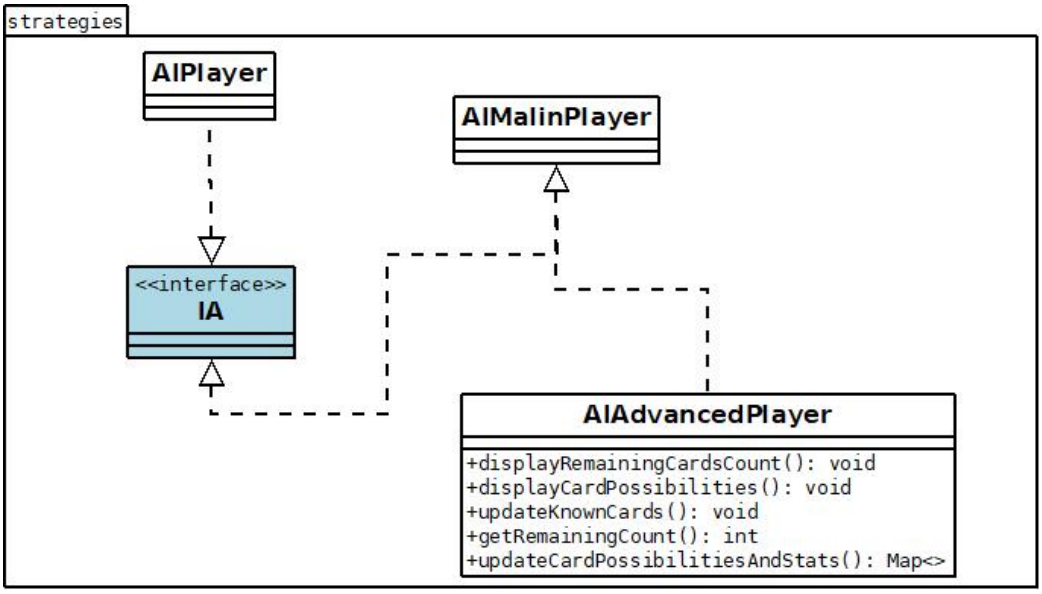
- Construire un modèle de jeu qui peut être facilement étendu et adapté.
- Implémenter des stratégies pour améliorer la prise de décision de l'IA.
- Évaluer les performances de l'IA dans différents scénarios de jeu et avec différents partenaires de jeu.

2 Architecture du Projet

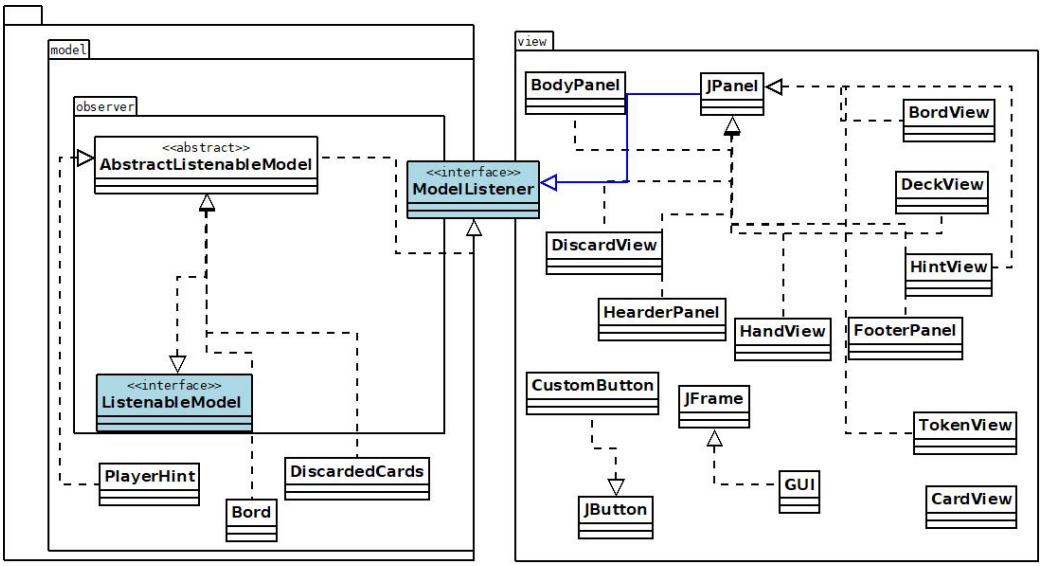
2.1 Vue d'Ensemble de l'Architecture MVC



2.2 Mise à jour package Modèle(strategies)



2.3 Mis à jour package View et Modèle



3 Description des Composants Principaux

3.1 Contrôleur

La classe `GameControllerTerminal` : Initialise l'état du jeu, traite les entrées des joueurs et orchestre la logique du jeu en appelant les méthodes appropriées du modèle de jeu. Ce contrôleur gère également la transition entre les tours des joueurs, qu'ils soient humains ou IA, en permettant à chaque joueur de choisir entre jouer une carte, la défausser, ou donner un indice. Le contrôleur fait appel à des méthodes de la classe `GameTerminalView` pour afficher l'état du jeu et pour demander des actions aux joueurs humains. De plus, il prépare le jeu en créant le deck, les joueurs, et en distribuant les cartes.

La classe **HomeController** gère l'écran d'accueil, traite les entrées des utilisateurs pour les joueurs IA, et lance le jeu en initialisant le deck et l'interface utilisateur.

3.2 Vue

La vue est responsable de l'affichage et de l'interface utilisateur. Elle comprend les éléments suivants :

- **Mise à jour de l'interface utilisateur** : les classes de la vue écoutent les changements dans le modèle de jeu et mettent à jour l'interface utilisateur en conséquence. Cela garantit que l'état du jeu affiché est toujours synchronisé avec l'état interne du jeu.
- **Affichage des éléments graphiques** : les cartes, les jetons, les indices et d'autres éléments du jeu sont affichés de manière graphique pour améliorer l'expérience utilisateur.
- **Gestion des interactions utilisateur** : la vue gère les interactions utilisateur, comme les clics sur les boutons pour jouer ou défausser des cartes, ou donner des indices. Ces interactions sont ensuite transmises au contrôleur pour traitement.
- **Séparation de la logique de présentation** : en séparant la logique de présentation de la logique de jeu, la vue facilite la maintenance et l'évolution de l'interface utilisateur sans affecter les règles du jeu.
- **Réactivité aux changements** : La vue est conçue pour réagir de manière dynamique aux changements dans le modèle, offrant une interface utilisateur fluide et réactive.

3.3 Modèle

Le modèle orchestre les éléments fondamentaux du jeu, y compris :

- **Initialisation et gestion des cartes** : le modèle gère la création, la distribution et le suivi des cartes tout au long du jeu. Il assure que chaque joueur reçoit une main de cartes et maintient l'état du paquet de cartes.
- **Supervision des actions des joueurs** : Le modèle surveille et applique les actions des joueurs, qu'ils soient contrôlés par l'IA ou des humains, en respectant les règles du jeu.
- **Logique de décision pour les IA** : les joueurs contrôlés par l'IA utilisent des stratégies de décision basés sur les indices reçus et l'état actuel du jeu pour déterminer leurs actions.
- **Conditions de victoire et de défaite** : le modèle évalue les conditions de victoire et de défaite, en vérifiant les piles de cartes jouées et les jetons disponibles.

— **Gestion des jetons** : le modèle gère les jetons bleus et rouges, essentiels pour le déroulement du jeu, en suivant les indices donnés et les erreurs commises.
— **Génération d'indices** : Le modèle génère des indices basés sur la couleur ou la valeur des cartes, aidant les joueurs à prendre des décisions stratégiques. Il s'assure que les indices sont pertinents en fonction de l'état actuel du jeu.
— **Interaction avec la classe Game** : Le modèle interagit étroitement avec la classe Game, veillant à ce que toutes les actions soient cohérentes avec les règles et l'état du jeu.

4 Développement des IA

Nous avons développé trois IA pour Hanabi : "AIPlayer", "AIMalinPlayer" et "AIAdvancedPlayer".

4.1 AIPlayer

La classe **AIPlayer** implémente une approche basique et aléatoire, se concentrant sur des actions simples dictées par le hasard.

4.2 AIMalinPlayer

AIMalinPlayer est conçue pour améliorer la prise de décision dans le jeu. Elle contient des strategies qui s'adapte intelligemment aux situations du jeu pour choisir entre jouer une carte, la défausser ou donner un indice.

4.2 Fonctionnement AIMalinPlayer

L'IA utilise **chooseAction** qui sert à décider et exécuter la meilleure action dans le jeu. Elle commence par analyser les indices reçus à travers la méthode **analyseHints**, cette methode recueille les indices reçu par les joueurs, les traitent en se basant sur l'état actuel du jeu (les feux d'artifices, les cartes jouer et defaussés), pour verifier si ces cartes peuvent être jouées ou doivent être défaussées.
Pour les cartes à **jouer** elle les testent avec la methode **playCard** : Si feux d'artifice correspondant est vide et que la carte a une valeur de 1, elle peut

être jouée. Sinon la carte peut être jouée si sa valeur est supérieure d'un point à la dernière carte de la pile (feux d'artifice correpondant).
Pour la liste des cartes à **defausser** après le traitement des indices reçu elle prend toutes les cartes qui n'ont pas été classés comme cartes jouables.
Après cette classification, la méthode décide si le joueur doit jouer une carte, défausser une carte, ou donner un indice à un autre joueur.
Pour jouer ou defausser une carte, on prend la premiere carte de la liste (des cartes à jouer ou à defausser).
Pour donner une indice, on choisi le joueur qui a le plus de carte bénéfique pour l'état actuel du jeu en verifiant si on a des jetons bleu disponibles aussi.

5 SECOND SEMESTRE

6 Développement de AIAdvancedPlayer

AIAdvancedPlayer représente une évolution de AIMalinPlayer, offrant une capacité d'analyse approfondies des cartes. Le but avec cette classe est de permettre au joueur "qui ne voit pas ses cartes" d'avoir une idée de ce qu'il peut detenir comme cartes (avoir une connaissance de la possibilite des cartes qu'il a), en exploitant les indices reçu, les cartes joués et defaussées et les cartes que detiennent ces coéquipiers.

6.1 Méthodes d'Analyse des Possibilités et Identification des Cartes Connues

Les méthodes `updateCardPossibilitiesAndStats` et `generatePossibleCards` jouent un rôle important dans le developpement de AIAdvancedPlayer. Elles permettent de déterminer quelle cartes peuvent être jouées ou défaussées en fonction des indices reçus et de l'état actuel du jeu, optimisant ainsi les décisions stratégiques de l'IA.

6.2 Fonctionnement de `updateCardPossibilitiesAndStats`

Cette méthode commence par traiter tous les indices reçus par le joueur, qu'ils soient relatifs à la couleur ou à la valeur des cartes. Ces indices sont utilisés pour mettre à jour (`cardsToPlayByColor` et `cardsToPlayByValue`),

qui stockent les positions des cartes correspondant aux indices reçus. Ensuite, pour chaque carte dans la main du joueur, elle appelle `generatePossibleCards` pour générer un ensemble de cartes possibles à cette position. Par la suite `updateCardPossibilitiesAndStats` filtre les possibilités de cartes en fonction des indices de couleur et de valeur reçus. Si un indice de couleur est reçu pour une position donnée, toutes les cartes possibles qui ne correspondent pas à cette couleur sont exclues.
De même, si un indice de valeur est reçu, toutes les cartes possibles qui ne correspondent pas à cette valeur sont également exclues. Cette filtration réduit les possibilités des cartes de la main du joueur et mieux cibler les décisions à prendre.

6.3 Fonctionnement de `generatePossibleCards`

`generatePossibleCards` se concentre sur la génération des cartes possibles en fonction des cartes visibles dans le jeu. Elle commence par compter les cartes dans les mains des autres joueurs, ainsi que celles qui ont été jouées et défaussées. Ensuite, pour chaque couleur et valeur de carte possible, elle applique des règles d'exclusion basées sur le nombre de cartes visibles et le nombre de cartes restantes dans le jeu. Les cartes qui ne sont pas exclues et qui sont encore disponibles sont ajoutées à l'ensemble des cartes possibles.

Cette méthode garantit que seules les cartes valides et disponibles sont considérées comme possibilités, en tenant compte de l'état global du jeu et des cartes déjà visibles.

6.4 Identification des Cartes Partiellement et Complètement Connues

L'identification des cartes partiellement et complètement connues est gérée par la méthode `updateKnownCards`. Elle examine chaque carte dans la main du joueur pour déterminer si sa couleur ou sa valeur est connue en se basant sur les indices reçus. Si les deux caractéristiques sont connues, la carte est classée comme complètement connue. Si une seule des deux caractéristiques est connue, la carte est classée comme partiellement connue.
Cette classification est essentielle pour que l'IA puisse prendre des décisions. Les cartes complètement connues peuvent être jouées ou défaussées en toute confiance, tandis que les cartes partiellement connues peuvent etre jouées ou defaussées en dernier recours.

6.5 Méthodes d'Affichage des Possibilités et Classification des Cartes

Les méthodes d'affichage des possibilités et de classification des cartes aident l'IA à visualiser les options disponibles et à catégoriser les cartes en fonction de leur jouabilité, inutilité ou importance.

6.6 Fonctionnement de `displayCardPossibilities`

`displayCardPossibilities` affiche les possibilités de chaque carte dans la main du joueur en fonction des indices reçus. Elle commence par mettre à jour les possibilités des cartes en appelant `updateCardPossibilitiesAndStats`. Ensuite, pour chaque carte dans la main du joueur, sa catégorie est determiné.

L'affichage se fait dans un tableau qui montre l'index de la carte, sa catégorisation, et les possibilités précieuses correspondantes.

6.7 Classification des Cartes

Les méthodes de classification déterminent si une carte est jouable, inutile ou précieuse :

- **Forcément jouable** : Une carte est classée comme jouable si toutes les possibilités pour cette carte le sont.
- **Forcément inutile** : Une carte est considérée comme inutile si toutes les possibilités pour cette carte sont inutiles.
- **Peut-être précieuse** : Une carte est considérée comme précieuse si au moins une possibilité pour cette carte est précieuse.

Ces classifications aident l'IA à décider si une carte doit être jouée, défaussée ou gardée en main.

6.8 Conclusion de AIAdvancdPlayer

En résumé, `updateCardPossibilities`, `generatePossibleCards`, et `displayCardPossibilities`, ainsi que la classification des cartes connues, forment le cœur de l'IA avancée. Elles permettent de filtrer et de déterminer les meilleures actions possibles en fonction des informations disponibles et des indices reçus, optimisant ainsi la stratégie globale du joueur. La méthode `displayCardPossibilities` aide à visualiser les possibilités de chaque carte

dans la main du joueur, tandis que les méthodes de classification permettent de catégoriser les cartes en fonction de leur jouabilité, inutilité ou importance.

7 Mode de jeu

— **Mode Console**
— **Mode Graphique**

L'interface graphique a été développée principalement pour les interactions entre les joueurs humains et les IA (AIMalin et AIAdvanced).

Pour tester exclusivement les interactions entre différentes IA, il faut choisir le **mode console**, qui est adapté pour observer les stratégies des IA sans intervention humaine.

A noté que l'interface graphique est mieux adapter à IA AdvancedPlayer.

7.1 Mode console

Le mode console du jeu permet de tester et d'observer les interactions entre différentes IA de manière efficace, et aussi faire joué des joueur Humain avec IA. Ce mode est particulièrement utile pour l'analyse des stratégies des IA sans l'intervention humaine.

Configuration

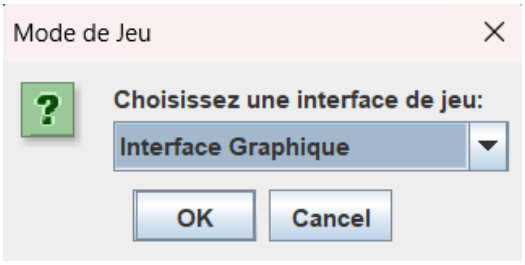
Combien de joueurs au total ? (2-5)
Ici donner le nombre total de joueurs.
Combien de ces joueurs sont des IA standard ? (0-5)
Ici indiquez parmis les joueurs ceux qui sont des IA.
Parmi les joueurs IA, combien sont des AIMalinPlayer ? (0-5)
Ici indiquez parmis les joueurs IA ceux qui sont des AIMalinPlayer.
Combien des joueurs IA restants sont des AIAdvancedPlayer ? (0-5)
Ici indiquez parmis les joueurs restant ceux qui sont des AIAdvancedPlayer.

7.2 Mode Graphique

Pour améliorer l'expérience utilisateur du jeu, nous avons développé une interface graphique pour le jeu. L'interface permet aux utilisateurs de jouer de

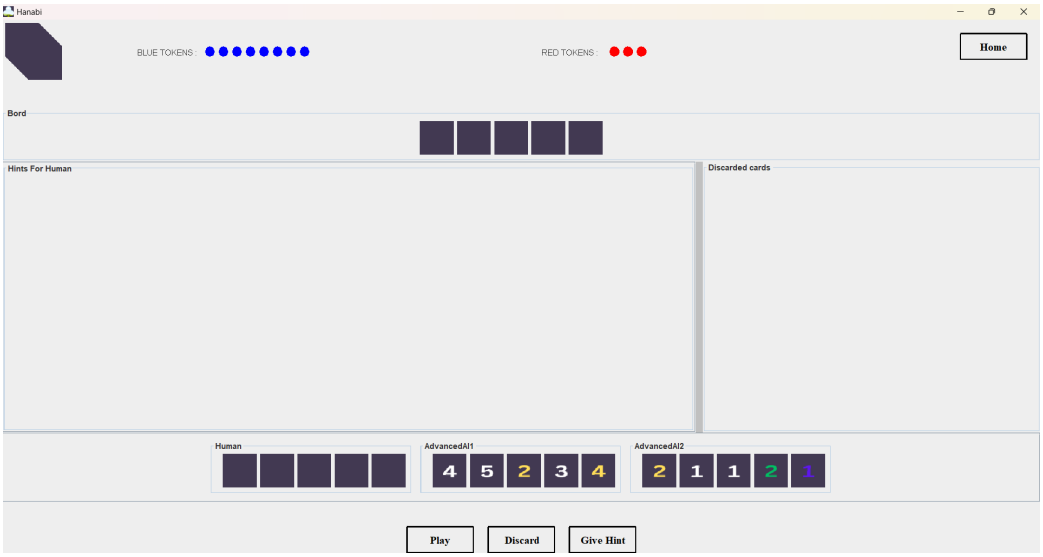
manière interactive en offrant une visualisation claire des cartes, des indices, des jetons et des actions possibles.

Voici un aperçu de l'interface :



L'écran de sélection permet aux joueurs de choisir entre le mode console et Graphique.

L'écran de configuration permet aux utilisateurs de spécifier le nombre de joueurs `AIMalin` et `AIAdvanced`, le joueur humain est crée automatiquement.



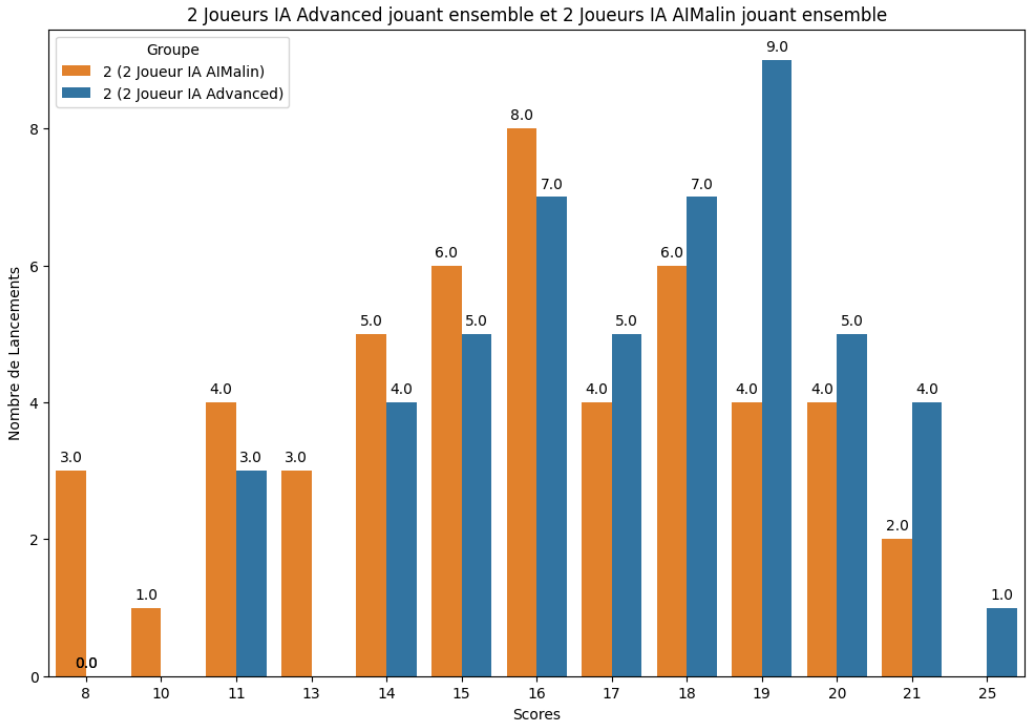
L'écran de jeu principal présente une vue d'ensemble du plateau de jeu, incluant :

- Les jetons bleus et rouges en haut de l'écran, indiquant les jetons d'indices disponibles et les erreurs respectivement.
- La main des joueurs, affichée en bas de l'écran, permettant de voir les cartes des autres joueurs.
- Des boutons pour les actions possibles : jouer une carte, défausser une carte, et donner un indice.
- Un panneau central affichant les feux d'artifice en cours de construction et les cartes défaussées.

8 Evaluations des performances des IA

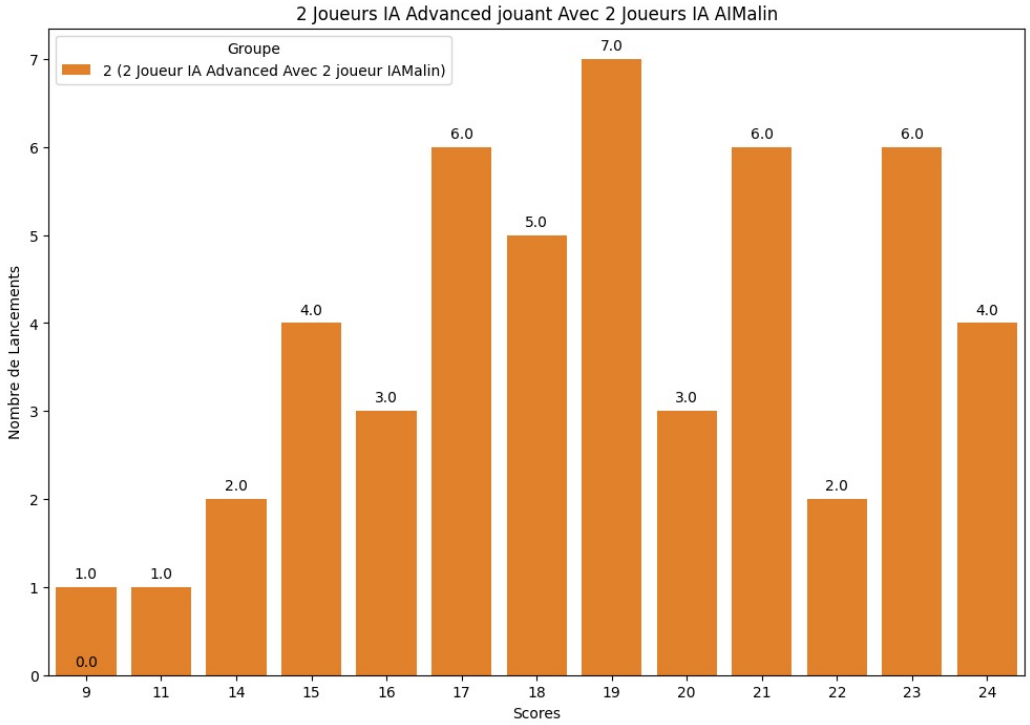
Dans les graphiques résultants, les scores obtenus sont représentés sur l'axe horizontal (X), tandis que le nombre de lancements correspondants est affiché sur l'axe vertical (Y).
Chaque barre du graphique illustre la fréquence à laquelle un score particulier a été atteint pour un nombre donné de lancements, le tout organisé en fonction du nombre d'IA participant dans chaque configuration. Des annotations précises sont apposées sur chaque barre pour indiquer le nombre exact de lancements ayant abouti à un score spécifique pour chaque configuration testée.

8.1 2 AIMalinPlayer et 2 AIAdvancedPlayer jouant séparément



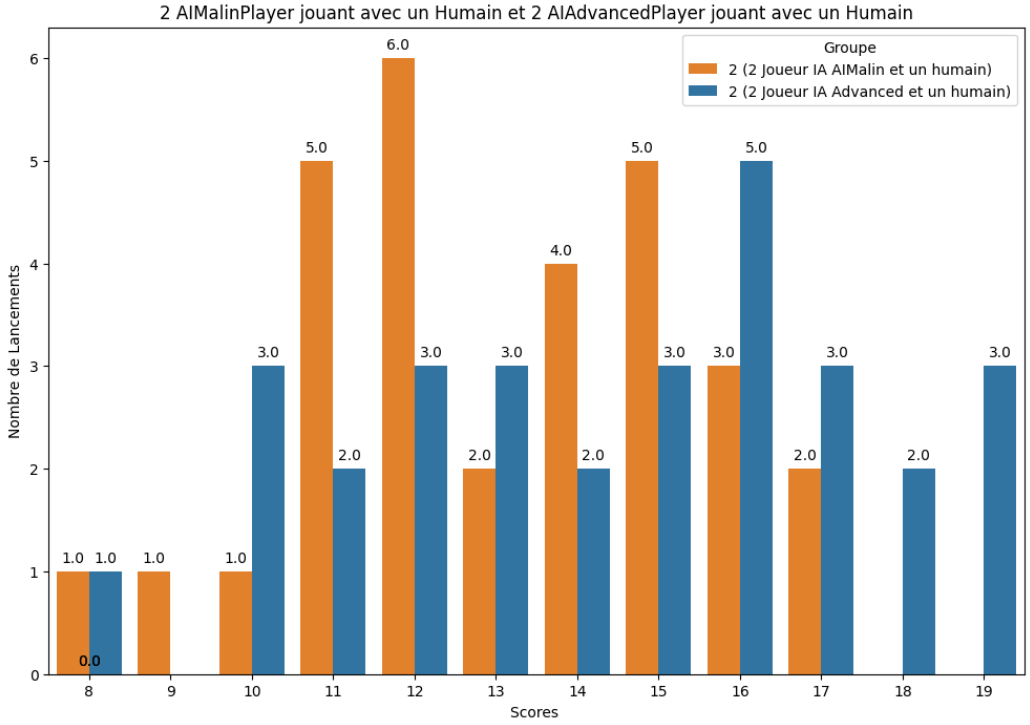
Les résultats montrent que AIAdvancedPlayer obtient de meilleurs scores, en particulier pour les scores élevés (19-18-20), où ils surpassent AIMalinPlayer. Les AIAdvancedPlayer atteignent plus fréquemment des scores très élevés avec un pic à 25 points, cela montre une meilleure stratégie que les AIMalinPlayer.

8.2 AIMalinPlayer et AIAdvancedPlayer jouant ensemble



Les scores sont particulièrement élevé on observe des points de (24-23 et 21) obtenu plusieurs fois. Montrant une meilleur collaboration entre les deux types d'IA .

8.3 AIMalinPlayer avec Humain et AIAdvancedPlayer avec Humain



Le joueur humain jouant avec AIAdvancedPlayer obtient de meilleurs scores notamment pour les scores de 18 et 19, où il atteint pour 2 et 3 lancements respectivement, contre 16 et 17 pour 3 et 2 lancements pour le joueur humain jouant avec AIMalinPlayer.

9 Difficultés Rencontrées et solution

Pour **AIMalinPlayer** nous avons rencontré des défis majeur, en particulier concernant l'analyse des indices reçu par les joueurs.
solution : avec la methode analyseHints() on a pu traiter les indices. Elle identifie les couleurs, les valeurs et positions des cartes, puis met à jour les ensembles de cartes jouables ou défaussables en conséquence. Cela permet de classifier les cartes de manière précise selon les indices reçus.

Pour **AIAdvancedPlayer** appliquer les indices de couleur ou de valeur en excluant les cartes non conformes et en s'assurant que seules les cartes

valides restent après le filtrage a été une tâche complexe.
solution : nous avons utilisé la méthode updateCardPossibilitiesAndStats. Elle traite les indices reçus, génère des possibilités pour chaque carte avec generatePossibleCards, puis filtre ces possibilités selon les indices de couleur et de valeur.

10 Conclusion

Le développement des IA pour le jeu Hanabi montre une progression significative dans l'efficacité des stratégies de prise de décision. Chacune des IA représente un niveau croissant de difficulté dans l'analyse des situations de jeu et la coopération entre les joueurs.

AIPlayer, la version la plus basique, repose sur des actions aléatoires, offrant une approche simple mais inefficace pour gérer les défis du jeu.

AIMalinPlayer marque une amélioration notable, intégrant des stratégies plus réfléchies. Elle utilise des méthodes telles que *chooseAction* et *analyseHints* pour optimiser les décisions, en tenant compte des indices reçus et de l'état actuel du jeu. Cette IA démontre une capacité à faciliter la prise de decision classant les cartes à jouer ou à défausser et en choisissant intelligemment les moments pour donner des indices.

AIAdvancedPlayer représente le niveau le plus haut de notre développement, offrant une analyse approfondie des possibilités des cartes. En exploitant des méthodes comme *updateCardPossibilitiesAndStats*, *updateKnownCards* et *generatePossibleCards*, cette IA permet une identification précise des cartes, qu'elles soient partiellement ou complètement connues. Elle utilise des indices et des données de jeu pour affiner ses décisions stratégiques, assurant ainsi une coopération avec les autres joueurs.
Les méthodes comme *displayCardPossibilities* et les classifications des cartes en jouables, inutiles ou précieuses offrent des outils visuels et analytiques pour améliorer la compréhension et l'efficacité de l'IA.

En somme, les progrès réalisés dans le développement de ces IA montrent une amélioration continue des capacités de prise de décision et de coopération dans le jeu Hanabi. Ces IA non seulement augmentent les chances de gagner, mais elles offrent également des connaissance précieuses sur l'application des stratégies dans des environnements complexes.