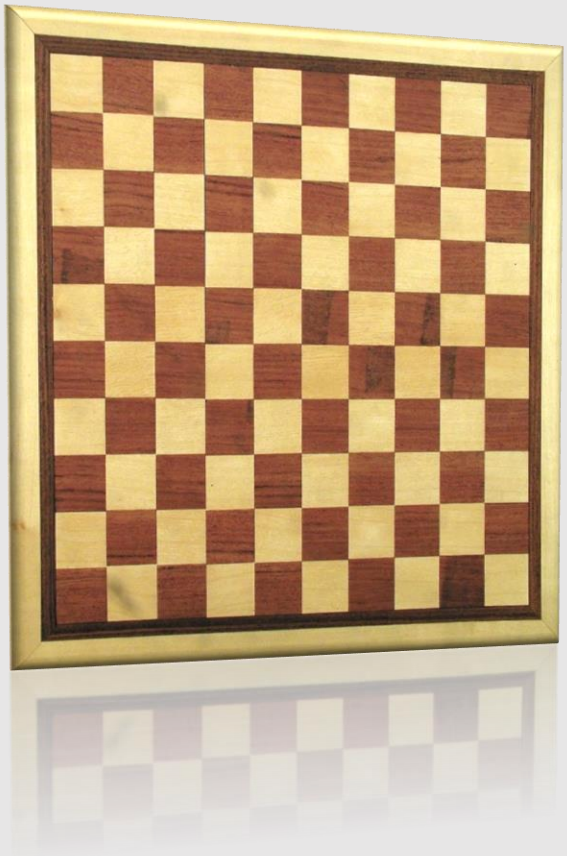
UNIVERSITÉ DE ROUEN NORMANDIE

Théorie des jeux
Projet : jeu des dames
2018 – 2019



Elias GUESSOUM
Gatien LAMULE
Malik BENCHEMAM
Sandra KLJAKIC
Paritosh Sharma MAHABIRSINGH
M1 GIL

Table des matières

Manuel d'utilisation

Lancement de l'application

Pour lancer l'application, il suffit de d'ouvrir le fichier **home.html** avec un **navigateur internet**

SCREEN MENU

Afficher les règles

Il suffit de cliquer sur le menu "**Règles**"

SCREEN REGLES

Lancer une partie

Pour lancer une partie :

- Choisissez un niveau de difficulté parmi les deux existants
- Cliquez sur **Start New Game**

SCREEN

Jouer une partie

S'il s'agit d'une partie **Humain vs IA** :

- **L'humain** se voit attribuer **les pions blancs** et **l'IA les noirs**
- **La partie débute dès que l'utilisateur effectue un déplacement**, c'est-à-dire lorsqu'il **fait glisser un de ses pions sur le plateau** avec sa souris en ***Drag n Drop***. L'IA répond automatiquement après que l'utilisateur ait terminé son coup.
- Si un pion de l'adversaire se trouve à portée de prise, il suffit de **glisser par-dessus ce pion** pour réaliser cette dernière. Faites glisser le pion souhaité là ou vous souhaitez le déposer

SCREEN Humain Vs IA

S'il s'agit d'une partir **IA vs IA**, vous pouvez :

- Voir la partie se dérouler en coups par coups avec le bouton ***next move***
- Laisser la partie se dérouler avec le bouton ***launch***

SCREEN IA vs IA

Descriptif des algorithmes utilisés

Minimax

L'algorithme **Minimax** est un algorithme s'appliquant aux jeux à somme nulle, **permettant de rechercher le meilleur coup à jouer pour le joueur courant** (dont c'est le tour), à partir d'une configuration de jeu donné.

Ceci se fait à travers la génération d'un arbre binaire, dont les nœuds représentent les différentes configurations du jeu à l'issue du coup précédemment joué, et dont chaque branche représente un des coups possibles pour l'adversaire du joueur courant. L'un des deux joueurs est référencé par « **Max** » et l'autre « **Min** ». La profondeur de l'arbre est définie préalablement.

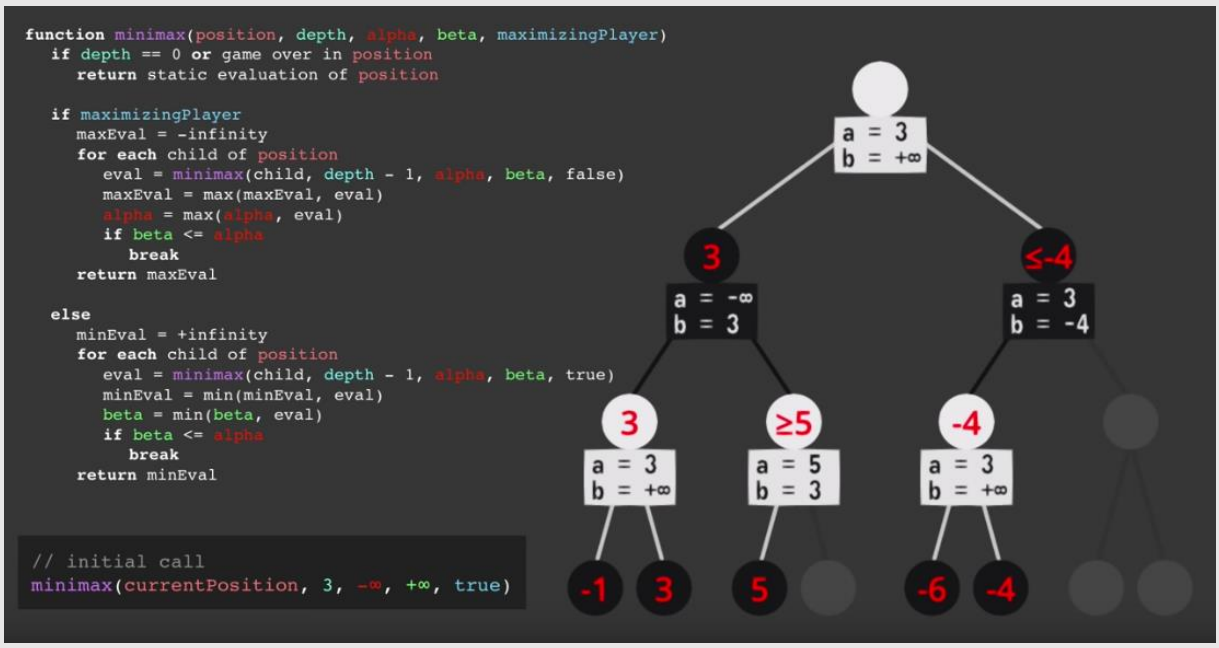
Une fois l'arbre généré, une valeur réelle appelée « **heuristique** » est attribuée à chaque feuille de l'arbre. Celle-ci est déterminée par une fonction dite d'« **évaluation** » et **permet d'indiquer à quel joueur la configuration représentée est favorable**. Les nœuds parents se voient attribuer une valeur parmi celles de leurs enfants : si le nœud est un « **nœud Max** » (resp. « **Nœud Min** ») (configuration où **Max** (resp. **Min**) était le joueur courant), la valeur choisie est **la valeur maximale** (resp. **minimale**) parmi celles des nœuds enfants. Finalement, la valeur prise par la racine permet de déterminer le meilleur coup à choisir pour le joueur courant, dans cette configuration.

Alpha-Beta

Pour notre jeu, bien que l'algorithme **Minimax** permette à lui seul de mettre en œuvre une intelligence artificielle contre laquelle jouer, le fait qu'il parcourt l'arbre généré en entier constitue un inconvénient quant à sa complexité.

L'élagage **Alpha-Beta** est une technique **permettant d'omettre l'évaluation d'une ou plusieurs branches de l'arbre** : en effet, il est possible d'éliminer un ensemble de sous arbres de l'arbre généré, si nous pouvons prouver que, à partir de certains nœuds, le joueur courant ne considérera emprunter un chemin ne favorisant pas sa stratégie. Pour cela, nous disposons de deux variables : la variable **Alpha**, qui permet **d'enregistrer la meilleure heuristique pour Max**, et **Beta**, **qui en fait de même pour Min**, toutes deux ajoutées en tant que paramètres à l'algorithme **Minimax**.

Voici un exemple de l'exécution de l'algorithme Minimax, utilisant le principe d'élagage Alpha-Beta :



Source : https://www.youtube.com/watch?v=l-hh51ncgDI

Implémentation des algorithmes et description des heuristiques

Notre approche

Notre implémentation du jeu de dames a été réalisée en JavaScript, suivant une architecture MVC :

- Modèle : nous disposons d'un modèle de données, regroupant les entités du jeu (telles que les joueurs, le plateau de jeu, les pions, les déplacements et les cellules) et les fonctions métiers permettant de les manipuler et de les utiliser afin de réaliser l'implémentation des algorithmes Minimax et Alpha-Beta.
- Vue : la vue dispose de fonctions réalisant des opérations sur l'interface graphique du jeu, selon les opérations effectuées dans le modèle.
- Contrôleur : le contrôleur définit les événements que l'utilisateur déclenche au travers des actions qu'il effectue lors du jeu. Ici, ces événements sont définis par des actions de glisser-déposer (*drag and drop*).

Modèle de données

Pour représenter les différentes entités intervenant dans le jeu, nous disposons des structures de données suivantes :

- board : correspond à une entité de plateau, représentée par ses cases (*cells*), ses pions (*pieces*), son joueur ayant le trait (*turn*) et d'un booléen indiquant la fin de partie (*gameOver*)

- piece : correspond à une entité de pion, représentée par sa position sur le plateau, c'est-à-dire sa ligne (*row*) et sa colonne (col), et par son état (***state***), c'est-à-dire le joueur à qui il appartient.
- cell : correspond à une entité de case, représentée par sa position (***row et col***), et par son état (*state*), c'est-à-dire le joueur qui l'occupe. Cet état vaut '*empty*' si la case est vide.
- move : correspond à une entité de déplacement, représentée par son type de déplacement (move_type), c'est-à-dire s'il s'agit d'un saut (de pion) ou d'un simple glissement, sa case de départ (*from*) et sa case d'arrivée (*to*).

Implantation

L'algorithme **Minimax** est implanté à l'aide des fonctions *min_value, max_value* et *heuristic_value*.

La fonction *heuristic_value* représente la **fonction d'évaluation** des feuilles de l'arbre des configurations : elle est élaborée à partir de plusieurs heuristiques, dont **la différence de pièces entre les joueurs, la différence de dames et la différence entre les heuristiques de position**. L'heuristique de position est une valeur attribuée à chaque pièce selon sa position sur le plateau : si une pièce se trouve sur une case appartenant à un rebord du plateau, cette valeur est plus importante que si elle se trouve sur les autres cases.

L'algorithme **Minimax** est décomposé en les fonctions *min_value* et *max_value*, permettant, respectivement, de **récupérer l'heuristique minimale** et **l'heuristique maximale** parmi les nœuds enfants du nœud courant, et **de parcourir l'arbre des possibilités de déplacements de chaque joueur**, à partir de ce dit nœud courant (qui est en fait une configuration). Celles-ci fonctionnent en s'appelant l'une l'autre, de la même façon que l'algorithme **Minimax** effectue son appel récursif lorsqu'il effectue le parcours de l'arbre des configurations.

La fonction *max_value* **recherche l'évaluation maximale** pour **Max** : une fois qu'elle dispose des nœuds enfants de la configuration courante, elle appelle la fonction *min_value* afin de **rechercher l'évaluation minimale pour Min**, qui elle-même appelle la fonction *max_value*, et ce jusqu'à entièrement parcourir l'arbre des configurations, selon la profondeur donnée.

De plus, ces fonctions mettent en œuvre l'algorithme **Alpha-Beta** : elles reçoivent en paramètres les valeurs Alpha et Beta, et la fonction *max_value* **met à jour la valeur Alpha** tandis que la fonction *min_value* **met à jour la valeur Beta**. Elles disposent toutes deux d'une instruction « *break* » **lorsqu'Alpha est supérieure ou égale Beta**, ce qui équivaut à un **élagage** de l'arbre des configurations.

Le reste des fonctions du modèle sont des fonctions utilitaires et métiers.

Joueur vs. IA

Pour cette partie de notre application, l'utilisateur se voit attribué les pions blancs, et l'IA les noirs. La partie débute dès que l'utilisateur effectue un déplacement, c'est-à-dire lorsque fait glisser un de ses pions sur le plateau. L'IA répond automatiquement après que l'utilisateur ait terminé son coup. Si un pion de l'adversaire se trouve à portée de prise, il suffit de glisser par-dessus ce pion pour réaliser cette dernière. Le pion pris disparait alors du plateau et le score du joueur ayant effectué la prise est mis-à-jour.

Quant à la réalisation de l'IA, nous disposons notamment des fonctions *alpha_beta_search* et *black_move*. La première se charge d'effectuer l'algorithme Alpha Beta à partir d'une configuration de plateau et selon une profondeur, toutes deux passées en arguments. Après le parcours de l'arbre des configurations, la fonction retourne le déplacement menant à l'heuristique la plus favorable pour l'IA. La fonction *black_move* utilise la fonction *alpha_beta_search* pour récupérer ce déplacement afin de le réaliser sur le plateau de jeu à l'aide de la fonction *movePiece.*

IA vs. IA

Pour la partie de jeu entre IA, l'utilisateur n'a qu'à appuyer sur le bouton *'next move'* : ce bouton permet d'effectuer, sur le plateau, le déplacement choisi par l'IA ayant le trait.

Pour la mise en place de ces deux IA, nous utilisons les fonctions suivantes :

- *alpha_beta_for_min*
- *alpha_beta_for_max*
- *white_move*
- *black_move*

La fonction *alpha_beta_for_max* s'avère posséder le même corps que la fonction *alpha_beta_search*, puisque dans les deux parties de l'application, l'IA jouant les pions noirs est désignée en tant que joueur Max. La fonction *alpha_beta_for_min* est une implantation de son antagoniste (*alpha_beta_for_max*) pour le joueur Min, c'est-à-dire l'IA jouant les pions blancs. La fonction *white_move* fait de même que la fonction *black_move*, mais pour le joueur Min.

Améliorations apportées

Ergonomie et design

Afin que l'expérience utilisateur soit la plus agréable possible, nous avons accentué le développement sur l'ergonomie de notre application ainsi que son design.
Pour cela, nous avons utilisés un langage de style CSS, ce langage nous a permis de concevoir une IHM (Interface Homme Machine) simple d'utilisation et intuitif.

Nous avons donc réalisé une maquette et établit une charte graphique avant même le développement, nous permettant de développer une IHM moderne et cohérente dans les meilleures conditions.

SCREEN JEU EN COURS

Pédagogie

Pour que le déroulement d'une partie de Dame soit pédagogique et instructif, nous avons implémentés plusieurs fonctionnalités permettant de faciliter la compréhension des différents algorithmes à l'utilisateur.

Un composant de logs est présent à droite de la grille, indiquant les différents résultats issus de l'exécution des algorithmes.

Conclusion

Bien que le sujet du projet ne soit pas difficile en soit, il fut plus long à réaliser qu'on ne le pensait. Nous avons éprouvé plusieurs difficultés lors de l'implémentation des algorithmes, leur exécution dépendant de nombreux facteurs et contraintes. Il a fallu effectuer un très gros travail de documentation afin de concrétiser nos idées.

Ce projet aura permis de mobiliser une très grosse partie des connaissances, que ce soit en théorie des jeux (avec les différents algorithmes abordés en cours) mais aussi en architecture logicielles (les différents principes et patrons de conception) ou en gestion projet (l'organisation ainsi que la communication autour du groupe).

Bibliographie