Institut national des sciences appliquées de Rouen

INSA de Rouen

Projet MSRO GM3 - Vague 2 - Sujet 1

TP4 : Pompe à chaleur réversible

Auteurs :
Thibaut André-Gallis
thibaut.andregallis@insa-rouen.fr
Kévin Gatel
kevin.gatel@insa- rouen.fr

Enseignants :
Nathalie Chaignaud
nathalie.chaignaud@insa-rouen.fr

07 Décembre 2021

Table des matières

0.1 But du projet :

Afin de mettre en $\frac{1}{2}$uvre les compétences acquises dans le codage en langage objet, nous avons choisi le projet consistant à implémenter l'algorithme Min-Max dans le cas d'un jeu de dames.

Nous avons donc réaliser des recherches pour comprendre ces concepts et pouvoir les implémenter. Cela nous a aussi permis de travailler en équipe avec des personnes différentes de nos précédents projets.

0.2 Explication de l'algorithme du Minimax

L'algorithme minimax ou minmax est un algorithme s'appliquant dans le cas d'un jeu (donc dans le cadre de la théorie des jeu) à deux joueurs lorsqu'il s'agit d'un jeu à somme nulle. Son objectif est de minimiser la perte maximum. Un jeu est à somme nulle si la somme des gains et des pertes de tous les joueurs est égale à 0, c'est à dire la perte de l'un est le gain de l'autre. Ce type de jeu répond à plusieurs caractéristiques, démontrées notamment par le théorème du minimax de Von Neumann, dès 1926 (présence de configurations d'équilibre, existance de l'algorithme...).

Le principe est globalement assez simple. L'ordinateur passe en revue tous les possibilités pour chaque pièce sur un nombre limité de coup, créant ainsi un arbre des possibilités (dont nous parlerons d'avantage dans la 4ème partie dans nos exemples). Ensuite chaque noeud se voit affecter une valeur en fonction des bénéfices du joueur et de l'adversaire. Le choix retenu sera la branche partant d'une feuille de cette arbre jusqu'à la racine, indiquant ainsi le coup qui doit être réalisé.

Un problème de mémoire se pose alors car chaque pièce peut se déplacer à gauche ou à droite (sauf si une pièce la bloque ou si le plateau de jeu ne continue pas) et cela sur un seul coup. Si on réalise plusieurs coups (ce qui est nécessaire), l'arbre devient rapidement très vaste. En pratique, on explore souvent, lorsque l'algorithme est optimisé une partie seulement de cet arbre à l'aide de méthodes dites d'élagage.

0.3 Explication de la méthode alpha-béta et de l'élagage

L'élagage alpha-bêta (ou ...) est une méthode grâce à laquelle on va pouvoir réduire la taille de l'arbre en enlevant certaines branches à l'aide de conditions choisies. Ainsi, on réduit le nombre de noeuds évalués et donc le temps de calcul de la branche "à choisir". Il s'agit d'une optimisation du minimax sans perdre des informations.

Cet élagage repose sur le fait qu'il n'est pas nécessaire d'examiner les sous arbres dont la configuration et le résultat ne permettra pas une amélioration du gain. On évalue pas les noeuds (et leur sous-arbre) dont la qualité (le gain, l'intérêt) sera inférieur à un noeud déjà évalué. Pour cela on va d'ailleurs travailler sur l'arbre dans un sens fixé, ici de gauche à droite.

0.4 Application de la méthode à des exemples

Pour mieux comprendre cette méthode et les différents cas possibles, voici plusieurs exemples (simplifier car on a choisit de représenter deux pions sur plusieurs coups en ayant attribué des valeurs aux noeuds).

0.5 Modélisation UML

0.5.1 diagramme de cas d'utilisation

L'utilisateur peut ainsi choisir de jouer à 2, de jouer contre l'ordinateur ou afficher les règles, stockées dans un fichier texte extérieur.

0.5.2 diagramme de séquences

Nous avons réalisé plusieurs diagrammes de séquences pour détailler la partie algorithme Minimax. Les voici :

0.5.3 diagramme de classes

Voici le diagramme de classe de notre projet. Celui-ci est susceptible d'évoluer jusqu'à la fin de notre code, il se sépare en deux parties : la première pour l'implémentation du jeu de Dames la seconde pour celle de l'algorithme Minimax.

Un certain nombre de fonction seront détaillées dans les parties suivantes.

0.6 Réalisation du projet

0.7 Implémentation du jeu de Dames

Nous avons tout d'abord fixé les règles que nous allions respecter, les variantes sur un jeu aussi populaire que les dames étant nombreuses.

Les pions ne peuvent pas se déplacer en arrière ni manger en arrière.

Il est obligatoire pour un pion de manger lorsqu'il peut.

Les dames peuvent manger en arrière et se déplacer dans les quatres diagonales sur l'espace de plusieurs cases.

Nous avons réalisé notre code en java, langage que nous maitrisions d'avantage. Notre but va être de nous centrer sur la partie Algorithme Minimax et d'y incorporer si vous pouvons une amélioration de type alphaBeta.

0.8 Implémentation de la partie Algorithme Min Max et ses améliorations

Le coeur de ce projet ce trouve dans cette partie. Nous avons d'abord fait un travail de recherches pour maîtriser ces concepts pour avancer ensuite plus rapidement et sereinement. Nous avons découpé notre travail comme présenté dans la partie 5 avec la modélisation UML.

La fonction euristique est la pierre angulaire de cette partie, nous avons donc réfléchis et nous sommes renseignés sur les différentes positions (positions imprenables sur les côtés du plateau car les pions ne peuvent y être mangé ...) pour établir un ordre de priorité entre toutes et une valeur choisie sur une échelle de -5 pour la moins intéressante (position pour une dame de se faire manger) et 14 pour la plus intéressante (faire une dame). Cette échelle est encore sucesptible d'évoluer selon l'euristique que nous auront choisi puis si nous en testons plusieurs. Un seul déplacement peut en combiner plusieurs car un pion peut se mettre en position de manger et d'être manger aussi.

L'action de manger étant obligatoire nous ne passont pas par la fonction euristique lorsque cette action se présente. C'est une des optimisations que nous comptons mettre en place.

Nous avons aussi une fonction listecoups qui établit pour une composition donnée, une liste dynamique des positions possibles pour chaque pions. Cette liste est ensuite utilisée pour construire l'arbre vide sous forme de liste chainée puis de le remplir avec les valeurs des noeuds grâce à la fonction euristique.

Enfin, nous avons la fonction parcoursminimax qui parcourt l'arbre avec l'algorithme minimax pour ressortir la branche la plus intéressante au niveau du gain.

En bonus nous essayerons de rajouter la fonction élagage qui réalisera l'élagage détaillé dans les parties 3 et 4 juste avant la fonction parcoursminimax.

0.9 Pour aller plus loin :

Comme dit précédemment nous ne sommes qu'au milieu de notre projet et nous n'avons donc pas fini les fonctions de la partie implémentation du mininmax.

Nous aimerions réaliser la fonction élagage, ainsi que plusieurs euristiques que nous comparerions pour obtenir la meilleur (il s'agit de la difficulté à laquelle sont confrontées les personnes ayant travaillé sur cet algorithme etc : choisir la bonne euristique).

Si le temps qu'il nous reste nous le permet nous aimerions pouvoir afficher à la demande du joueur, l'arbre des possiblités de sa composition actuelle et la branche choisie.

Nous voulons aussi réaliser un menu pour notre jeu avant le début de la partie où l'on peut choisir de jouer à deux, seul contre l'ordinateur ou bien d'afficher les règles.

0.10 Conclusion :

Cette première partie nous a permis de nous renseigner sur la théorie des jeux, les spécificité du jeu de Dames (les différents types de positions des pions...). Nous avons aussi commencé à coder après avoir réalisé nos diagrammes en essayant de rendre notre code le plus portable possible, tout cela en travaillant en équipe à trois. Nous allons poursuivre le codage de notre programme, tout en réalisant

certaines parties en pseudo-code d'abord car la fonction euristique est très importante mais aussi assez importante.

Annexe