



**École Nationale Supérieure d'Informatique pour l'Industrie et  
l'Entreprise**

**Intelligence Artificielle  
Rapport - Jeu de dames**

**Année universitaire 2015-2016**

**2<sup>ème</sup> année**

**Réalisé par : Romain EPIARD  
Fabien GOUGET**

# Table des matières

<b>1</b>	<b>Projet</b>	<b>3</b>
1.1	Règles du jeu de dames . . . . .	3
1.2	Algorithme du Min-Max . . . . .	5
1.3	Algorithme Alpha-Béta . . . . .	5
1.4	Fonction d'évaluation . . . . .	5
1.5	Apprentissage et Mémorisation . . . . .	6
1.5.1	Sauvegarde et Chargement des données . . . . .	6
1.5.2	Recherche et utilisation des états mémorisés . . . . .	8

## Introduction

Nous avons choisi de développer l'intelligence artificielle d'un jeu de dame. Nous allons d'abord implémenter une IA de base, comprenant un algorithme de type Min-Max, un Alpha-Beta et une Fonction d'évaluation viable. Une fois cette IA implémentée, nous voudrions implémenter une IA un peu plus poussée, avec une mémoire. Cela lui permettrait d'apprendre les coups à ne pas faire, une fois un certain point atteint. Nous pensons que le plus difficile dans l'implémentation d'une mémoire serait de définir un point de non retour, qui nous permettrait de définir si l'on doit enregistrer une partie ou non dans la mémoire de l'IA. Mais tout d'abord, il nous faut implémenter les règles du jeu.

# Chapitre 1

## Projet

### 1.1 Règles du jeu de dames

Les règles suivantes sont celles que nous avons implémenté dans notre projet. Le jeu de dames se joue sur un plateau de 100 cases (10 cases par 10 cases). Chaque joueur possède 20 pions. Il y a deux joueurs. Un des joueurs utilise les pions blancs, l'autre les pions noirs. Les pions ne peuvent se déplacer qu'en diagonale. Pour prendre un pion du joueur adverse, il faut qu'il y ait une case vide de l'autre côté du pion à prendre.

Pour implémenter le jeu, nous avons commencé par récupérer une I.H.M. Cette interface homme-machine a été récupérée sur internet. Ensuite, à partir de l'interface récupérée, nous avons ajouté les règles du jeu de Dames décrites plus haut. Nous avons essayé de rester fidèle le plus possible aux règles officielles du jeu de Dames. Nous avons eu quelques problèmes lors de l'implémentation du déplacement des pions. En effet, un pion qui peut prendre un autre pion est dans l'obligation de le faire. Si il peut en prendre plusieurs à la suite, il faut gérer le déplacement du pion. Et c'est la prise de plusieurs pions à la suite qui nous a posé problème. Car il fallait détecter la prise d'un pion, mais aussi détecter si une fois qu'il avait pris un pion, il pouvait en prendre d'autres, et ainsi de suite. Comme cette étape nous a posé plus de problèmes que prévu, nous avons décidé de ne pas implémenter la transformation d'un pion en Dame, quand il arrive au bout du plateau. Nous avons à la place fait en sorte que quand un pion arrive au bout du plateau, il fasse demi-tour et reparte dans l'autre sens.

Les conditions de victoire de notre jeu sont les suivantes :

- L'autre joueur se trouve dans l'impossibilité de jouer,
- N'a plus de pions.

Le codage des règles du jeu était la première chose à faire. L'interface se lance immédiatement sur le plateau jeu. Quand on lance le jeu, on se trouve sur un damier, avec des pions déjà disposés. Si on souhaite changer de type d'Intelligence Artificielle, on dispose d'un menu horizontal en haut de la fenêtre. Dans les choix d'Intelligence Artificielle, trois types sont proposées :

- Algorithme du Min-Max,
- Algorithme de l'Alpha-Beta,
- Algorithme de l'Alpha-Beta avec une mémorisation des parties précédentes.

Pour finir, voici une capture d'écran de notre jeu.

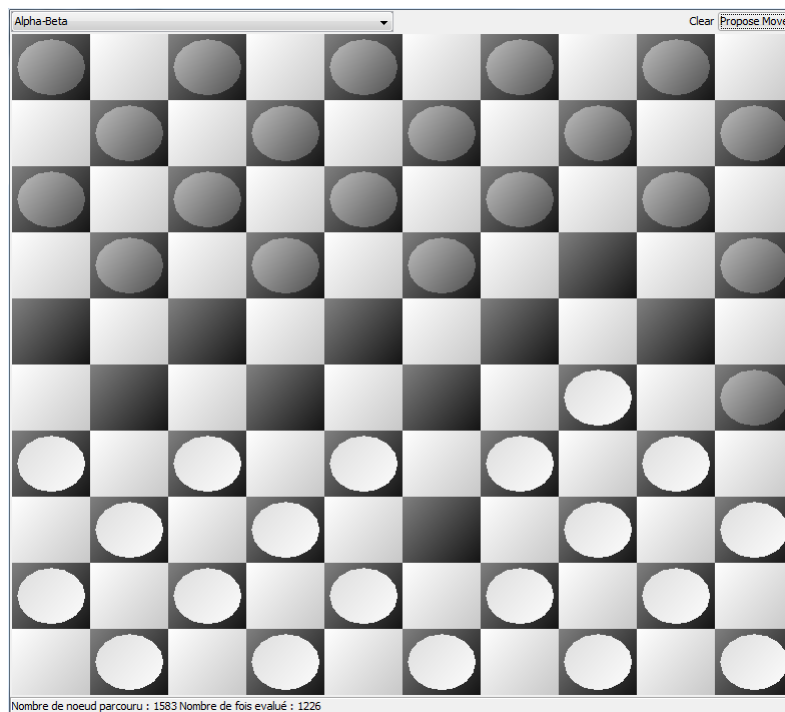


FIGURE 1.1 – Capture d'écran de l'interface du jeu

## 1.2 Algorithme du Min-Max

Une fois le jeu de dames implémenté, nous avons récupéré l'algorithme du min-max d'un ancien projet que l'un de nous avait réalisé. Il a fallu qu'on adapte l'algorithme, car au départ, il n'était pas générique. Nous l'avons donc rendu générique, pour pouvoir le réutiliser par la suite.

## 1.3 Algorithme Alpha-Béta

L'algorithme de l'Alpha-Beta a aussi été récupéré dans l'ancien projet. Nous l'avons aussi adapté en rendu générique.

## 1.4 Fonction d'évaluation

Pour pouvoir déterminer les nœuds les plus prometteurs, lors du parcours de l'arbre, nous avons codé une fonction d'évaluation. Celle-ci est utilisée par les deux algorithmes utilisés, à savoir le Min-Max, et l'Alpha-Béta.

Notre fonction d'évaluation est calculée en fonction des critères suivants :

- Le nombre de pions qu'il reste à chaque joueur. On l'appelle le critère matériel (mat).
- Le nombre de coups possibles pour chaque joueur. On l'appelle le critère de mobilité (mob).
- Le nombre de pions qui sont ou qui vont entrer dans la zone gagnante. C'est-à dire dans la zone de l'adversaire. Ce critère est appelé position (pos)

Ainsi, on définit notre fonction d'évaluation par un polynôme tel que :

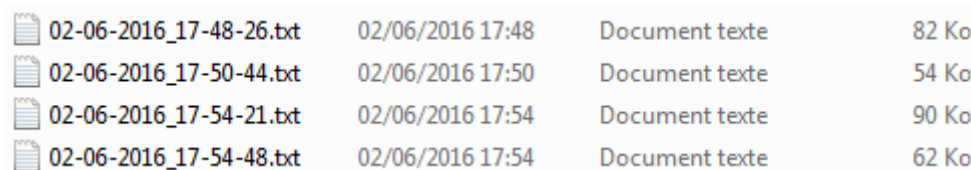
$$P_{mat}C_{mat} + P_{mob}C_{mob} + P_{pos}C_{pos}$$

Dans chacun des termes, on repère un coefficient, qui nous permet de déterminer l'importance de chaque paramètre. En d'autres termes, plus le coefficient multiplié à un critère calculé est élevé, plus le critère est important.

## 1.5 Apprentissage et Mémoire

### 1.5.1 Sauvegarde et Chargement des données

Finalement, nous avons implémenté un système de mémorisation des parties précédentes. Nous enregistrons les parties dans des fichiers textes. Ces fichiers textes sont enregistrés à l'endroit où se trouve l'exécutable du projet, ou sur Eclipse, à la racine du projet. Le nom de ces fichiers est constitué en utilisant la date et l'heure à laquelle l'enregistrement a été fait.







 02-06-2016_17-48-26.txt	02/06/2016 17:48	Document texte	82 Ko
 02-06-2016_17-50-44.txt	02/06/2016 17:50	Document texte	54 Ko
 02-06-2016_17-54-21.txt	02/06/2016 17:54	Document texte	90 Ko
 02-06-2016_17-54-48.txt	02/06/2016 17:54	Document texte	62 Ko

FIGURE 1.2 – Capture d'écran de fichiers de parties sauvegardées

On remarque sur nos fichiers textes, qu'ils n'ont pas tous la même taille. C'est dû au fait que suivant la partie, on ne sauvegarde pas le même nombre d'états du plateau.

Dans ces fichiers textes, chaque ligne correspond à un état du plateau de jeu (un tableau à deux dimensions, en somme). Sur chacune de ces lignes, pour chaque case du plateau, on a un chiffre suivi de deux points, suivi d'un autre chiffre. Le premier chiffre correspond à la couleur du pion sur la case en question :

- -1 : C'est une case vide. Aucun pion n'est placé dessus.
- 0 : C'est un pion noir qui se trouve sur cette case.
- 1 : C'est un pion blanc qui se trouve sur la case.

Les deux points suivants servent de séparateur entre les deux chiffres, pour dire qu'ils vont ensemble. Enfin, le deuxième chiffre correspond au sens de direction du pion :

- 0 Le pion se déplace vers le bas.
- 1 Le pion se déplace vers le haut.

Nous avons décidé de sauvegarder l'état du pion, comme nous avons décidé de ne pas implémenter les pions et de leur faire faire demi-tour, à la place. Par conséquent, pour que l'ordinateur se rappelle d'une façon réaliste le plateau précédent, il fallait garder le sens des pions.

```
1:0 -1:1 -1:1 0:0 -1:1 -1:1 -1:1 -1:1
1:0 -1:1 -1:1 0:0 -1:1 -1:1 -1:1 -1:1
1:0 1:0 1:0 1:0 1:0 1:0 1:0 1:0 1:0
```

FIGURE 1.3 – Capture d'écran des lignes dans un fichier

On remarque sur la figure 1.3 que la dernière ligne contient le même chiffre, c'est parce que la dernière ligne du fichier correspond à un plateau de même taille que les précédents, mais avec la couleur du gagnant, en premier chiffre.

Ainsi, prenons par exemple un état d'une case pour exemple, l'état 0 :0. Cet état signifie que c'est un pion noir, et qu'il se déplace vers le bas.



### 1.5.2 Recherche et utilisation des états mémorisés

Les fichiers nous permettent de construire un arbre de mémorisation. Chaque noeud de l'arbre représente un état du jeu (un plateau). Une fois cet arbre construit, nous appliquons un algorithme qui permet de récupérer les points de non retour des parties perdantes, pour un joueur qui utilise les pions noirs ou les pions blancs.

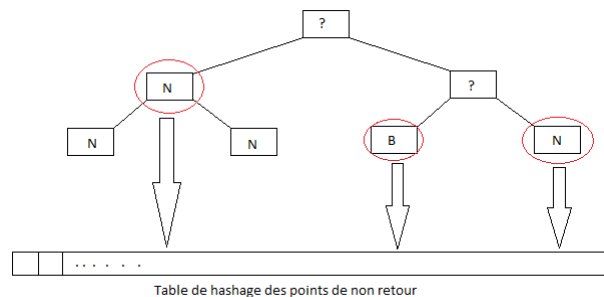


FIGURE 1.4 – Schéma explicatif de l'utilisation de notre mémoire

L'algorithme consiste à descendre jusqu'en bas de l'arbre et récupérer le pion perdant. En remontant dans l'arbre, on compare les fils de chaque noeuds. Si les deux fils sont égaux, on remonte la couleur du pion. Sinon, on remonte -1 et on enregistre tous les enfants qui ne sont pas égaux à -1. Cela nous donne les points de non retour possibles. Ces points de non retour sont enregistrés dans une table de hashage. Cette table est réutilisée dans l'algorithme Alpha-Beta, pour élaguer davantage l'arbre de décisions.

L'algorithme de calcul de la clé de hashage réalise une opération sur chaque case du plateau. Cette opération est la suivante :

Premièrement, chaque couleur de chaque case prend une nouvelle valeur (appelons cette valeur  $x$ ) :

- $-1 \Rightarrow 0$
- $0 \Rightarrow 2$
- $1 \Rightarrow 3$

Ensuite, pour chaque case, on incrémente un compteur élevé à la puissance  $x$  ( $x$  étant la nouvelle valeur correspondante à une couleur) :

$$\sum_{i=0}^9 \sum_{j=0}^9 (ij^x)$$

## Conclusion

Ce projet nous a permis de mettre en oeuvre les algorithmes vus en cours, et ainsi de les comprendre. Nous pensons que nous pourrions améliorer notre projet, tout d'abord en implémentant la transformation d'un pion en Reine, et aussi en réalisant un élagage de l'arbre de la mémoire. En effet, nous avons remarqué que nous enregistrons certaines parties, alors qu'elles n'ont pas vraiment lieu d'être enregistrées.