Othello

Camil BRAHMI Erwan LE CORNEC

13 mai 2018

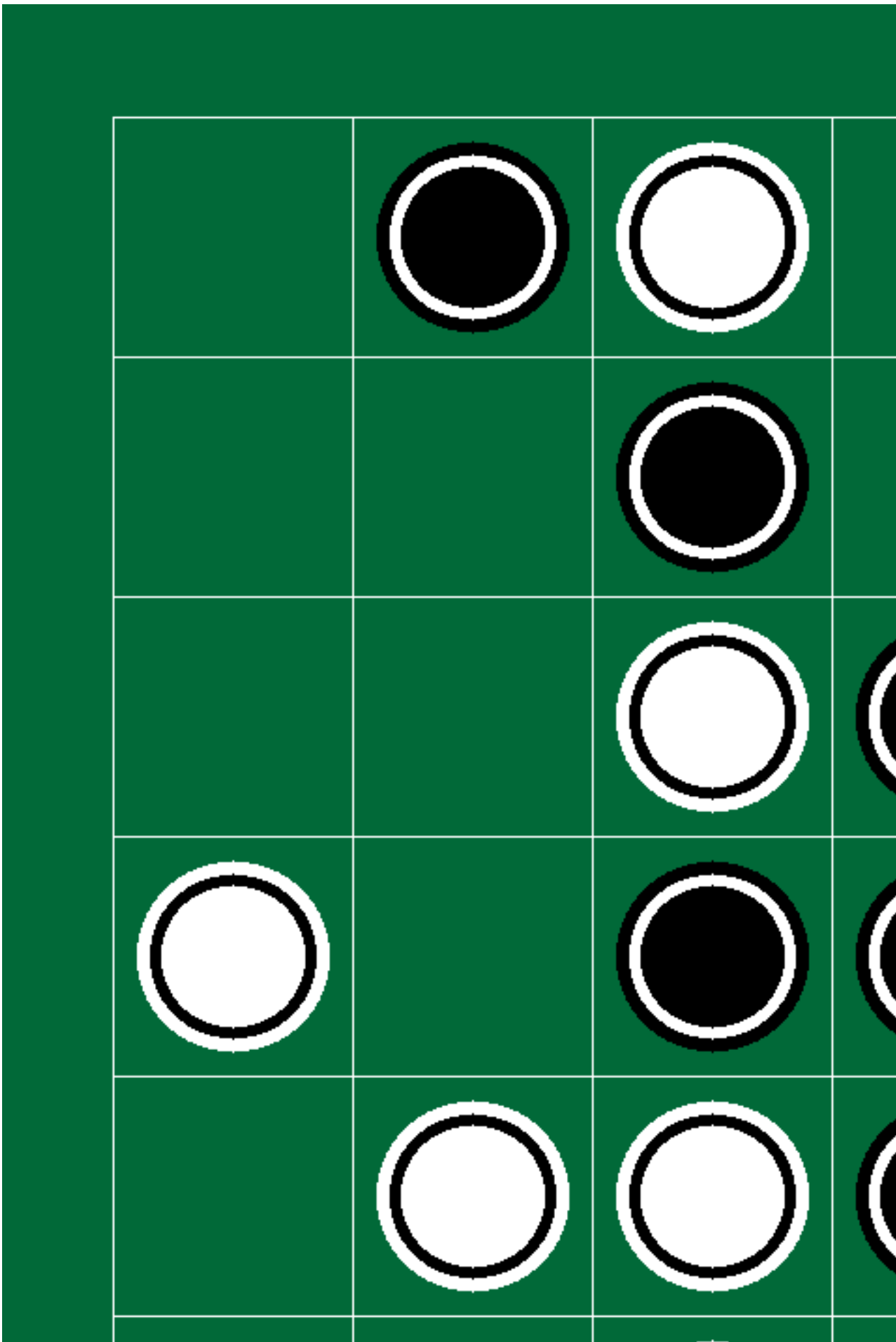

Table des matières

500	-150	30	10	10	30	-150	500
-150	-250	0	0	0	0	-250	-150
30	0	1	2	2	1	0	30
10	0	2	16	16	2	0	10
10	0	2	16	16	2	0	10
30	0	1	2	2	1	0	30
-150	-250	0	0	0	0	-250	-150
500	-150	30	10	10	30	-150	500

FIGURE 2 – Stratégie gagnante

1 Introduction

L'objectif de ce projet était de faire un premier pas vers la notion d'"Intelligence Artificielle" en codant le jeu de l'Othello. Dans ce rapport nous allons nous intéresser à l'I.A. de notre jeu, plus précisément à la façon dont nous l'avons codé et pourquoi de cette manière.

2 Moyens utilisés

— Machines avec distribution Ubuntu
— Pour nous faciliter le travail de groupe : la plateforme github

3 La fonction d'évaluation

4 Les niveaux de l'I.A.

4.1 Niveau 0

Pour ce niveau il suffisait simplement de compléter une liste de coups possibles pour une couleur choisie, puis jouer un coup au hasard.

4.2 Niveau 1, 2 et 3

Pour ces trois niveaux qui reposaient sur le principe d'un parcours d'arbre de possibilités et de notation des fils avec la fonction d'évaluation. Nous avons décidé de ne pas utiliser de structure d'arbre n-aire (int **plateau, struct arbre **fils, int nbfils) pour une raison d'optimisation.
En effet, supposons qu'un plateau a n coups possibles, donc n fils puis que chaque fils a, par exemple, n+1 coups possibles. Nous devons donc stocker 1+n*(n+1) noeuds en mémoire et nous sommes à peine a une profondeur

1 !
Pour remédier à ce problème nous avons pris la décision de faire de la façon suivante :

- Si nous avons pas encore atteint la profondeur souhaitée, copier le plateau (plateau_bis) puis jouer un coup (pile de coup possibles) sur celui-ci. Rappeler la fonction avec plateau_bis en paramètre et avec une profondeur de profondeur-1 tant que la pile de coups possibles n'est pas vide. Enfin libérer l'espace mémoire de plateau_bis.
- Si nous avons atteint la profondeur souhaitée, noter ce fils avec la fonction d'évaluation.

Donc avec l'algorithme minmax, c'est-à-dire pour les niveaux 1 et 2, il fallait regarder la note de tous les fils tandis qu'avec l'algorithme alpha-beta, pas besoin de parcourir tous les fils d'un noeud. Il est donc logique que le niveau 3 était plus réactif que le niveau 1 et 2 pour une même profondeur de parcours.

4.3 Niveau 4

Pour ce niveau, il s'agissait d'optimiser la gestion de la mémoire en utilisant qu'un seul plateau pour modéliser les coups possibles. Nous avons donc modéliser tous les coups possibles sur le même plateau de jeu que le joueur avec l'aide d'un pile de coups joués :

- À partir du plateau actuel, compléter la liste de coups possibles.
- Jouer un coup de cette liste et l'empiler sur la pile de coups joués et le stocker
- Répéter la même action (fonction récursive) jusqu'à avoir atteint la profondeur souhaitée et noter le plateau obtenue, stocker la note.
- Enfin on dépile jusqu'à revenir au plateau initial.

En sortie du programme on obtient le coup le mieux noté.

5 Conclusion