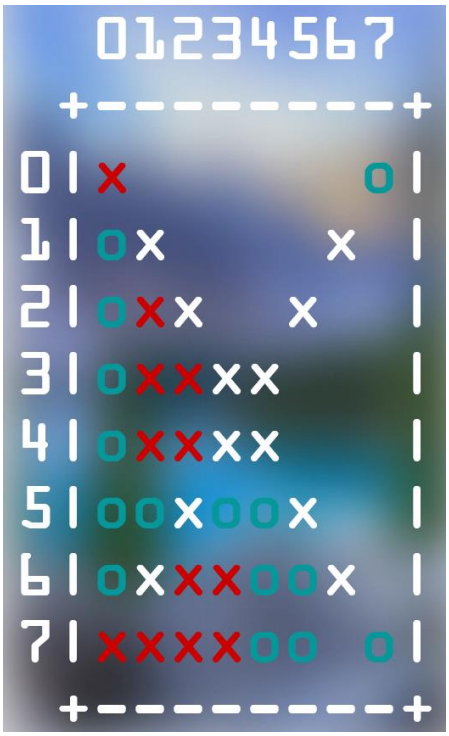
École nationale supérieure
d'informatique et de mathématiques appliquées
de Grenoble

Année 2021-2022

Projet Blobwar

Algorithmique avancée, algorithmes d'approximation, parallèles et probabilistes, complexité



Réalisé par :

LUO Wen
WANG Caroline

Encadré par :

BOULME Sylvain
DENNEULIN Yves
ROCH Jean-Louis
WAGNER Frederic

0. Introduction

L'objectif de ce TP est de mettre en place deux algorithmes, min-max et alpha-beta dans le jeu de blobwar afin de minimiser les chances de perdre à celui-ci. Pour cela, en plus d'implémenter ces deux algorithmes, nous allons essayer de les optimiser et effectuer des tests adéquats afin pour maximiser nos scores durant le **Tournoi BlobWar 2021/2022** !

1. L'implémentation

Afin d'implémenter les deux algorithmes min-max et alpha-beta, nous avons utilisé une méthode itérative comme nous le conseille leurs descriptions sur les pages associées.

L'algorithme min-max est séparée en 2 fonctions. La première ***MinMax*** est une fonction « interface » qui appelle une fonction récursive ***minmax***. Tant que la profondeur n'est pas nulle ou qu'il reste des nœuds, nous passons à la profondeur suivante (ie une de moins) en faisant attention à prendre le minimum ou le maximum en fonction du cas que l'on veut étudier (joueur ou opposant).

L'algorithme alpha-beta possède la même structure en 2 fonctions que la précédente : la fonction ***AlphaBeta*** et la fonction ***alpha_beta***. Dans la seconde fonction, elle aussi récursive, nous travaillons toujours par une méthode d'itération tant que la profondeur n'est pas nulle ou qu'il reste des nœuds ; à chaque appelle récursive de la fonction, nous diminuons la profondeur. En fonction de l'appartenance du tour, nous choisirons de maximiser ou minimiser, et de changer la valeur de alpha ou de beta.

2. L'optimisation

Dans ce projet, nous avons choisi d'optimiser légèrement à l'aide d'un parallélisme l'algorithme min-max à l'aide de la fonction ***par_bridge***. En ce qui concerne l'algorithme alpha-beta, nous n'avons pas su introduire du parallélisme gracieux, et en voyant des résultats plutôt satisfaisants, nous avons décidé de continuer avec notre idée de départ.

Afin de réaliser des tests plus concluants, une introduction de parallélisme s'est faite dans l'algorithme greedy : cela nous a permis de pouvoir mieux observer les performances de nous deux algorithmes.

3. Les tests

L'évaluation de notre code s'est faite en interne (bataille entre nos propres algorithmes, mais aussi contre des adversaires de taille : nos camarades (l'amitié a été laissée de côté durant cette guerre sanglante. Nous tenons à préciser qu'aucun ordinateur ou humanoïde n'a été blessé durant ces évènements).

Pour ce faire, nous avons testé de faire varier à la fois la configuration du plateau de jeu, les algorithmes, leurs profondeurs, leur ordre de jeu (qui commence), , mais aussi les joueurs humains qu'il y avait sur place. Les combats humain vs humain n'étant pas très utiles pour l'expression de nos résultats, nous allons décider par la suite de pas les prendre en compte dans l'exposition de nos résultats.

4. Les résultats

Concernant la différence de performance entre l'algorithme de min-max et alpha-beta, pour une profondeur de 1, nous observons qu'alpha-beta est :

- 2 à 3 fois plus rapide,
- 6 à 7 fois pour une profondeur de 2,
- 10 à 15 fois pour une profondeur de 3,
- 10 à 20 fois pour une profondeur de 4
- 15 à 30 fois pour une profondeur de 5

En termes de performance simple, min-max dépasse la seconde à partir de la profondeur 5, alors qu'alpha-beta ne le fais qu'a partir de la profondeur 7.

Temps observé pour MinMax(5)

Temps observé pour AlphaBeta(7)

En termes de matchs gagnés, nous avons observé de nombreux matchs nuls pour de mêmes profondeurs (peu importe l'algorithme utilisé), mais dès que la profondeur devait être différente, il y avait une différence de niveau : nos algorithmes respectent donc bien les consignes.

Conclusion :

Nous obtenons des résultats assez satisfaisants et sommes très heureux et de pouvoir participer au tournoi.

Par contre un problème que nous n'arrivons pas à expliquer quelque chose : l'algorithme AlphaBeta est très bon en termes de résultat pour des nombres impairs mais pas pour des nombres pairs. Nous pensons cependant qu'il doit y avoir un petit souci dans la détermination du tour du joueur. Ainsi, nous pensons qu'il serait préférable de jouer avec notre meilleure arme : **AlphaBeta(5).**