Algorithmique avancée: Blobwar

Anatole Gallouet & Rémy Lavainne

11 avril 2018

1 Travail demandé

Les algorithmes Greedy, MinMax et AlphaBeta sont fonctionnels, il est probablement possible d'en améliorer la performance, en effectuant les calculs en parallèle par exemple.

2 Extensions

Toutes nos idées d'extension avaient pour but de diminuer le nombre d'appels de l'algorithme AlphaBeta.

La première était simplement de choisir aléatoirement une partie des branches de l'arbre et de n'évaluer que celles-ci, cependant cela n'est pas du tout puissant car les mouvements possibles au sein d'une branche sont trop différents et l'algorithme devait couper trop de branche pour réellement gagner en efficacité, il se retrouvait alors à rater des coups évidents et perdait quasiment toutes ses parties, même contre l'algorithme glouton.

Notre deuxième idée était de faire varier la profondeur choisie durant les appels récursifs. Le principe est le suivant :

La fonction prends en paramètre un entiers supplémentaire n et en manipule un autre k. n représente le un nombre de pions (initialement 1), et k le nombre de coups déjà rencontrés qui mangeaient au moins n pions à l'adversaire dans l'immédiat (sans descendre dans l'arbre récursif). L'algorithme se contente de calculer k et dès que celui ci atteint une valeur seuil arbitraire, on incrémente n de 1, si aucun coup n'est trouvé avec un score supérieur à n, on décrémente alors celui-ci. L'idée est ensuite de ne pas évaluer les coups qui mangent moins de n pions.

Cet algorithme ne s'est pas révélé très efficace malgré de nombreux essais de valeurs de seuil et modification apportées. Nous l'avons implémenté sur MinMax car au moment ou nous avons eu l'idée, notre code AlphaBeta ne marchait pas encore (les erreurs ont été trouvées après, c'était principalement des erreurs d'inattention) mais notre idée était que si cela marchait sur MinMax cela devait aussi marcher sur AlphaBeta.

Notre dernière amélioration et en fait un cas particulier de la précédente, en effet l'idée pour réduire le nombre de coup était tout simplement de ne pas évaluer les coups qui ne mangent aucun pion. Cette méthode c'est révélée efficace car l'algorithme perd très peu en performance, en effet il est rare qu'un coup qui ne mange aucun pion soit le bon choix pour l'un des deux joueurs, cependant le gain de temps apporté n'est significatif qu'en début de partie et ne se remarque pas énormément dans une partie avancée.