Algorithmique avancé, TP2: Algorithme Min-Max

Nicolas Gast, Clément Pernet, Jean-Louis Roch, Frédéric Wagner

Année 2014-2015

Afin d'améliorer les performances de l'IA créée séance dernière on se propose d'implanter un algorithme plus sérieux : le min-max. L'algorithme devra prendre en paramètres la profondeur maximale visitée dans l'arbre de décision.

On utilisera pour le moment une fonction objectif simpliste : nombre de blobs rouges - nombre de blobs bleus.

Pour rappel, l'algorithme min-max fonctionne de la manière suivante : on parcourt récursivement l'arbre correspondant aux coups possibles. Arrivé à la profondeur limite, la fonction d'évaluation est utilisée pour estimer la valeur de chaque configuration. L'information est ensuite remontée vers la racine (voir fig.1), en même temps que les choix de chaque joueur :

— je choisis toujours le meilleur coup parmi tous les coups possibles (max sur toutes les évaluations)

— mon adversaire choisit toujours le pire coup parmi tous les coups possibles (min sur toutes les évaluations)

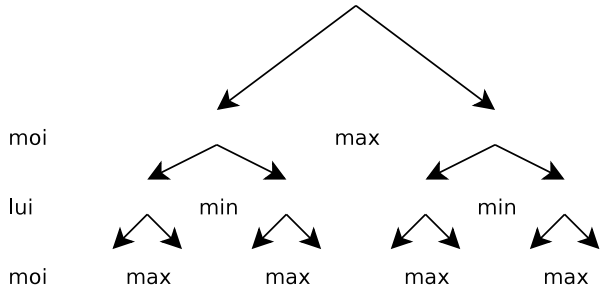

FIGURE 1 – minmax sur un jeu à 2 mouvements

Limitation dans le temps

Le temps de calcul disponible pour votre programme est limité dans le temps. Par défaut, 1 seconde est disponible pour éffectuer le calcul du meilleur coup. Ce temps peut-être changé grâce au paramètre `-t` de `blobwar`. Par exemple, lancez `./blobwar -t 5`

pour laisser 5 secondes de calcul à votre algorithme. Le programme principal (blobwar) récupérera le dernier coup que vous avez sauvegarder grâce à la fonction `_saveBestMove`. Plusieurs appels succéssifs sont possible, seul le dernier sera pris en compte. Si vous ne faites aucun appel à cette fonction avant la fin du temps imparti, vous perdez la parti.

1 Implémentation

Ajoutez à votre classe `Strategy` une méthode qui implémente la stratégie min-max :
`move& findMoveMinMax(move& mv, int profondeur)`
Modifiez la méthode `computeBestMove` pour qu'elle utilise `findMoveMinMax`.

2 Progression itérative de la profondeur

Afin de garantir l'obtention de bons résultats dans le temps limité imparti on utilise l'algorithme en augmentant de manière itérative la profondeur limite : on commence par une première recherche avec une profondeur maximale de 1, puis on lance une seconde recherche jusqu'à une profondeur de 2, etc... Si l'algorithme est stoppé au milieu de l'exploration de la profondeur 7, il utilisera le résultat calculé à la profondeur 6.

— Programmez une telle recherche itérative.
— Faites afficher à votre programme le temps de calcul pour chaque profondeur. Que pensez-vous de la méthode de progression itérative de la profondeur ?
— Est-ce bien de mémoriser les calculs déjà faits entre une itération et la suivante ?