

Un jeu à deux joueurs - IGI3006 - Michel Couprie

A. Le problème

On considère un jeu de plateau à deux joueurs : go, échecs, dames, othello/reversi, go-moku (morpion)... Dans la suite on se limitera au plus simple de ces jeux, le go-moku. Cependant, le principe que nous découvrirons pour ce cas particulier s'adaptera aux autres jeux de ce type.

Les règles du go-moku sont très simples : chaque joueur doit poser à son tour un pion (blanc pour le joueur 1, noir pour le joueur 2) sur un plateau de go (ou un damier, ou un échiquier...). Le premier qui arrive à constituer un alignement de 5 pions consécutifs de sa couleur, en ligne, colonne, ou diagonale, a gagné.

Notre but est de programmer un ordinateur pour qu'il joue à ce jeu contre un adversaire humain ou contre un autre ordinateur, et surtout, pour qu'il gagne !

B. Structure de données

Proposer une structure de données adaptée à la représentation "statique" d'un état du plateau de jeu, ou "position".

C. Analyse statique d'une position de jeu

Proposer un algorithme prenant en données d'entrée une position, et délivrant en sortie un diagnostic.

Dans un premier temps, le diagnostic sera sous la forme d'un entier valant -1000 si les noirs ont gagné, +1000 si les blancs ont gagné, et 0 si aucun des joueurs n'a gagné (partie en cours ou match nul).

Dans un second temps, on cherchera à affiner ce diagnostic. En effet, certaines positions semblent plus favorables aux noirs qu'aux blancs, ou inversement, en fonction des alignements partiels déjà réalisés. Dans les programmes jouant aux échecs par exemple, ce diagnostic se nomme "fonction d'évaluation", et l'habileté du programme à gagner des parties dépend principalement de sa qualité. Faites preuve d'invention, il peut y avoir plusieurs possibilités également pertinentes, et leur combinaison peut donner d'encore meilleurs résultats !

D. Un premier programme capable de jouer

Écrivez un schéma de programme permettant de mener à bien une partie, en utilisant avant chaque coup du programme la fonction d'évaluation pour choisir le coup qui amène la position ayant la meilleure évaluation d'après la fonction précédente.

E. Prise en compte des réponses possibles de l'adversaire, etc.

On veut maintenant tenir compte des possibilités de réponses de l'adversaire, puis des réponses que l'on pourra opposer à ces réponses, etc, pour affiner le diagnostic. On appellera "horizon" le nombre de ces niveaux de réponses à étudier : ainsi l'horizon 0 correspond au schéma de la question D, et ainsi de suite. Le nombre de niveaux explorés est idéalement le plus grand possible, cependant il est limité par le temps de calcul requis.

Dans le cas d'un plateau de jeu de 7×7 cases, donnez le nombre de positions à évaluer pour obtenir un horizon de $0, 1, 2, 3, 4, \dots, n$.

Ré-écrivez votre schéma de programme (voir question D) pour qu'il utilise une stratégie avec horizon n , l'entier n étant un paramètre fixé par l'utilisateur.

F. Pour aller plus loin

Quelles améliorations proposez-vous pour améliorer l'efficacité de cette méthode ?