Introduction à l'IA - Recherche en présence d'un adversaire

Christophe Rodrigues

25 mars 2019

1 But du TP

D'après l'idée lancé par un étudiant, vous allez devoir développer et confronter vos IA. Vous allez devoir implémenter un minimax pour un jeu de Morpion (A faire en binôme à rendre dans 2 semaines) dans le langage de votre choix.

2 Détails de l'implémentation

Dans un premier temps il est nécessaire d'implémenter une structure de données permettant de répresenter les différents états possibles du jeu en lui-même. Ensuite il est nécessaire d'implémenter toutes les fonctions utiles au minimax:

- Actions(s) : listes les actions possibles
- Result(s,a) : applique l'action a dans l'état s
- Terminal-Test(s) : test si s est terminal(fin de jeu)
- Utility(s) : attribue une valeur à l'état s

3 Minimax

L'algorithme du minimax a été vue en cours. Ci-dessous le détail de son pseudo-code. Il est a noter que l'algorithme du minimax est avant tout un parcours en profondeur d'abord :

function MINIMAX-DECISION(*state*) **returns** *an action*
  return $\arg\max_{a \in \text{ACTIONS}(s)}$ MIN-VALUE(RESULT(*state*, *a*))

---

function MAX-VALUE(*state*) **returns** *a utility value*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow -\infty$
  for each *a* **in** ACTIONS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(RESULT(s, a)))
  return *v*

---

function MIN-VALUE(*state*) **returns** *a utility value*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow \infty$
  for each *a* **in** ACTIONS(*state*) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(RESULT(s, a)))
  return *v*

4 Alpha-Beta

Incorporer l'élagage Alpha-Beta présenté en cours à votre minimax. Ci-dessous le détail de son pseudo-code :

function ALPHA-BETA-SEARCH($state$) **returns** an action
  $v \leftarrow$ MAX-VALUE($state, -\infty, +\infty$)
  return the $action$ in ACTIONS($state$) with value v

---

function MAX-VALUE($state, \alpha, \beta$) **returns** *a utility value*
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow -\infty$
  for each a **in** ACTIONS($state$) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(RESULT(s,a), α, β))
    if $v \geq \beta$ **then return** v
    $\alpha \leftarrow$ MAX(α, v)
  return v

---

function MIN-VALUE($state, \alpha, \beta$) **returns** *a utility value*
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow +\infty$
  for each a **in** ACTIONS($state$) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
    if $v \leq \alpha$ **then return** v
    $\beta \leftarrow$ MIN(β, v)
  return v

5 Bonus

Rajouter en paramètre de votre minimax la possibilité de borner la profondeur de l'arbre. Il vous sera alors nécessaire de développer une fonction heuristique (comme pour le A*) afin d'estimer l'intérêt d'un état sans forcément développer toute une branche au-delà de la profondeur fixée. (il est a noté que le calcul de l'arbre se fera au fur et à mesure des actions.

6 Bonus

Si votre minimax et alpha-beta est bien modulaire alors vous pouvez l'adapter à d'autres jeux. Essayer de le déployer sur le jeu de puissance 4.