TP - n° 1

Algorithmes pour les jeux – Mise en œuvre en Java

Le but de ce TP est de vous familiariser avec les techniques de recherche dans les arbres de jeux. Vous allez écrire un programme JAVA permettant de jouer au jeu des dominos et au jeu de Nim vus en cours.

Vous ferez particulièrement attention à ce que la mise en oeuvre des algorithmes vus en cours reste *générique*. Les algorithmes que vous allez écrire doivent pouvoir en fait fonctionner pour n'importe quel jeu, pourvu que sa mise en oeuvre respecte les interfaces prévues par vos algorithmes. Ceci implique notamment que le code de vos algorithmes ne doit tenir compte d'aucune particularité du jeu des dominos mais uniquement s'appuyer sur les interfaces prévues pour la caractérisation d'un jeu.

Téléchargez l'archive mise en ligne à l'adresse suivante : `http://www.lri.fr/~ma/Polytech_App5/tp1.zip`

I) Organisation du code

Le code qui vous intéresse se trouve dans le répertoire `src/main/java`. Il contient :

- un package `jeux`, qui regroupe différentes classes liées à ce cours, lui même organisé en fonction des thématiques abordées. Il est lui même décomposé en :
  - un package `modèle`, regroupant l'essentiel des classes servant à la modélisation abstraite de jeux à deux joueurs (ou plus)
  - un packgage `alg` où se trouvent les algorithmes.
- un package `monjeux`, destiné à recevoir les modèles des différents jeux que l'on souhaite coder.

Afin de pouvoir décrire des algorithmes *génériques* sans avoir à faire de suppositions sur la nature des jeux réellement codés, un certain nombre d'interfaces sont proposées, qui modélisent les opérations essentielles qui doivent être implémentées pour pouvoir être traitée par des algorithmes. Par exemple :

- L'interface `PlateauJeu.java` permet de parler d'un plateau d'un jeu, sans avoir à faire de supposition sur les caractéristiques des jeux réellement traités. Seule un ensemble restreint de méthodes abstraites sont spécifiées.
- L'interface `CoupJeu.java` permet de représenter un coup lors d'une partie.
- L'interface `AlgoJeu.java` permet de parler d'un algorithme de jeu.
- ...

Pour modéliser un jeu réel il faut introduire des classes et des méthodes concrètes pour chacune de ces notions (PlateauJeu, CoupJeu, Heuristiques,...).

Dans le code qui vous est donné vous avez un exemple concret avec le jeu des dominos vu en cours. Vous trouverez également une classe PartieDominos qui montre comment utiliser le tout pour faire s'affronter deux joueurs artificiels utilisant un algorithme de recherche de type minimax.

Questions sur le jeu des dominos

1. Dans un premier temps, en observant la façon dont le jeu des dominos est modélisé, essayer de comprendre l'articulation des différents composants de cette modélisation.
2. La modélisation du jeu des dominos n'est pas encore tout à fait complète. Il manque en effet, les heuristiques d'évaluation. Compléter le fichier HeuristiquesDominos.java en ce sens, sachant que vous trouverez tout ce qui peut vous être utile pour la coder dans la classe `PlateauDominos.java`.

Algorithme de jeu

1. Complétez la classe `Minimax` donnée pour coder les algorithmes simples de minimax. Ajoutez à cette classe une technique permettant de compter le nombre de feuilles et de noeuds développés par la recherche.
2. Testez votre minimax contre vous-même sur une grille 4x4 puis sur une grille 6x6, en faisant varier la profondeur de recherche de votre algorithme.

Questions sur le jeu de Nim

1. Donner une modélisation du jeu de Nim et la coder dans le package `monjeu.Nim`.
2. En prenant comme heuristique $h(n) = N - n$ pour $n > 1$ (attention : $h(1) = -\infty$ et $h(0) = +\infty$ pour Ami, et $h(1) = +\infty$ et $h(0) = -\infty$ pour Ennemi), simulez une partie entière pour $N = 10$ entre deux I.A., l'une voyant à 2 demi-coups, et l'autre à demi-coups. Comparez ce résultat avec celui que vous avez calculé à la main.