UV IA41 – Responsable : Fabrice Lauri
Concepts fondamentaux en Intelligence Artificielle et langages dédiés

Examen Final - Printemps 2013
Juin 2013

Coefficient : 40%.
Aucun document autorisé.

Remarques et conseils :

1) Lisez **attentivement** chaque question avant d'y répondre.
2) Indiquez clairement sur votre copie le numéro de l'exercice avant d'y répondre.
3) Lorsque vous définissez un prédicat, son profil et sa définition formelle doivent être au moins indiqués. Un jeu d'essais est facultatif.
4) Expliquez autant que possible vos choix lors de la définition d'un prédicat ou d'une fonction.
5) Le barème défini ci-après est susceptible d'être modifié.

Exercice 1 : Questions de cours (4 points)

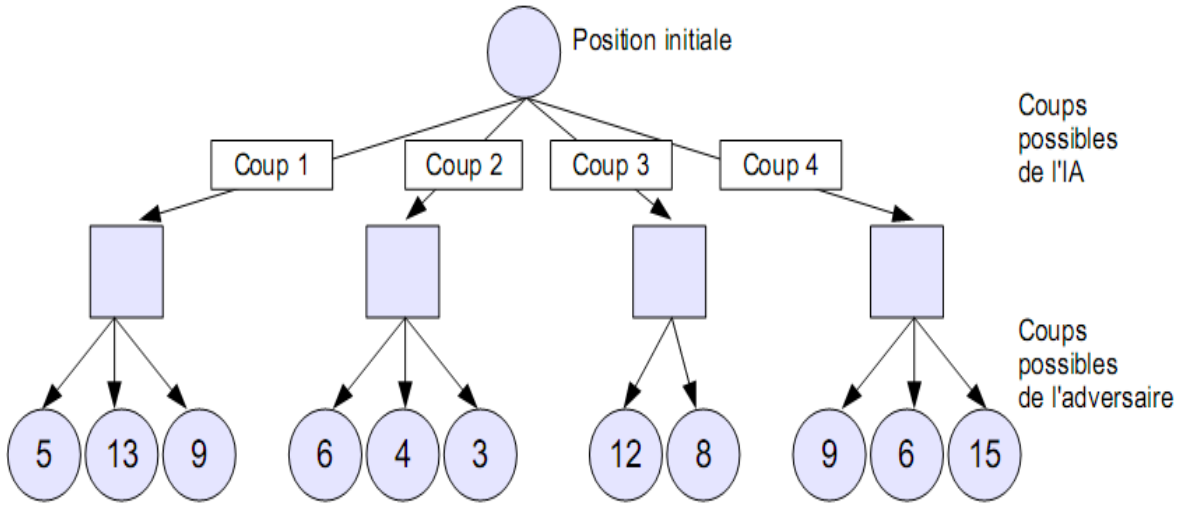
1. Quelle est la différence entre un arbre de jeu et un arbre de recherche ? **(1 point)**
2. Dans un programme Prolog, à quoi sert l'opérateur de coupure ? Quelle différence y a t'il entre une coupure verte et une coupure rouge ? **(1 point)**
3. Qu'affiche la fonction Lisp suivante : **(1 point)**

(eval (cons '* (list (cons '+ (cdr (list 5 8 7 10))) 3 2)))

4. Définissez aussi précisément que possible ce qu'est un problème de satisfaction de contraintes. Donnez plusieurs exemples.
   Rappelez la terminologie pour ce type de problème. **(1 point)**

Exercice 2 : Algorithmes Min-Max et Alpha-Bétâ (6 points)

Les algorithmes Min-Max et Alpha-Bétâ sont largement utilisés pour déterminer le meilleur coup qu'une intelligence artificielle (joueur cybernétique) doit réaliser lorsqu'elle est confrontée à un autre joueur dans le cadre d'un jeu à somme nulle.
L'arbre ci-dessous représente l'arbre de recherche d'un tel jeu à deux joueurs. L'évaluation de chaque position pouvant être atteinte à un niveau de profondeur 2 à partir de la position initiale est indiquée dans les feuilles de l'arbre.



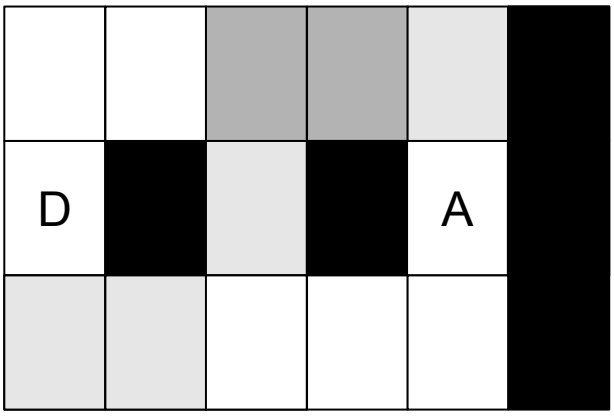
L'IA est supposée maximiser ses gains, tandis que l'adversaire les minimise.

Questions :

1. Rappelez l'algorithme Min-Max.
2. En utilisant l'algorithme Min-Max, précisez pour chaque nœud son évaluation et indiquez le coup qu'effectuera l'IA.
3. Rappelez l'algorithme Alpha-Bétâ.
4. En utilisant l'algorithme Alpha-Bétâ, précisez pour chaque nœud ses bornes finales [alpha; bétâ]. Indiquez également avec une croix les coups qui seront élagués par l'algorithme et donnez enfin le coup qu'effectuera l'IA.
5. Quel critère dans la construction de l'arbre de recherche permettrait d'améliorer l'efficacité de l'élagage de Alpha-Bétâ ? Si ce critère est appliqué, combien de nœuds dans l'arbre de recherche ci-dessus pourraient alors être élagués ?

Exercice 3 : Algorithme A* (6 points)

La figure ci-dessous représente l'environnement d'un personnage virtuel, que nous nommerons Jack, qui doit se déplacer de la case D (départ) vers la case A (arrivée).



A chacun de ses déplacements cardinaux (haut, bas, droite ou gauche), Jack consomme des points d'énergie, si bien qu'il désire atteindre A à partir de D en empruntant le chemin le moins coûteux en énergie.
Les cases noires représentent des obstacles infranchissables : aucune de celles-ci ne peuvent être atteintes par Jack.
Un point d'énergie est dépensé lorsque Jack arrive sur une case blanche, 2 points lorsqu'il arrive sur une case grise claire et 3 points sur une case grise foncée.
Pour résoudre le problème dont doit faire face Jack, utilisons l'algorithme A*...

Questions :

1. Donnez une représentation d'un état pour résoudre le problème exposé ci-dessus avec A*.
2. Exprimez la fonction successeur **succ(x)** pour chacun des états **x** nécessaires à la résolution.
3. Est-ce que la distance de Manhattan est une heuristique admissible ? Pourquoi ? Etayez votre réponse en donnant un exemple.
4. Utilisez une heuristique, si possible admissible, et résolvez ce problème avec l'algorithme A* en indiquant à chaque itération de l'algorithme (voir figure à la page suivante) :
   - le contenu de la liste Open,
   - le contenu de la liste Closed,
   - les valeurs de g, de h et de f pour le nœud courant extrait de la liste Open.

On suppose qu'à l'itération 1, la liste *Open* ne contient que le nœud correspondant à l'état initial D.

Itération	Liste *Open*	Liste *Closed*	Noeud N extrait de la liste *Open*	Valeur g du noeud N	Valeur h du noeud N	Valeur f du noeud N
1						
2						
3						
4						
...						

5. Combien de nœuds sont développés sans utiliser d'heuristique (c'est-à-dire que h(x) = 0 quel que soit l'état x)

Rappel : la distance de Manhattan est définie par : $d((x_1,y_1), (x_2,y_2)) = |x_1-x_2| + |y_1-y_2|$.

Exercice 4 : Prolog (4 points)

On considère qu'un objet est caractérisé par un poids et une utilité. Un objet peut donc être décrit par une liste de deux éléments [Poids,Utilité].

Questions :

1. Ecrivez le prédicat *objet(L)* qui vérifie que la liste L est un objet. **(0,5 point)**
2. Ecrivez le prédicat *sac(L)* qui vérifie que la liste L est un sac. Un sac peut être vide ou contenir uniquement des objets. **(0,5 point)**
3. Ecrivez le prédicat *appObjet(O,L)* qui vérifie si l'objet O est présent dans le sac L valide, à n'importe quel niveau. **(1 point)**
4. Un sac valide est un sac tel que la somme des poids de ses objets ne dépasse pas un poids de contenance maximum. Un sac valide F=[L,M] peut être décrit par une liste de deux éléments [L,M], ou L est le contenu du sac et M le poids de contenance maximum.
   Ecrivez le prédicat *sacValide(L)* qui vérifie que L est un sac valide. **(2 points)**