Université d'Orléans **Master Informatique**
Intelligence Artificielle Année 2010-2011

Contrôle continu

Durée : 2h. Documents autorisés : notes de cours et de TD.

Exercice 1 On s'intéresse à un agent détectant des pièces défectueuses dans une usine d'électronique. Les pièces passent sur un tapis roulant. Elles sont filmées par une caméra. L'agent exploite les images de cette caméra pour repérer des soudures défectueuses. Un aiguillage permet d'orienter les pièces défectueuses et les pièces correctes vers deux zones de stockage différents.
Les pièces satisfaisant le premier contrôle subissent ensuite des tests électroniques. A l'aide d'un bras robotisé, les cartes sont enfichées une par une sur un module de contrôle, qui teste une partie des fonctionnalités. Les pièces satisfaisant ce second test sont stockées pour conditionnement. Les autres sont stockées pour contrôle.
Un pourcent des pièces ayant satisfait tous les tests est prélevé, avant conditionnement, pour contrôle manuel plus complet. Il est ainsi possible de détecter des pièces défectueuses ayant satisfait tous les tests (faux positifs).
Un contrôle manuel des pièces défectueuses est effectué systématiquement afin d'établir la source du défaut, et éventuellement de repérer des pièces ayant été jugées défectueuses à tort (faux négatifs).
L'agent doit commettre le moins d'erreurs possibles, tout en respectant une cadence de contrôle donnée.

Donner la description de l'environnement PEAS de l'agent de détection des pièces défectueuses.

Exercice 2 On s'intéresse ici à un réseau social d'informaticiens, dans lequel on recherche (nom, prénom, email) un chef de projet (fonction) spécialisé en calcul de complexité (compétence). On suppose que le graphe est connexe. Le nombre d'amis d'un membre est limité à 100.

1. Modéliser ce problème sous forme d'une exploration d'arbre : quels informations sont stockées dans chaque noeud / état ? Quelles est (sont) la (les) fonction(s) successeur(s) ? La configuration de départ ? Le but ?
2. Quel est le facteur de branchement ? Quelle est la profondeur maximum de l'arbre ?
3. Lors d'une descente dans l'arbre, comment éviter de repasser par un noeud déjà visité ? Que devient la profondeur maximum avec cette précaution ?
4. Quelles sont les complexités (temps et espace) au pire des approches en profondeur d'abord et en largeur d'abord pour les configurations particulières suivantes : a/ Le graphe est une clique ; b/ le graphe est une chaîne.
5. Suggérez quelques propriétés potentielles (et pas trop irréalistes) de ce réseau social qui permettraient d'introduire des stratégies d'exploration.

Exercice 3 A l'état initial on a des lettres de A à Z qui sont classées dans un ordre aléatoire. On souhaite arriver à l'état final où les lettres sont ordonnées dans l'ordre alphabétique. On peut permuter deux lettres adjacentes ou séparées par une seule lettre. On modélise alors un état par une liste de lettres.

1. Dans l'objectif d'utiliser le moteur de recherche en profondeur qu'on a implanté, écrivez les prédicats nécessaire pour présenter les actions possibles. On se rappelle que la spécification du prédicat action est
   `action(+EtatCourant, -EtatSuivant, -NomAction)`.
2. Dans l'objectif d'utiliser le moteur de recherche A* qu'on a implanté, proposez une heuristique consistante et écrivez des prédicats nécessaires pour l'implanter. Rappelons que le prédicat h a la spécification `h(+Etat, -ValeurH)`.

Exercice 4 On considère un algorithme de recherche généralisée de la recherche locale comme suit :

- On commence avec k états initiaux, qui sont générés aléatoirement.
- A chaque étape on choisit parmi *tous* les voisins des k états courants k meilleurs voisins. Ces voisins deviennent les états courants pour l'étape suivante.
- Au bout de n étapes, on s'arrête et on choisit parmi les k états courants le meilleur état. Cet état est la solution du problème.

On dispose des prédicats suivants :

- `etat_initial(-E)` : qui calcule un état initial aléatoire E,
- `f(+E,-F)` : qui calcule la valeur F de la fonction à optimiser à l'état E,
- `suivant(+E1,-E2)` : qui calcule un état voisin $E2$ de l'état $E1$.

On cherche à minimiser la valeur F. Écrivez les prédicats nécessaires pour implanter l'algorithme. Écrivez un prédicat `go(+K,+N)` qui permet de lancer l'algorithme et afficher la solution.