**Université d'Orléans** **Master Informatique**
Intelligence Artificielle Année 2011-2012

# Contrôle continu

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

**Exercice 1** On souhaite modéliser un agent logiciel de microblogging. Cet agent est connecté à un site de microbloggage (par exemple, Twitter). Il a son propre compte de publication de messages, et est aussi abonné à différents auteurs. En fonction des messages lus dans ces abonnements, il décide de publier des messages.

L'agent cherche, dans les messages lus, des "signes d'intérêt" : présence de mots clés prédéfinis, mots devenant très fréquents dans les messages d'un auteur ou de plusieurs auteurs, variations dans la fréquence de publication de messages par certains auteurs, etc.

En fonction de ces signes d'intérêt, il publie des messages, dont la nature dépend du signe d'intérêt. Par exemple, ce peut être "Ce que dit tel auteur est intéressant", ou encore "Tel thème semble à la mode".

Dans un premier temps, l'agent avait été programmé pour surveiller le nombre d'abonnés à ses messages, ce qui était censé refléter la pertinence des messages publiés. Toutefois, le nombre d'abonnés présente une forte inertie (les gens ne se désabonnent pas tout de suite, même si les messages ne les intéressent pas). Le créateur de l'agent a donc imaginé un second moyen, complémentaire au premier, d'indiquer à l'agent s'il est pertinent dans ses messages. Pour rester dans le cadre du microblogging, le créateur a ouvert un compte, auquel l'agent est abonné. Sur ce compte, le créateur publie des messages indiquant la pertinence des publications de l'agent (les messages du créateur contiennent un identifiant de message de l'agent suivi de "ok" ou "ko").

Donner la description de l'environnement PEAS de cet agent.

**Exercice 2** On s'intéresse ici à un programme qui fait du "morphing" entre deux images. Ce programme dispose d'une banque d'images. On choisit dans cette banque une image de départ et une image d'arrivée. Le programme choisit alors une suite d'images de transition permettant d'aller agréablement de l'image de départ à l'image d'arrivée. On souhaite que l'enchainement forme un équilibre entre la douceur des transitions et le nombre total d'images. On va résoudre ce problème par l'exploration d'un arbre.

1. Quelles informations sont stockées dans chaque noeud / état ? Quelles est la fonction successeur ? La configuration de départ ? Le but ?
2. L'évaluation de la distance entre deux images est un élement important du coût de transition. Proposez une mesure de distance entre images. *On supposera que les images sont en niveau de gris, et qu'elles sont toutes de taille identique $c * c$.*
3. Comment prendre en compte, dans le coût d'une solution, les deux aspects "douceur des transitions" et "nombre total d'images" ?

4. Pour cette exploration, il existe une précaution évidente à prendre, concernant les successeurs potentiels ; laquelle ? Que se passe-t-il si on ne la prend pas ?
5. Quel est le facteur de branchement ? Quelle est la profondeur maximum de l'arbre ? *On supposera que la banque contient $N$ images.*
6. Quelle stratégie d'exploration aveugle suggérez-vous ? Vos mesures de distance et de coût ont-elles un impact sur la stratégie choisie ?
7. On envisage une exploration informée de type A*. Proposez une fonction heuristique.

**Exercice 3** Implantation de l'algorithme de recherche en coût uniforme.

Dans l'algorithme de recherche en largeur d'abord, à partir d'un nœud, on explore d'abord les nœuds à distance d'une action avant d'explorer des nœuds à une distance plus grande (rappelons que la liste d'attente des nœuds à explorer est organisée sous forme d'une file). Le coût d'un nœud peut être vu comme le nombre d'actions menant au nœud à partir du nœud initial.

On s'intéresse maintenant au cas où les actions ont chacune un coût, et le coût d'une action peut être différent de celui d'une autre action. Le coût d'un nœud $N$ est donc le coût *cumulé* des coûts des actions menant du nœud initial au nœud $N$. L'algorithme de recherche en coût uniforme est comme suit : soit $N$ le nœud courant (qui est le nœud initial lors de l'initialisation de l'algorithme)

- si $N$ est un nœud final alors arrêter,
- sinon calculer les nœuds successeurs de $N$ et les ajouter dans la liste d'attente tout en maintenant l'ordre croissant des coûts, prendre le nœud de coût minimal de la liste d'attente comme le nœud courant de la prochaine étape (c'est donc la tête de la liste).

A la fin on souhaite afficher le chemin menant du nœud initial au nœud final. Dans le cas où les actions ont toutes le même coût, cet algorithme devient l'algorithme de recherche en largeur d'abord.

On dispose des prédicats suivants

- `action(+EtatCourant, -EtatSuivant, -NomAction, -CoutAction)`,
- `noeud_initial(Noeud)`, `noeud_final(Noeud)`, ici un `Noeud` est représenté par un terme `node(Etat, Action, CoutNoeud)`, où `Etat` est l'état correspondant, `Action` est l'action permettant d'arriver à cet état et `CoutNoeud` le coût cumulé pour arriver du nœud initial au `Noeud`.

1. Ecrivez un prédicat `noeud_suivant(+N, -NS)` qui calcule un nœud successeur $NS$ du nœud $N$.
2. En s'inspirant de l'implantation de l'algorithme de recherche en largeur d'abord vue en TD, pour pouvoir se rappeler le chemin à partir du nœud initial, on maintient pour la liste d'attente une liste de listes. La liste d'attente est donc sous forme $[L_1, \ldots, L_n]$, où chaque $L_i$ est une liste de nœuds, telle que le premier nœud est le nœud en attente et le reste est le chemin du nœud initial au nœud en attente dans l'ordre inversé. Ecrivez les prédicats nécessaires pour implanter l'algorithme de recherche en coût uniforme. Expliquer brièvement le rôle de chaque prédicat.