Travail N° 1 --20%--

- Compléter l'implantation de la classe **VacuumModel** du projet lab1 (utiliser le package fournie dans le dossier Devoir-IA.zip (2pts).
- Lancer les testes de performances pour les deux types d'agents et commenter les résultats obtenus (2pts).

Travail N° 2 --80%--

1. Description

L'objectif de ce laboratoire est de vous familiariser avec la recherche dans un espace d'états à l'aide de l'algorithme A*. Le monde dans lequel évoluera votre agent est exactement le même que pour le laboratoire précédent, à quelques détails près. Votre agent aura connaissance du monde qui l'entoure, et devra par conséquent trouver la solution optimale minimisant son nombre d'actions pour ramasser toutes les poussières sur le terrain. Seulement, votre agent ne sera pas seul à se promener sur le terrain, un agent salissant se déplacera aléatoirement sur le terrain et salira une case du terrain de temps en temps. Il se trouve être plus lent que votre agent car il ne pourra exécuter qu'une seule action pendant le temps où votre agent en réalisera 2.

2. Travail à réaliser

Pour la réalisation de ce laboratoire, votre agent verra le monde en entier contrairement au laboratoire précédent où il ne voyait que la case sur laquelle il se trouvait. Vous aurez encore à compléter le code de votre agent dans le fichier **AstarBasedVacuumAgent**.java, principalement en implémentant la fonction *chooseAction*. Vous pouvez consulter la classe `SearchAgent` pour voir comment cette fonction peut être implantée pour un agent de recherche. La fréquence à laquelle vous allez re-planifier la recherche A* dans votre agent aura un grand impacte sur les performances de l'agent.

L'algorithme de recherche A* est déjà implémenté pour vous dans le package lab2.core.search[1]. Le teste unitaire pour cette classe réalisé sur le problème 8-puzzle (voir support de cours) est donné dans le package test.lab2.Astar. Par contre vous devrez implémenter vous-même votre espace d'états ainsi que l'heuristique que vous utiliserez pour résoudre ce problème. Une ébauche des squelettes vous est fournie dans le package lab2.vacuum. Vous devez compléter les classes qui permettent de définir l'espace d'état de votre agent :

- **VacuumPlanningState** : 3 fonctions à compléter *forward(),turnLeft(), et turnRigth*.
- **VacuumGoalTest** : Une fonction à completer *isGoalState(Object state)*
- **VacuumFunctionFactory** : Deux fonctions à compléter *actions(...) et result(...)*
- **VacuumHeuristicFunction** : P une heuristique *h(Object state)*. C'est cette fonction qui ferra la différence entre vos codes.

Une partie de la note du laboratoire sera attribuée en fonction de votre résultat à la compétition entre vos différentes implémentations. Il est à noter que pour la compétition les actions de l'agent salissant seront prédéterminées de manière à être les mêmes pour tout le monde de même pour la position initiale de votre agent et de l'agent salissant. La performance de chacune des équipes sera déterminée en fonction du temps d'exécution et du nombre de nœuds visités lors de la recherche.

1 : Code source de aima-java Java implementation of algorithms from Norvig And Russell's "Artificial Intelligenc e - A Modern Approach 3rd Edition." http://code.google.com/p/aima-java/

3. Déroulement de la Compétition

Votre résultat à la compétition sera déterminé par le temps d'exécution moyen de votre agent pour résoudre une grille et le nombre de nœuds qui auront été visités lors de l'exécution de la recherche avec l'algorithme A*. Ce dernier point évalue à la fois l'efficacité de l'heuristique que vous avez choisi ainsi que la fréquence à laquelle vous re-planifiez votre exécution.
La note de la compétition sera donnée sur 16 pts :
- 8 points pour la note globale
- 4 points pour la note en fonction du temps d'exécution
- 4 points pour la note en fonction de l'efficacité de votre algorithme en termes de nombre de nœud visité, …

Votre pointage en fonction de votre position sera calculé selon le percentile auquel vous appartenez à raison d'un point par percentile (i.e. Pour 4 points et 20 équipes, les 5 premiers auront 4, les 5 suivants auront 3.5 et les 5 derniers auront 2.5)

Bien que la compétition soit importante, elle ne vaut que pour 20% de la note pour ne pas pénaliser les équipes qui auront réalisé des agents un peu moins performants mais tout aussi valides que ceux des autres.

4. Directives pour la remise

Le travail sera réalisé en équipe de deux ou trois personnes. Vous téléchargerez sur le site du cours le projet eclipse. Un programme pour évaluer votre programme vous est fourni dans le fichier CompetitionNotee.java. Vous devez mettre vos noms dans l'emplacement indiqué par "Tappez les noms des membres de votre équipe pour remplacer cette phrase : ".

Vous remettrez un fichier .jar exécutable (qui lance la fonction main de la classe CompetitionNotee) et un document contenant une brève explication de la façon dont vous avez implémenté votre espace d'états et l'heuristique que vous avez choisi pour la résolution du problème. Tout devra être remis avant le 10 novembre à 23h. Tout travail en retard sera pénalisé d'une valeur de 10% pour chaque jour de retard. Le barème pour l'évaluation est le suivant :

Si vous avez des questions n'hésitez pas de me contacter.

Bonne chance.