INF4215 - Travail pratique #1.2
Hiver 2014

1 Description

Dans la deuxième partie du premier mini-projet, vous aurez à implémenter deux nouveaux modèles d'agents intelligents, et ce, dans sensiblement la même situation que lors du travail pratique précédent. Pour ce travail, le premier agent à implémenter utilisera l'algorithme A* et le deuxième une recherche locale quelconque afin de trouver une solution dans l'espace d'états.

Un robot travaillant dans un entrepôt a comme tâche de déplacer des boîtes avec chacune une destination vers une autre section. La figure 1 présente un exemple d'entrepôt avec 6 sections (A, B, E, F, G et I). Pour ce travail, il est possible d'ignorer les intersections lors des déplacements.

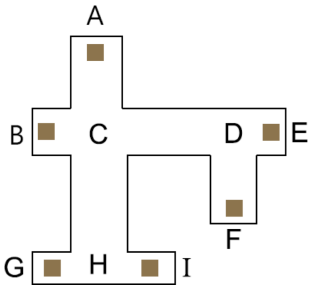

FIGURE 1 – Exemple de problème

Chaque section peut contenir plusieurs boîtes à être transférées et ces dernières possèdent toutes un poids. De plus, le robot possède une contrainte de poids, c'est pourquoi il est possible qu'il ne puisse recueillir tous les paquets présents dans une section en une seule visite. Ainsi, le robot devra revenir au comptoir ultérieurement afin d'accomplir sa tâche. L'énergie joue un rôle plus important dans ce travail puisque c'est cette dernière qui devra être minimisée pour l'algorithme A*.

2 Représentation du problème en Python

Une partie de la représentation du problème vous est fournie. Voici une brève description de chaque élément :

Agent : Représente votre agent. La classe possède les implémentations des actions que le robot peut effectuer dans l'environnement telles que *take, drop*, etc. En particulier, vous y trouverez la méthode *chooseAction*, qui détermine l'action choisie selon l'interprétation faite de l'environnement, qui lui est passée en paramètre. Ici, si on n'a jamais exécuté l'alrgorithme de recherche, on l'exécutera afin d'obtenir un plan. Ensuite, chaque fois que cette méthode sera appelée, on extraira la prochaine action du plan. Une implémentation partielle de cette classe vous est fournie.

Environnement : Cette classe est la scène où le robot se promène afin d'accomplir sa tâche et contient les méthodes pour manipuler l'environnement. C'est donc dans cette classe que l'agent sera appelé à faire ses actions. Une implémentation complète de cette classe vous est fournie.

Graph : Cette classe contient les nœuds et les arcs d'un graphe afin de représenter les comptoirs et les couloirs de l'entrepôt. La classe possède des méthodes pour obtenir le chemin entre deux comptoirs et sa distance. Une implémentation complète de cette classe vous est fournie.

Package : Représente les boîtes à déplacer d'un comptoir à un autre. Un paquet contient principalement une liste d'attributs spécifiant son identificateur unique, son poids et sa destination. Une implémentation complète de cette classe vous est fournie.

Search : Cette classe implémente l'algorithme A*. Une implémentation complète de cette classe vous est fournie.

Node : Cette classe représente chaque noeud de la recherche. Rappelez-vous que chaque noeud doit contenir, entre autres, l'état qu'il représente, la dernière action qui nous a mené à cet état, ainsi que le coût pour se rendre à cet état. Vous devrez implémenter cette classe.

3 Travail à réaliser

Dans ce travail pratique, vous devrez implémenter deux nouveaux agents qui permettront de résoudre le problème du robot. Dans un premier temps, vous devrez implémenter l'espace d'état (classe Node) et l'heuristique utilisée par l'algorithme A*. Remarquez ici que lors de la création d'un nouveau noeud dans la recherche, on utilisera une copie de tout l'environnement pour représenter l'état.

Par la suite, vous aurez à implémenter un agent utilisant une recherche locale. Plusieurs algorithmes ont été présentés en classe, par exemple l'exploration par escalade, le recuit simulé et les algorithmes génétiques. Bref, votre tâche est d'implémenter un agent utilisant l'un des algorithmes de recherche locale vus en cours.

Veuillez noter que dans cet environnement, le coût d'une action est l'énergie dépensée pour exécuter cette action. Ce coût dépend d'une constante k qui est fixée à 1 dans le code fourni (cette valeur pourrait être modifiée). Soit d la distance parcourue par l'agent, p_s et p_c son poids et le poids total de sa charge (c'est-à-dire le poids total des paquets qu'il transporte), l'énergie dépensée est égale à $k \times d \times (p_s + p_c)$. Pour prendre un paquet dont le poids est p_p, l'énergie dépensée est tout simplement $k \times p_p$.

Dans ce travail, vous devrez :

1. Compléter la méthode *chooseAction* de la classe Agent.
2. Implémenter les méthodes *expand*, *estimateHeuristic* de la classe Node.
3. Implémenter un agent utilisant **une recherche locale**. Pour ce cas, il faut lancer plusieurs recherches et parmi ces dernières, il faut tout simplement exécuter le plan

utilisant le moins d'énergies. La représentation du problème contenant n comptoirs est la suivante :

$$(C_1, C_2, C_3, \ldots, C_n, Charge, Position)$$

où C_i représente l'ensemble de paquets se trouvant au comptoir i, $Charge$ et $Position$ représentent les charge et position actuelles du robot.

4 Déroulement de la compétition

Il y aura également une compétition entre les équipes afin de déterminer la meilleure implémentation de l'agent A*. Cette compétition sera basée sur **l'énergie moyenne utilisée** pour résoudre le problème.

Comme pour le travail pratique précédent, la ou les meilleures équipes auront la note maximale de 15. Par la suite, les autres équipes seront divisées en groupe égal (le plus possible) et auront une note relative à leur position. Par exemple, pour 26 groupes et deux équipes qui se sont démarquées :

15/15	2 équipes
12/15	6 équipes
9/15	6 équipes
6/15	6 équipes
3/15	6 équipes

5 Questions

Question 1 : Indépendamment de vos résultats, lequel des deux agents est le plus approprié pour la situation ?

6 Directives pour la remise

Le travail sera réalisé en équipe de deux personnes. Vous téléchargerez sur le site du cours un fichier unique contenant les classes mentionnées dans la deuxième section. Vous pouvez modifier les classes présentes dans le fichier *AStar.py* selon vos besoins. De plus, implémentez vos agents dans les fichiers *Agent.py* (déjà partiellement implémenté) et *LocalSearchAgent.py*. N'oubliez pas de commenter votre code.

Remettez tous vos fichiers pythons dans un fichier compressé (zip, rar, etc.) où ce dernier est nommé comme suit : *matricule1_matricule2.zip*. Vous devez également remettre un fichier (pdf, doc, docx, etc.) contenant une explication de votre choix d'heuristique ainsi que la réponse à la question.

Tout devra être remis avant **le 16 février à 23h55**. Tout travail en retard sera pénalisé d'une valeur de 20% pour chaque jour de retard. Le barème pour l'évaluation est le suivant :

Implémentation de l'espace d'états	30 %
Choix de l'heuristique	20 %
Implémentation de l'agent utilisant une recherche locale	25 %
Résultat de la compétition	15 %
Questions	10 %

7 Bibliothèque

Pour les étudiants qui travaillent sur leur ordinateur personnel, il est nécessaire d'installer la bibliothèque *python-graph-core* afin de pouvoir exécuter le code donné. Pour ce faire, si vous avez *easy-install*, la commande est tout simplement :

```
easy-install python-graph-core
```

Autrement, il est également possible de télécharger la bibliothèque à l'adresse suivante : `https://code.google.com/p/python-graph/` et exécuter la commande ci-dessous à l'intérieur du dossier *core*.

```
python setup.py install
```