INF4215 - Travail pratique #1
Hiver 2016

1 Description

Dans le cadre de ce premier travail pratique, vous explorerez des méthodes basées sur la recherche dans un espace d'états afin de résoudre des problèmes de positionnement d'antennes de télécommunication.

Soit un ensemble de positions qui doivent être désservies par au moins une antenne de télécommunication. Chaque position est représentée par ses coordonnées cartésiennes (x, y). Chaque antenne a une portée spécifiée par son rayon r. Le coût associé à chaque antenne est $K + Cr^2$, où C et K sont des constantes. La figure 1 montre un exemple de 8 positions à desservir, et une solution (non optimale) comprenant trois anntennes.

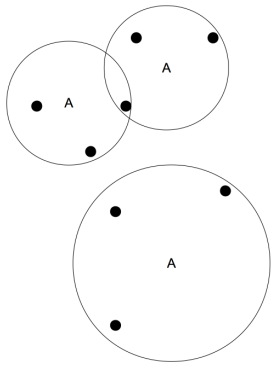

FIGURE 1 – Exemple

2 Travail à réaliser

Vous devrez développer deux programmes pour résoudre un problème de positionnement d'antennes tel que défini ci-haut. Le premier programme utilisera un algorithme de recherche arborescente, alors que le second utilisera un algorithme de recherche locale. Pour simplifier les calculs, vous supposerez que les coordonnées x et y des points à desservir et des antennes sont toujours des valeurs entières. De même, le rayon r de la portée d'une antenne sera lui aussi un nombre entier.

Chacun de vos deux programmes doit contenir une fonction `search(Positions,K,C)`. Le paramètre `Positions` est une liste de positions cartésiennes. Cette fonction effectue une recherche dans un espace d'états et retourne une liste de triplets (x, y, r), où x et y spécifient les coordonnées cartésiennes d'une antenne et r le rayon de la portée de cette antenne. Les paramètres `K` et `C` sont, évidemment, les constantes qui seront utilisées pour calculer le coût d'une antenne.
Prenons par exemple le problème illustré à la figure 2a. Si on a $K = 200$ et $C = 1$, on fera l'appel suivant :

```
search([(30,0),(10,10),(20,20),(30,40),(50,40)],200,1)
```

On devrait alors obtenir la liste suivante (qui correspond à la solution illustrée à la figure 2b) :

```
[(20.0, 10.0, 15.0), (40.0, 40.0, 10.0)]
```

(a) (b)

FIGURE 2 – Exemple

3 Compétition

Il y aura également une compétition entre les équipes afin de déterminer la meilleure implémentation. Cette compétition sera basée sur le temps nécessaire pour résoudre un problème et le coût de la solution retournée. Lors de votre remise, vous spécifierez laquelle de vos deux implémentations devra être utilisée pour la compétition. Plusieurs problèmes seront soumis pour l'évaluation et la note sera la moyenne de notes obtenues pour chaque problème. Pour l'évaluation d'un problème, les équipes seront divisées en cinq groupes égaux (le plus possible) et auront une note relative à leur position. Par exemple, pour 26 groupes :

10/10	6 équipes
8/10	5 équipes
6/10	5 équipes
4/10	5 équipes
2/10	5 équipes

4 Questions

Question 1 : Soit le code suivant :

```
def fct(predList, inputList):
    return filter(lambda x: all([f(x) for f in predList]), inputList)
```

Expliquez ce que fait cette fonction et fournissez un exemple utilisant cette fonction.

Question 2 : Quelles sont les points forts et les faiblesses de vos implémentations ?

5 Directives pour la remise

Le travail sera réalisé en équipe de deux personnes. Vous remettrez deux fichiers .zip (un pour chacune de vos implémentations), chacun contenant tous les fichiers Python nécessaires à l'exécution du programme. Les deux fichiers de votre remise seront nommés comme suit : *Code1_matricule1_matricule2.zip* et *Code2_matricule1_matricule2.zip*. Vous devez également remettre un fichier pdf contenant une explication de vos implémentations ainsi que les réponses aux deux questions.

Tout devra être remis avant **le 14 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 :

Première méthode implémentée	40%
Deuxième méthode implémentée	40%
Résultat de la compétition	10%
Rapport	10%