haute école neuchâtel berne jura arc ingénierie www.he-arc.ch

Semestre automne 2021

tags: `P3 - Paraglider Landing Training`

TP A*

Auteur :
@simon.meier

Table des matières :

Le but de ce TP va être d'utiliser l'algorithme A* pour trouver des chemins optimaux entre ces villes.

Questions

1. Heuristiques

Supposons que l'on veuille se rendre à la ville B. Pour tout noeud n, on va s'intéresser aux heuristiques suivantes:

- h0(n) = 0
- h1(n) = "la distance entre n et B sur l'axe des x"
- h2(n) = "la distance entre n et B sur l'axe des y"
- h3(n) = "la distance à vol d'oiseau entre n et B"
- h4(n) = "la distance de Manhattan entre n et B"

Parmi ces heuristiques, lesquelles sont admissibles ?

Réponse 1

Les heuristiques admissibles sont :
`h0(n), h1(n), h2(n), h3(n)`
`h4(n)` n'est pas admissible car elle surestime le coût.

2. A*

Implémentez en python une fonction qui:

- prend en paramètre deux villes et une heuristique
- utilise l'algorithme A*
- retourne le chemin le plus court entre ces deux villes en indiquant combien de villes on étées "visitées" pour trouver ce chemin optimal.
- fournir une interface graphique ou en ligne de commande pour tester votre implémentation.
- implémentez également les 5 heuristiques.

3. Expérimentation

Cherchez quelques chemins optimaux à l'aide de votre programme et des différentes heuristiques.

- a) l'utilisation des différentes heuristiques a-t-elle une influence sur l'*efficacité* de la recherche ? (en termes du nombres de noeuds visités)
- b) pouvez-vous trouver des exemples où l'utilisation de différentes heuristiques donne des résultats différents en termes de chemin trouvé ?
- c) dans un cas réel, quelle heuristique utiliseriez-vous ? pourquoi ?
  - aller plus loin : chercher la définition d'heuristique "consistente" ou "monotone".
  - si vous assurez à votre algorithme une heuristique monotone, comment pourriez-vous améliorer votre code ?
  - parmi les 5 heuristiques ci-dessus, est-ce qu'il y en a des *monotones* ?
    Si **non**, proposez une heuristique monotone pour notre problème du voyageur (pas besoin de l'implémenter)

Réponses 2

a) Oui. Le nombre de villes visitées varie en fonction de l' `heuristique` utilisée.

Heuristique	Explications
Si h(n) = 0 :	il n'y a que g(n) qui influence.
Si h(n) est <= :	A* trouve le chemin le plus court. Plus h(n) est petite, plus A* est lent
Si h(n) est == :	A* trouve le meilleur chemin.
Si h(n) est >= :	A* est rapide, mais donne pas forcément le chemin le plus court.
Si h(n) est >> :	g(n) n'influence presque plus le comportement, A* trouve un chemin plus vite, mais donne pas forcément le chemin le plus court.

b) Avec la distance de Manhattan, deux cas de figure ressortent:

- *Paris* à *Prague*
- *Brussels* à *Prague*

Résultats différents

```
Astar result :
>—Origin : Paris  ▷—»—→ To >—»—◇  Destination : Prague;

[Visited cities / nodes] before finding the shortest path :
{'Brussels', 'Hamburg', 'Amsterdam', 'Prague', 'Berlin', 'Bern', 'Munich', 'Genoa', 'Paris'}
[Number of visited nodes] : 9
[distance]: 1089 [km]
Heuristic :        >—< [as crow fly] >—<
```

```
Astar result :
>—Origin : Paris  ▷—»—→ To >—»—◇  Destination : Prague;

[Visited cities / nodes] before finding the shortest path :
{'Brussels', 'Hamburg', 'Amsterdam', 'Prague', 'Berlin', 'Paris'}
[Number of visited nodes] : 6
[distance]: 1128 [km]
Heuristic utilisée : >—< [manhattan distance] >—<
```

```
Astar result :
>—Origin : Brussels  ▷—»—→ To >—»—◇  Destination : Prague;

[Visited cities / nodes] before finding the shortest path :
{'Brussels', 'Hamburg', 'Amsterdam', 'Prague', 'Berlin', 'Bern', 'Munich', 'Paris'}
[Number of visited nodes] : 8
[distance]: 864 [km]
Heuristic :        >—< [as crow fly] >—<
```

```
Astar result :
>—Origin : Brussels  ▷—»—→ To >—»—◇  Destination : Prague;

[Visited cities / nodes] before finding the shortest path :
{'Brussels', 'Hamburg', 'Amsterdam', 'Prague', 'Berlin', 'Paris'}
[Number of visited nodes] : 6
[distance]: 903 [km]
Heuristic utilisée : >—< [manhattan distance] >—<
```

c) La distance à “vol d’oiseau”, soit `crow-fly` , semble être la meilleure dans le cas présent. Elle ne surestime jamais et offre un bon compromis entre “pas assez” et suffisamment. C’est donc celle que j’utiliserais.

- i) “*Une fonction heuristique est dite consistente ou monotone, si son estimation est toujours inférieure ou égale à la distanciation estimée e de tout sommet voisin de l’objectif,* ***plus le coût d’atteindre ce voisin***.” source (https://datafranca.org/wiki/Fonction_consistente)
- ii) Si j’avais fait une classe *City*, j’aurais pu enlever history et utiliser l’attribut `parent` . J’aurais d’ailleurs dû faire ça en fait.
- iii) h0, h3; je ne mettrais pas h4 dans la liste puisqu’elle n’est pas admissible.