Travail 1 : Recherche dans un espace d'états / Rétablir l'électricité

Ancelin SERRE	SERA23029704
Baptiste BOUVIER	BOUB01099701

Utilisation du programme

Comme spécifié dans les consignes du TP, le programme est écrit en Python3 et affiche par défaut le chemin emprunté par l'algorithme pour résoudre le problème de coupure d'électricité sur une carte donnée.

En plus de cela, il est possible d'afficher les métriques $M1$ et $M2$ via l'utilisation des arguments `-m1` et `-m2` respectivement.

Il est enfin possible d'afficher le temps d'exécution du programme en utilisant l'argument `-t`.

Heuristique 1 : Réduire le nombre de maisons non alimentées

Fonctionnement

Avec cette heuristique, on cherche à aider l'algorithme à faire un choix entre deux états se trouvant à une distance égale. On souhaite que l'état choisi par l'agent intelligent soit celui qui minimise le nombre de maisons non alimentées. On a présente ainsi la situation suivante :

Figure 1 - Carte utilisée pour illustrer l'heuristique 1

On souhaite que l'équipe de réparation allume l'interrupteur situé au sud en premier car ce dernier est responsable de plus de maison que l'interrupteur au nord.

Exemple sans utilisation de l'heuristique :

```
TP_IA> python .\TP1.py .\tests\grille19.txt -m1 -m2
N 1 S S 1
Runtime = 0:00:00.000997
--------------------------
M1: Total actions = 5
--------------------------
House (2, 10) : 2 TU
House (4, 12) : 5 TU
House (5, 6) : 5 TU
House (5, 8) : 5 TU
House (5, 10) : 5 TU
--------------------------
M2: Total outage time = 22 Time Unit
```

Figure 2 - Trace d'exécution du programme utilisant la carte présente en Fig. 1 et sans l'utilisation de l'heuristique 1

Ici, on voit que par défaut, l'équipe d'entretien va aller allumer l'interrupteur au Nord en premier alors que celui-ci est responsable de seulement une maison ce qui n'est pas optimal.

Exemple avec utilisation de l'heuristique :

```
TP_IA> python .\TP1.py .\tests\grille19.txt -m1 -m2
S 1 N N 1
Runtime = 0:00:00.000997
--------------------------
M1: Total actions = 5
--------------------------
House (4, 12) : 2 TU
House (5, 6) : 2 TU
House (5, 8) : 2 TU
House (5, 10) : 2 TU
House (2, 10) : 5 TU
--------------------------
M2: Total outage time = 13 Time Unit
```

Figure 3 - Trace d'exécution du programme utilisant la carte présente en Fig.1 et avec l'utilisation de l'heuristique 1

Avec l'utilisation de l'heuristique visant à réduire le nombre de maisons non alimentées, on obtient une meilleure métrique $M2$ puisque l'équipe d'entretien a cette fois favorisée la remise en service du réseau au sud de sa position car il est responsable de plus de maisons (4 contre 1).

Admissibilité

Notre heuristique est admissible car nous avons fait en sorte que sa valeur soit toujours comprise entre 0 et 1. Pour cela, nous calculons l'heuristique de la manière suivante :

$$Heuristique = 1 - \frac{1}{nb\ maisons\ non\ alimentées\ après\ réalisation\ de\ l'état}$$

$$OU$$

$$Heuristique = 0\ ssi\ toutes\ les\ maisons\ sont\ alimentées$$

De plus, on sait que notre valeur de G est au minimum de 1.
On s'assure donc que $H < G$ dans tous les cas ce qui garantit que cette heuristique est admissible.

Heuristique 2 : Réparation en premier

Fonctionnement

Avec cette heuristique, l'objectif est proche de celui de l'heuristique 1. On favorise ici la remise en service d'une ligne totalement réparée plutôt que la réparation d'une autre ligne endommagée à distance égale. On présente ainsi la situation ci-dessous :

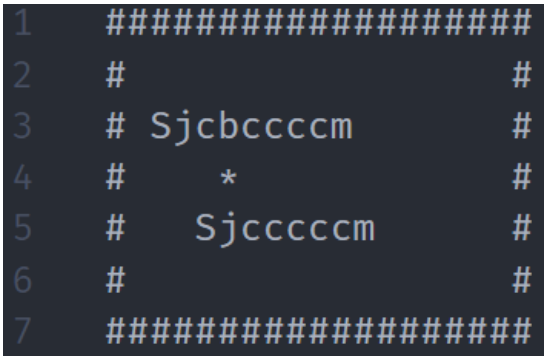

Figure 4 - Carte utilisée pour illustrer l'utilisation de l'heuristique 2

Dans ce contexte là, on devrait normalement aller rallumer l'interrupteur au sud de la position initiale car le réseau en question est déjà réparé.

Exemple sans utilisation de l'heuristique :

```
TP_IA> python .\TP1.py .\tests\grille20.txt -m1 -m2
N R W W 1 S E S E 1
Runtime = 0:00:00.000971
--------------------------
M1: Total actions = 10
--------------------------
House (2, 10) : 5 TU
House (4, 11) : 10 TU
--------------------------
M2: Total outage time = 15 Time Unit
TP_IA>
```

Figure 5 - Trace d'exécution du programme utilisant la carte présente en Fig. 4 et sans l'utilisation de l'heuristique 2

On voit que sans l'utilisation de l'heuristique 2, l'algorithme ne prend pas le chemin optimal et va au nord en premier ce qui n'est pas le résultat souhaité.

Exemple avec utiilisation de l'heuristique :

```
TP_IA> python .\TP1.py .\tests\grille20.txt -m1 -m2
S 1 N N R W W 1
Runtime = 0:00:00.000997
--------------------------
M1: Total actions = 8
--------------------------
House (4, 11) : 2 TU
House (2, 10) : 8 TU
--------------------------
M2: Total outage time = 10 Time Unit
TP_IA>
```

Figure 6 – Trace d'exécution du programme utilisant la carte présente en Fig. 4 et avec l'utilisation de l'heuristique 2

Avec utilisation de l'heuristique 2, on obtient maintenant le chemin optimal. Comme on peut le voir, l'équipe de réparation démarre son opération par le rallumage du réseau au sud ce qui était attendu.

On constate également une amélioration de la métrique $M1$ et de la métrique $M2$ avec respectivement une baisse du nombre d'actions effectuées de 10 à 8 et une baisse de la durée de coupure d'électricité pour les maisons passant de 15 à 10 unités de temps.

Admissibilité

Comme l'heuristique précédente, l'heuristique 2 est toujours comprise entre 0 et 1 via le même procédé. On la calcule de la manière suivante :

$Si\ l'état\ correspond\ à\ un\ interrupteur\ alors$:

$$Heuristique = 1 - \frac{1}{nb\ bris\ restants\ après\ réalisation\ de\ l'état}$$

$Si\ l'état\ correspond\ à\ un\ bris\ alors$:

$$Heuristique = 1 - \frac{1}{(nb\ bris\ restants\ après\ réalisation\ de\ l'état + 1)}$$

$$OU$$

$$Heuristique = \ 0\ ssi\ il\ n'y\ a\ plus\ de\ bris\ à\ réparer$$

De plus, on sait que notre valeur de G est au minimum de 1.
On s'assure donc que $H < G$ dans tous les cas ce qui garantit que cette heuristique est admissible.

Comparaison entre les deux heuristiques

Pour réaliser nos tests, nous utiliserons la grille suivante :

Figure 7 - Carte utilisée pour la comparaison des heuristiques

	Nombre d'actions	Temps CPU (sec)	Métrique M2
Heuristique 1	264	≈0.477	2676
Heuristique 2	264	≈0.453	2676

Sur cet exemple de taille conséquente, on peut noter tout d'abord que le temps d'exécution est légèrement plus faible avec l'utilisation de l'heuristique 2. Ensuite, on n'observe pas de distinctions entre les deux heuristiques en ce qui concerne le nombre d'actions effectuées et la métrique $M2$.

Cela s'explique par le fait que notre programme effectue de base (hors heuristique) un tri assez conséquent des prochains états possibles pour l'équipe de réparation.

En effet, notre programme fait en sorte que :

- Un bris n'est accessible que s'il n'est pas alimenté en électricité.
- Un interrupteur est accessible s'il est allumé et s'il dispose de bris à réparer sur son réseau.
- Un interrupteur est accessible s'il est éteint et que son réseau est réparé et prêt à être rétabli.

Cependant, notre programme seul ne suffit pas à couvrir de manière optimale des cas particuliers comme vu précédemment lors de la présentation des heuristiques 1 et 2.