Gondouin Fanny
Souvigny Camille

Rapport phase 2

Les fonctions

Phase 1

defi_voc(n=4) :
La fonction définit le vocabulaire du Wumpus. On prend en compte quatre variables par cases : W, P, S, B. On n'inclut pas l'or en tant que variable.

liste_voisins(case, n=4):
Pour chaque case, on renvoie la liste des voisins.

definitionregles(gs, nn=4):
On envoie les règles du jeu dans les deux sens : P -> B chez les voisins et B -> P chez au moins 1 voisins (pour envoyer cette clause on crée un tableau temporaire qui stocke les voisins bien envoyés un OU et non pas un ET). Les mêmes règles s'appliquent à W et S.
On ajoute une règle indiquant qu'un seul Wumpus est présent sur la carte.

ajout_clauses(gs,size):
Cette fonction permet d'ajouter à notre base de connaissance les clauses obtenues avec la fonction précédente.

ajout_de_deduction_clauses(gs,b,case):
Cette fonction permet d'ajouter à notre base de connaissance les clauses déduites suite aux actions PROBE ou CAUTIOUS PROBE.

def cartographier(nn=4):
Fonctionnement de l'algorithme :

- Tant que on ne connait pas toutes les cases :
  - On parcourt toutes les cases et PROBE toutes celles qui sont sures (P ou W renvoie faux). A chaque probe on ajoute le résultats dans les clauses (si B alors on ajoute B et -S,-W,-P)
  - Si lors de l'itération précédente on n'a pas trouvé une seule case sure : on cautious probe la première case non cartographiée.
- On cautious probe une case qu'on ne connait pas

On ajoute les clauses trouvées à notre base de connaissance.
On retourne notre carte

Phase 2

etat_finaux(carte):
Cette fonction détermine les cases contenant de l'or, et les renvoie dans un tableau

Gondouin Fanny
Souvigny Camille

liste_successeurs(case, liste, n=4):
Cette fonction renvoie la liste de toutes les cases atteignables depuis une case donnée.

trier_liste(l):
Cette fonction nous permet d'avoir la liste des états finaux dans l'ordre. Elle utilise la méthode sorted().

heuristique(etat,final):
Définition de notre heuristique : évalue de manière optimiste la longueur chemin qui reste à parcourir entre notre case état et l'état final selon le chemin de Manhattan.

but_multiple_heuristique_manhattan(etat,Final):
On récupère la distance à l'état final le plus proche de la case état envoyé en paramètre. On a fait ce choix car les notions vues en RO03 sur la recherche du plus court chemin s'apparentent grossièrement à choisir à chaque fois la case la plus proche. La fonction renvoie l'état final choisi et la distance de Manhattan correspondante.

recuperer_chemin(chemin,out,predecesseur,initial):
Cette fonction nous permet de récupérer le chemin allant de la case « initial » à la case « out ».

recherche_or(chemin,predecesseur,liste,initial,Final):
Cette fonction permet de déterminer le chemin permettant d'accéder aux cases contenant de l'or en utilisant le pathfinding A*.
La fonction est récursive et déroule A* a chaque appel récursif. A chaque état final trouvé un appel est effectué pour trouver le prochain état final. Avant l'appel suivant, la portion de chemin est reconstituée avec **récupérer_chemin()** et ajouté dans **chemin** qui est une variable globale.
A chaque nouvel appel, on établit la destination avec **but_multiple_heuristique_manhattan()** qu'on stocke dans la variable destination. Dans le déroulement de A*, **destination** est utilisé pour appeler **heuristique()** et éviter que l'algo ne change de destination.
Le chemin est retourné quand la liste d'état finaux est vide et donc que toutes les cases d'or ont été traitées. Si une case n'est pas accessible car enfermée par des puits/wumpus, A* renvoie none. Pour y remédier, l'état est enlevé de la liste à la fin de l'appel pour ne pas bloquer l'algorithme dans la recherche des prochaines cases.

course_vers_or(ww,chemin):
Cette fonction permet de mettre en pratique le chemin déterminé par la fonction précédente. Pour chaque cases (i,j) du chemin la fonction **go_to(i,j)** est appélée.

Phase 3

treat_map_1(wwr: WumpusWorldRemote) :
Pour la compétition d'IA, nous avons implémenté toutes les phases dans cette fonction.
On récupère la cartographie dans la variable carte pour l'utiliser dans la phase 2. On effectue la recherche de l'or d'initial (0,0) à l'ensemble des états finaux, puis la recherche de chemin pour retourner en case initial depuis le dernier.

Gondouin Fanny
Souvigny Camille

Exécution

Dans le .zip se trouve le rapport, l'exemple d'utilisation « IA_Souvigny_Gondouin » à compiler avec le serveur version 12.0-rc3 et le fichier python Phase2__Souvigny_Gondouin qui comporte les algorithmes du TP de Path Finding.

Il suffit de compiler F5 pour exécuter les différents programmes.