Exercice 1 :

* pick-up :

Opérateur permettant de ramasser un cube libre sur la table

* put-down :

Opérateur permettant de déposer un cube en main sur la table

* stack :

Opérateur permettant d’empiler un cube en main sur un autre libre

* unstack :

Opérateur permettant de ramasser un cube libre empilé sur un autre

La différence entre put-down et stack repose sur la destination du cube initialement en main : put-down le dépose sur la table et stack sur un autre cube (libre).

« holding ? » permet de désigner un cube directement manipulable. Sans ce fluent les actions perdent l’intermédiaire de la main (le fluent « handempty » ne sert donc plus). Il ne reste que les actions stack et unstack qui deviennent alors :

(:action stack

:parameters (?x ?y)

:precondition (and (clear ?x) (clear ?y))

:effect

(and (not (clear ?y))

(on ?x ?y)))

(:action unstack

:parameters (?x ?y)

:precondition (and (on ?x ?y) (clear ?x))

:effect

(and (clear ?y)

(ontable ?x)

(not (on ?x ?y)))

Exercice 2 :

Le plan solution est de longueur 6. Il est trouvé en 0.41 secondes (temps total de l’algorithme).

Il n’y a eu qu’une seule itération (de longueur 6).

L’application du plan solution durant moins de 5e-03 secondes, les actions du plan solution sont trop rapides pour avoir accès à leur temps d’exécution (ne serait-ce que leur temps moyen).

On peut imaginer des plans-solutions plus longs à rallonge (il en existe une infinité) qui s’appuient sur des cycles pick-up/put-down par exemple. Ils ne sont pas fournis par ce planificateur car il s’arrête au premier qu’il trouve.

Exercice 3 :

Le plan solution est de longueur 10. Il est trouvé en 0.05 secondes. Il n’y a eu qu’une seule itération.

Exercice 4 :

Le plan solution est de longueur 32. Il est trouvé en 1.30 secondes. Il y a eu 7 itérations.

Exercice 5 :

A notre avis, cette méthode de planification est assez efficace dans le sens où la solution trouvée sera toujours de longueur minimale. La contrepartie de cette optimalité est que l’algorithme tournera potentiellement longtemps si la longueur minimale est plus grande que l’estimation a priori (pour de grands graphes notamment).

Exercice 6 :

Nous avons utilisé une plateforme en ligne pour éditer les fichiers .pddl nécessaires à l’élaboration des réponses et le solveur CPT a brusquement cessé de lire correctement ces fichiers en renvoyant le message d’erreur suivant :

D:\Scolarite\ENSTA\Cours\3A\rob316\ROB316\_TP5>cpt.exe -f singe-bananes01.pddl -o singe.pddl problem file : singe-bananes01.pddl domain file : singe.pddl Parsing domain.................... done : 0.00 Parsing problem................... done : 0.01 domain : singe problem : singeaffame Instantiating operators........... done : 0.00 Creating initial structures....... **Start or End has been removed...**

Nous ne pouvions donc pas poursuivre avec l’exploitation des données…

Exercice 7 :

Nous avons rencontré le même problème qu’à l’exercice précédent…

La complexité en nombre de coups pour l’algorithme donné dans l’énoncé est défini par la suite suivante :

C’est une suite arithmético-géométrique dont la résolution est triviale… Nous avons finalement une complexité en nombre de coups donnée par la suite suivante :