Report LINGI2261: Assignment 2

Group N°185

Student1: 27312000

Student2: 50351800

February 28, 2023

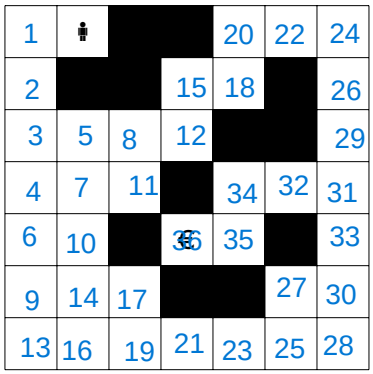
1 Search Algorithms and their relations (3 pts)

Consider the maze problems given on Figure 1. The goal is to find a path from 🚹 to € moving up, down, left or right. The black cells represent walls. This question must be answered by hand and doesn't require any programming.

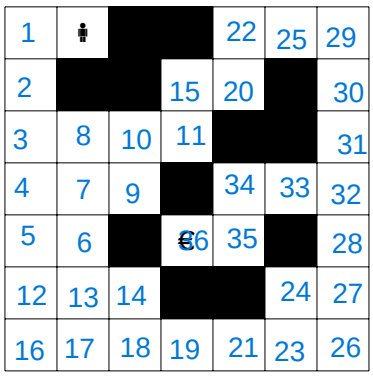
1. Give a consistent heuristic for this problem. Prove that it is consistent. Also prove that it is admissible. **(1 pt)**

Une heuristique consistante serait h(n) la somme des distances de Manhattan entre les coordonnées des extrémités des câbles jusqu' aux coordonnée de leurs sorties. Une heuristique est admissible si elle ne surestime jamais le coût pour atteindre le but. Ici c'est d'office le cas car on ne peut pas trouver de déplacement vers l'objectif plus court vu qu'on ignore la contrainte des murs. Cette heuristique est également consistante vu qu'il n'y a pas de chemin plus court que celui estimé par notre heuristique, il n'existe pas de chemin tel que le coût d'aller de n à n' par l'action a plus le coût estimé h(n') soit plus petit que h(n).

2. Show on the left maze the states (board positions) that are visited when performing a uniform-cost graph search, by writing the order numbers in the relevant cells. We assume that when different states in the fringe have the smallest value, the algorithm chooses the state with the smallest coordinate (i, j) ($(0, 0)$ being the bottom left position, i being the horizontal index and j the vertical one) using a lexicographical order. **(1 pt)**



3. Show on the right maze the board positions visited by A^* graph search with a manhattan distance heuristic (ignoring walls), by writing the order numbers in the relevant cells. A state is visited when it is selected in the fringe and expanded. When several states have the smallest path cost, they are visited in the same lexicographical order as the one used for uniform-cost graph search. **(1 pt)**



2 SoftFlow problem (17 pts)

1. Model the SoftFlow problem as a search problem; describe: **(2 pts)**
   - States
   - Initial state
   - Actions / Transition model
   - Goal test
   - Path cost function

Un état est une disposition de la grille et l'état initial est la disposition de la grille tel quel dans les instances, dans le programme cette grille est représenter par une liste. Les différentes actions possibles sont les déplacements d'une case de l'extrémité d'un câble non branché (représenter par les lettres) sur l'axe X ou Y si la case visé est libre (pas de mur # ni de câbles).
Chaque action crée une nouvelle disposition de la grille où l'extrémité s'est déplacé d'une case par rapport à l'état précédent et un chiffre remplace l'ancienne position de l'extrémité du câble (représentant le câble lui même).
Le goal test sera vrai si l'état respecte deux conditions: En 1, chaque câble du problème doit être relier à la sortie qui lui correspond et en 2, ce lien doit être le plus court possible.
Enfin, le coût d'un chemin est la somme des cases parcourus par les câbles, le déplacement d'une case à une autre est de 1.

2. Give an upper bound on the number of different states for a SoftFlow problem with a map of size $n \times m$, with k cables to place. Justify your answer precisely. **(1 pt)**

Une borne supérieur pourrait être:

$$\sum_{k=o/2}^{(m\ n)-o} \left(\frac{k!}{(k-o/2)!(o/2)!}\right) (4^{**}k)$$

Où (m n) est la dimension de la grille, k le nombre de cases câbles sur la grille et o le nombre de câbles/sorties présent initialement. En assumant qu'un câble puisse se déplacer dans 4 directions (k!/(k-o/2)!(o/2)!) (4**k) représenterait le nombre d'états possibles avec k case de câbles sur la grille multiplier par le nombre de combinaison de k case de câble avec o/2 type de câble différents.

3. Give an admissible heuristic for a SoftFlow instance with k cables. Prove that it is admissible. What is its complexity ? **(2 pts)**

Comme précédemment une heuristique admissible serait la distance de Manhattan entre les extrémités des câbles et leur sortie correspondante. Cette heuristique ne surestime jamais le coût du chemin puisqu'il n'existe pas de chemin plus court sur une grille que celui qui parcours la distance de Manhattan.
h(n) = abs((n.pos - n'.pos))
n est l'extrémité du câble et n' sa sortie, .pos renvoi la position du point sur la grille.
Cette fonction a une complexité en O(1)

5. **Implement** your solver. Extend the *Problem* class and implement the necessary methods and other class(es) if necessary. **(1 pt)**

6. **Experiment**, compare and analyze informed (*astar_search*) and uninformed (*breadth_first_graph_search*) graph search of aima-python3 on the 10 instances of SoftFlow provided. Report in a table the time, the number of explored nodes and the number of steps to reach the solution. Are the number of explored nodes always smaller with *astar_search*? What about the computation time? Why? When no solution can be found by a strategy in a reasonable time (say **3 min**), indicate the reason (time-out and/or exceeded the memory). **(4 pts for the whole question)**

On remarque que A* est toujours plus rapide que Best First Graph Search puisqu'on explore beaucoup moins de nodes. On s'en doutait puisque notre heuristique est admissible et que donc A* n'explore pas de nodes dont la somme du coût et de l'heuristique dépasse le coût optimale. Cependant les solutions trouvées restent optimales grâce à notre implémentation du problème, elles mette juste plus de temps à être trouvé. Pour l'instance 10 notre implémentation ne parvient pas à trouver de solution et renvoi none parce que notre recherche bloque lorsque les câbles s'entremêlent et BFS sur l'instance i08 nous donnes un Timeout à cause du nombre de nodes trop important à explorer.

	A* Graph			BFS Graph		
Inst.	NS	T(s)	EN	NS	T(s)	EN
i01	19	0.003	19	19	0.017	86
i02	23	0.052	166	23	34.375	7682
i03	19	0.014	37	19	38.481	6532
i04	19	0.018	53	19	1.134	1170
i05	25	0.019	64	25	0.062	248
i06	36	0.047	169	36	106.347	15497
i07	19	0.005	20	19	0.013	75
i08	48	1.484	1174	N/A	N/A	N/A
i09	20	0.080	189	20	3.249	2021
i10	N/A	N/A	N/A	N/A	N/A	N/A

NS: Number of steps — **T**: Time — **EN**: Explored nodes

6. **Submit** your program on INGInious, using the *A** algorithm with your best heuristic(s). Your file must be named *softflow.py*. You program must print to the standard output a solution to the SoftFlow instance given in argument, satisfying the described output format. **(5 pts)**

Fait.