

Report LINGI2261: Assignment 2

Group N°...

Student1:

Student2:

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 distance de Manhattan entre les coordonnées de n jusqu'aux coordonnées l'objectif. 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)

1	!				20	22	24
2				15	18		26
3	5	8	12				29
4	7	11			34	32	31
6	10			36	35		33
9	14	17				27	30
13	16	19	21	23	25	28	

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)

1	⚡			22	25	29
2			15	20		30
3	8	10	11			31
4	7	9		34	33	32
5	6		36	35		28
12	13	14			24	27
16	17	18	19	21	23	26

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 représentée par une liste. Les différentes actions possibles sont les déplacements d'une case de l'extrémité d'un câble non branché (représenté par les lettres) sur l'axe X ou Y si la case visée est libre (pas de mur # ni de câbles).

Chaque action crée une nouvelle disposition de la grille où l'extrémité s'est déplacée 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 relié 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 parcourues 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)

En ignorant les murs il pourrait y avoir $k \cdot ((m \cdot n) - (2k))$

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 parcourt la distance de Manhattan.

$h(n) = \text{abs}((n.\text{pos} - n'.\text{pos}))$

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 aim-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)

Inst.	A* Graph			BFS Graph		
	NS	T(s)	EN	NS	T(s)	EN
i01						
i02						
i03						
i04						
i05						
i06						
i07						
i08						
i09						
i10						

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

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