Report LINGI2261: Assignment 2

Group N°34

Student1: Diego Troch

Student2: David Lefebvre

March 16, 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)**

We want to prove that the heuristic is admissible, meaning that it never overestimates the actual cost of reaching the goal. We also want that it satisfies the triangle inequality, it would prove its consistency. A proper heuristic for this maze problem would be using the Manhattan distance. This heuristic is consistent since $h(n \rightarrow goal) \leqslant c(n, a, n\prime) + h(n\prime \rightarrow goal)$ and because we can only go one unit in four directions (up, right, down and left), the inequality is always respected. This is because h(n) assumes the shortest path that can go in any direction, which is ideal. We can therefore say that $h(n \rightarrow n\prime) \leqslant c(n, a, n\prime)$.

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	33	31
6	10	■	€	35	■	32
9	14	17	■	■	27	30
13	16	19	21	23	25	28

3. Show on the right maze the board positions visited by $A^{\star}$ 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	24	28
2	■	■	14	18	■	31
3	5	12	13	■	■	32
4	7	11	■	34	33	30
6	9	■	€	35	■	29
8	10	16	■	■	25	27
19	15	17	20	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

- State: Contains the number of rows and columns but also the start and end point of each cable.
- Initial state: The grid that we load at the beginning when we haven't moved yet.
- Actions: Move one unit in one of the four directions (up, right, left, down) and extend the cable until we find the endpoint.
- Goal test: If every cable is complete, stop the execution.
- Path cost function: The number of unit taken in order to reach the goal.

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)**

Let's first note that there are (n × m) possible locations where the cables can be placed. For the first cable, there are (n × m) possible entry points and (n × m) possible exit points, but since the entry and exit points cannot be the same, there are only (n × m) × (n × m - 1) possible combinations. For the k-th cable, there are (n × m - 2(k - 1)) possible entry and exit points and (k - (k - 1)) possible cables that it can connect to, so there are (n × m - 2(k - 1)) possible combinations. By simplifying using factorial notation, the total number of different states can be estimated as: (n × m)! / (n × m - k)!
This is an upper bound because it assumes that all combinations of cable placements are valid, which may not be the case due to the narrow spaces in the false ceilings.

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

The heuristic function h(node) used in the SoftFlow problem calculates the sum of the Manhattan distances between each cable's entry and exit points. The Manhattan distance between two points (startX1, startY1) and (endX2, endY2) is defined as abs(startX1 - endX2) + abs(startY1 - endY2). To prove that this heuristic is admissible, we need to show that it never overestimates the cost of reaching the goal state. That is, $h(node) <= cost\_to\_goal(node)$. The Manhattan distance heuristic is admissible because it always underestimates the actual distance between two points in a grid. In other words, the sum of the Manhattan distances between each cable's entry and exit points is always less than or equal to the actual cost of connecting the cables. Since we need to calculate the heuristic for each node in the search space, the total time complexity of the search algorithm would be $O(b^d * k)$, where b is the branching factor, and d is the depth of the shallowest goal node.

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)**

The BFS search algorithm, although it guarantees finding the shortest path, is not necessarily the fastest, especially if there are many dead ends and obstacles in a problem like this. This is due to the fact that it does not use heuristic functions that allow it to search for an optimal solution by starting in the right direction.
A* algorithm uses both the cost of reaching a node and a heuristic estimate of the distance to the goal to determine which path to explore next. This approach allows it to be much more efficient when there are a large number of obstacles.
As can be seen below, even for the simplest problem instances, bfs can take 10 to 100 times longer than an informed search with a star.

	A^* Graph			BFS Graph		
Inst.	NS	T(s)	EN	NS	T(s)	EN
i01	19	0.008	174	19	0.07588	/
i02	23	0.14	391	23	2.212	/
i03	19	0.006	77	19	0.219	/
i04	19	0.57	410	19	339.352	/
i05	25	0.01	25	/	TimeOut	/
i06	36	13.3	2465	/	TimeOut	/
i07	19	2.85	1292	/	TimeOut	/
i08	48	0.94	1432	/	TimeOut	/
i09	20	0.07	270	/	TimeOut	/
i10	24	0.16	581	/	TimeOut	/

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)**