

# Report LINGI2261: Assignment 2



Group N°29

Student1: Ben Haddou Mehdi

Student2: Hennebo Eliot

March 11, 2021

## 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)

Heuristic : The heuristic adapted to this problem is a simple calculation of the Manhattan distance to each possible successor state (moving left, right, up, down) if the move is legal (no wall), between the position of the two objects as this represents the minimum distance between them in tiles.

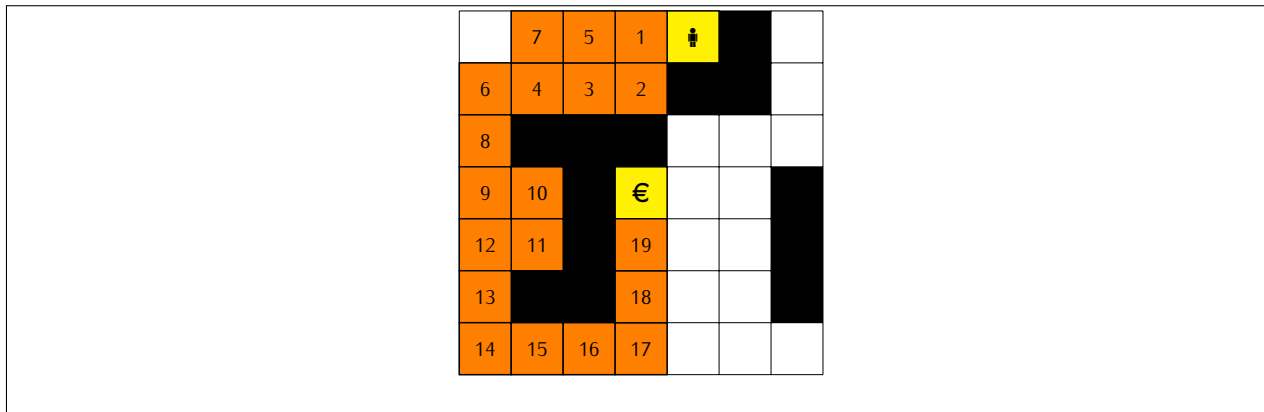
Admissible : The heuristic is admissible because any possible move, moves one tile, one step closer to the goal. And it does not overestimate the path to the goal since in the best case, the score of the heuristic is equal to the cost of the path (if there is no wall on the path).

Consistent : The heuristic is derived from a relaxed problem and it is an exact cost for this relaxed problem, so it must obey the triangle inequality (the side of one side cannot be longer than the sum of the two other sides) and is therefore consistent.

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)

6	4	2	1			
8	7	5	3			
9						
10	12					
11	14		21			
13			19	22		
15	16	17	18	20	23	

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 Blocks planning problem (17 pts)

1. Model the Blocks planning problem as a search problem; describe: (2 pts)

- States
- Initial state
- Actions / Transition model
- Goal test
- Path cost function

States : The description of a state describes the position of all elements on the grid as a row/column matrix. These elements are the walls (#), the movable blocks (lowercase letter), the target places (uppercase letter), well placed block (@).

Initial state : Can be any state as soon as moveable blocks placed on the grid, and as many target places of the same colour as moveable blocks. Also, the target places must be accessible.

Actions / Transition model : A block can move one square to the left or right, cannot cross a wall, but can cross a target square. An action (left or right) can lead a block to end up on top of empty squares, in which case it falls under the effect of gravity and stops on the first wall or block on which it falls. A block can also cause blocks above it to fall as it moves, these also suffer from gravity and fall. A block is placed is blocked on a target square when it is of the same color, and it cannot fall.

Goal test : It checks that all target places are filled with blocks of the good color.

Path cost function : Each step costs 1, so the path cost is the total number of steps in the path.

2. Consider the following state for the a01 instance:

```
#####  
#           #  
#  c       #  
#  a       #  
####      #  
####      #  
##  b     #  
#####
```

According to the goal state, such a situation cannot lead to a solution. Can you find other similar situations (in general, not only on that specific instance) that leads to a deadlock? If so, describe two. (2 pts)

Explanation of the example : When a block falls below one of it's target place, and there is no other block that can fill this specific target place, this means that this target place becomes unreachable. Very concretely, that's the example shown on the example, b is lower than it's target place, the target place is therefore unreachable and it is a deadlock.

Example 1 : When a block has reached its target place and is therefore impossible to move, and blocks the way for other blocks that have not yet reached their target place.

Example 2 : Another case of possible deadlock is for example when the map forms a kind of mountain, and the blocks are at the top of the mountain, if all the candidates blocks of a target place falls on the wrong side of the mountain, this leads automatically to a deadlock. This is an example with a mountain for the representation, but it appears every time a block has to choose a side to fall (if there is no way to return to the other side once the choice has been made).

3. Why is it important to identify dead states? How are you going to take it into account in your solver? (2 pts)

Why : Because it prevents from continuing in a direction that will never lead to the goal state. This allows to explore much less nodes by avoiding to go deeper in a branch with no goal state.

How 1 : Consider a block of color Z, in our solver, before accepting a new successor we test if it does not have all it's Z blocks that are in a X position lower than each of the Z target places.

How 2 : Consider a target block of color Z, before accepting a new successor we could test if all the target blocks are still reachable by at least one of their corresponding block Z. This is possible by checking the walls on each columns between a target block Z and his corresponding blocks Z.

4. **Describe** a possible (non trivial) heuristic to reach a goal state. Is your heuristic admissible and/or consistent? Why ? (2 pts)

Heuristic : The chosen heuristic is the sum of the horizontal distance between each block and it's target place (nearest and unique target place if several possible). It will also add a penalty of 2 if a full block separate a block from a target place on the column of the target place up to the line of the block. It will then add a penalty of 2 for each empty block between a target place and the floor.

Admissible : This heuristic is admissible as it uses the horizontal distance, and it is always less than or equal to the true distance to reach the goal. Even if we add penalties of 2, the minimum number of move necessary to reach a target place when a penalty is added is always greater or equals to the value of the penalty (2 here).

Consistent : The heuristic is consistent because it's obey the inequality triangle.

5. **Implement** this problem. Extend the *Problem* class and implement the necessary methods and other class(es) if necessary. **Experiment**, compare and analyze informed (*astar\_graph\_search*) and uninformed (*breadth\_first\_graph\_search*) graph searches of aima-python3 on the 10 instances of Blocks planning provided. Report in a table the time, the number of explored nodes, the number of remaining nodes in the queue and the number of steps to reach each solution. When no solution can be found by a strategy in a reasonable time (say 1 min), indicate the reason (time-out and/or swap of the memory).

Are the number of explored nodes always smaller with *astar\_graph\_search*? What about the computation time? Why? (3 pts)

A\* explores fewer nodes for the simple reason that the BFS is an A\* with a constant heuristic, which will never be more efficient than the heuristic we created. The computation time is also always lower, even if for a very specific instance, the computation time of our heuristic could give an advantage to BFS, but this does not happen here (for example, an instance with a tunnel and one block).

Inst.	A* Graph				BFS Graph			
	NS	T(s)	EN	RNQ	NS	T(s)	EN	RNQ
a01	12	0.013	60	28	12	0.045	330	27
a02	11	0.001	12	1	11	0.001	13	0
a03	5	0.005	28	23	5	0.026	158	80
a04	20	0.095	442	5	20	0.097	478	0
a05	48	1.890	2774	1404	/	>60.0	109132	33592
a06	21	2.649	4161	1940	21	12.40	29872	414
a07	16	1.893	3496	1726	16	9.353	21922	1929
a08	18	0.880	2296	1561	18	7.689	32451	5917
a09	25	3.429	4552	2905	/	>60.0	127225	24267
a10	38	10.04	19501	12991	/	>60.0	191425	21684

NS: Number of steps — T: Time — EN: Explored nodes — RNQ: Remaining nodes in the queue

6. **Submit** your program on INGIInious, using the A\* algorithm with your best heuristic(s). Your program must print to the standard output given a time limit of 1 minute, a solution to the Blocks planning instance passed as parameter to it, satisfying the described output format. Your program will be evaluated on 11 instances, one of which is hidden. We expect you go solve at least 8 out the 11. (6 pts)

The program has been submitted on INGINious.