INFO8006: Project 1 - Report

Maxime Goffart - s180521
Olivier Joris - s182113

October 2, 2020

1 Problem statement

a.
- The initial state is given by the layout of the maze which gives the initial position of Pacman and the initial positions of all the capsules and all the food dots. The possible states are a combination of Pacman position, and the positions of the remaining capsules and food dots.
- The legal actions are going north, going south, going west, and going east as long as Pacman does not go through a wall.
- The action taken will have the effect to change Pacman position in the maze (if the action is legal). If the cell on which Pacman arrives contains a food dot, the food dot is eaten and removed from the maze.
- The goal is reached when all the food dots have been eaten.
- The step cost can be defined as follows: each move of Pacman to a cell without a food dot costs 10 and each move to a cell with a food dot costs 1.

2 Implementation

a. The error lies in the `key` function and is the fact that it identifies a game state by taking only the position of Pacman. However, to identify the state of the game, the position of the food dots is also needed. Indeed, if Pacman has 2 positions that are the same, it may have eaten food dots in the meantime: the game state is then no longer the same.
This error can be corrected by returning a tuple containing both the current position of Pacman and the position of the food dots in the `key` function.

b. *Leave empty.*

c. The cost function $g(n)$ returns the cost of the path to arrive to the node n. This cost is a sum of step costs which are defined at the end of *section 1.*
The heuristic $h(n)$ is the Manhattan distance between Pacman and the farthest food dot in the maze associated to the game state given by node n.
The optimality is guaranteed if the heuristic $h(n)$ is consistent because we are using the graph search version of A*. The heuristic is consistent because we have: $h(n) \leq c(n, a, n') + h(n')$ with n a node, n' its successor, and a an action allowing to reach n' from n. The inequality is verified thanks to $c(n, a, n')$, the step cost between n and n', which is always positive.
In addition, the Manhattan distance between Pacman and the farthest food dot is always less than or equal to the true cost to a nearest goal.

d. If we take $h(n) = 0$ for all n, A* is equivalent to the UCS algorithm.
This algorithm is complete if all the step costs are greater than 0 which is the case (our step cost will always be 1 or 10).

Also, UCS is always optimal independently of the cost function (strange - heuristic ?) because it expands the nodes in order of their optimal path cost.

e. *Leave empty.*

f. In breath-first search, we want to expand the shallowest node in the fringe first. Using the depth of the node in the search tree as $g(n)$ allows to reach this strategy. So, the value of the heuristic $h(n)$ is no longer required.

3 Experiment 1

a. Plot using natural logarithmic values:

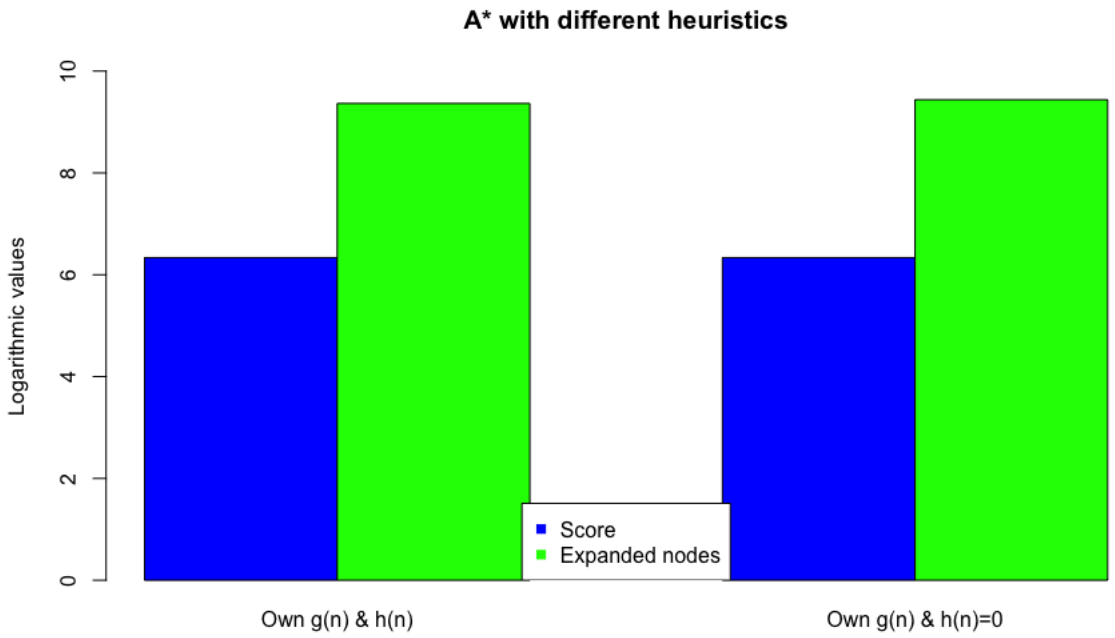

Figure 1: Comparison of A* algorithm with different heuristics in the middle layout.

b. The returned scores are the same with a value of 565. Using our heuristic function, the algorithm expanded 11674 nodes. While using $h(n) = 0$, the algorithm expanded 12560 nodes.

c. The version using $h(n) = 0$ expands more nodes because using $h(n) = 0$ is equivalent of using an uninformed search algorithm which does not take into account the possible location of a goal node.

d. A* with $h(n) = 0$ for all n corresponds to the uniformed-cost search algorithm.

4 Experiment 2

a. Plot using natural logarithmic values:

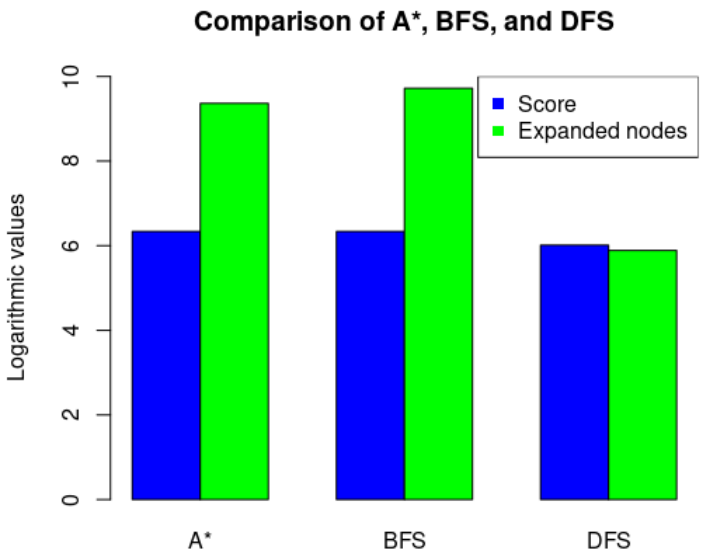

Figure 2: Comparison of the different algorithms in the middle layout.

b.
- A* is better than DFS in terms of score and expanded nodes. (A* is better than DFS in terms of score but worst when it came to the number of expanded nodes?)
- A* is better than BFS in terms of node expanded but is equal to it in terms of score.

c. These observations are due to the fact that:

- DFS is not optimal because it is going as deep as possible in the search tree until it finds a step goal which reduces the number of expanded nodes.
- BFS is optimal in this case but not in all cases because it is exploring the shallowest unexplored node which is not always the optimal one. Therefore, the number of expanded nodes is high.
- A* is getting the best score because, as we have seen in the theoretical course, this search algorithm is optimal. It is taking into account the possible location of a goal node so its number of expanded nodes is lower than the one for BFS.