MAI Coding Project

Didier Quintius and Thao Hoang Phuong

October 4, 2020

In this exercise a series of search algorithms are discussed. These search algorithms will be implemented on a partially connected network of nodes, to find a (the best) path from the start to the goal node. Each search algorithm has different properties, advantages and disadvantages, which are reported and then tested on the network. The next paragraph discusses the simulation of the network of nodes.

Some characteristics of the network are; each simulated has N nodes, node 1 is the start node and node N is the goal node and the cost of path AB is always equal to that of BA. The function used to simulate the network is `network_of_nodes(amount_of_nodes, [costs], [probabilities], [disconnect_start_goal], [seed])`. The function creates a symmetric matrix with NA values, which mean that there is no connection between the two nodes, and positive integers, equal to the costs of travelling between the nodes.

- The `amount_of_nodes` variable sets the amount of nodes and is compulsory variable.
- The `costs` variable is a list with costs that each connection could possible take on. Note that the NA cost needs to be added else a fully connected model is produced. The default list is `[NA, 1,2,3,4]`.
- The `probabilities` is a list with the probabilities of each of the costs defined in `costs`. Thus, must be of equal length. Note that the probabilities must be given in integers. The default value is `[5, 2, 1, 1, 1]`.
- The `disconnect_start_goal` removes the direct connection between the start and the goal if set to `True`. The default value is `True`.
- The `seed` is used to set the random seed when creating the network. The default value is `0`.

To obtain the connections of each node, use the function `return_connections(node)`. This function will return a list of the nodes connected to the given node:

```
network = network_of_nodes(100, probabilities = [1000, 1,1,1,1])
network.return_connections(0) = [2, 23, 56]
```

To obtain the cost between two nodes, use the function `get_distance(node1, node2)`. This function returns the cost between `node1` and `node2`.

Using the functions given above the search algorithms are tested on their speed, completeness and memory. The algorithm that are tested include: Depth-First, Breadth-First, Non-deterministic, Iterative Deepening and Bi-directional.