Task 2: Heuristic Search Methods

Hill-climbing 1, Hill-climbing 2, Beam search, Greedy search

The following section will describe how the simulation of the network functions. The network is fed **4** variables: **N, BranchingFactor, ConnectionRatio, Seed.**
N stands for the number of nodes in the network, including the start and goal node. The network is created by simulating a **NxN** matrix, the connection matrix C**.** Here C_{ij} is 1 if there is a connection between $node_i$ and $node_j$, and NA if there is no connection. Because a node cannot be connected to itself, the diagonal of the matrix is always NA. Next because a node is always connects two nodes $C_{ij} = C_{ji}$. Thus the matrix is symmetric and when we talk of all of the connections we refer to the lower triangular without the diagonal. These are the unique connections present in the network. Which is equal to $N\,(N-1)/2$ connections.

The **BranchingFactor** is the average number of connections of each node. Using this variable we can calculate the amount of connections that should be present in the network on average. We also use the fact that each connection always connects only two nodes, not more nor less. Therefore we arrive at the following equation:

$$TotalConnections = \frac{BranchingFactor \times N}{2}$$

The **ConnectionRatio,** has to do with the fact that the probability of a connection between $node_i$ and $node_j$ is dependent on the **IndexDifference**: $d_{ij} = |i - j|$. The **ConnectionRatio** is the equal to:

$$ConnectionRatio = \frac{P(node_i, node_j \mid d_{ij} = 1)}{P(node_i, node_j \mid d_{ij} = N-1)}$$

, thus the ratio between the probability of a connection between nodes with an **IndexDifference** of 1 divided by a connection between nodes with the maximum **IndexDifference** of $N-1$. The probability between nodes is obtained using the following function:

$$P(node_i,\ node_j) = \beta e^{-\alpha \times d_{ij}}$$

To find the value of α we substitute this function into that of the **ConnectionRatio:**

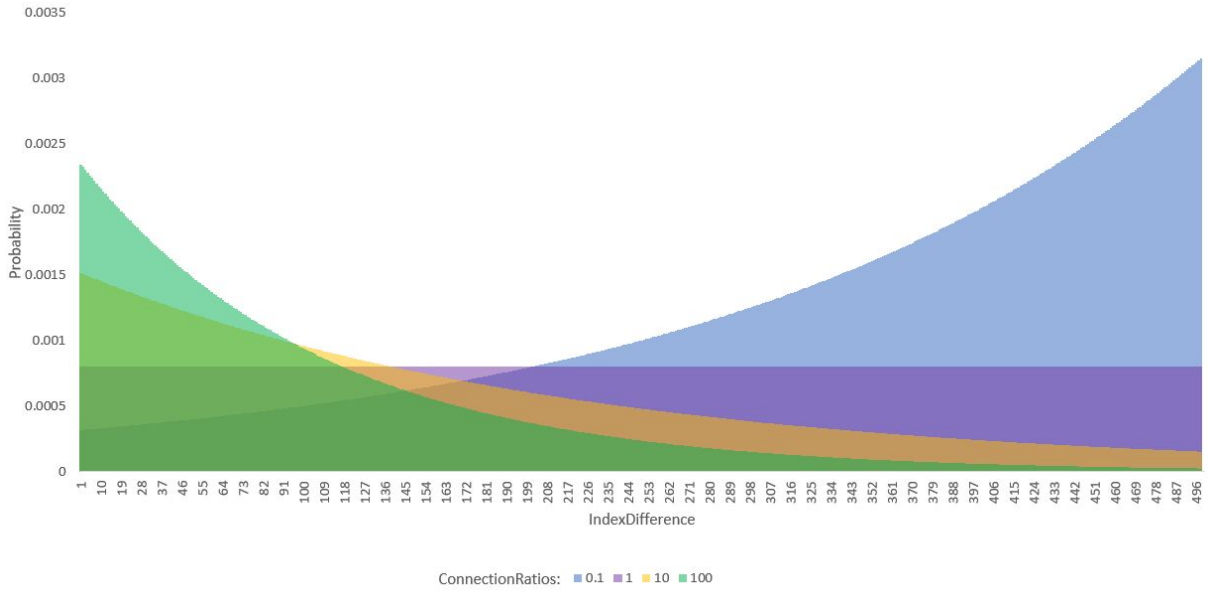
$$ConnectionRatio = \frac{\beta e^{-\alpha}}{\beta e^{-\alpha (N-1)}}$$
$$ConnectionRatio = e^{\alpha (N-2)}$$
$$\alpha = \frac{ln(ConnectionRatio)}{N-2}$$

To find the value of β we use the fact that the sum of the probability of all connections should be equal to **TotalConnections.** Note that when summing over the probabilities of all connections, there are multiple connections with a certain **IndexDifference,** namely: $n_d = N - d$, where n_d is the amount of connections with an underlying **IndexDifference** d**.** For example there are N - 1 connections with an underlying **IndexDifference** of 1 and only 1 probability with an underlying **IndexDifference** of N-1.

$$TotalConnections = \sum_{d=1}^{N-1} (N - d)\beta e^{-\alpha d}$$

$$\beta = \frac{TotalConnections}{\sum_{d=1}^{N-1} (N-d)e^{-\alpha d}}$$

The following graph shows the Probability of a connection for **N** = 500, **BranchingFactor** = 0.2 and different **ConnectionRatio**s:



At first glance one could expect that the graph of **ConnectionRatio** = 0.1, would be the inverse of that with **ConnectionRatio** = 10. In the first scenario the probability with an underlying **IndexDifference** of N - 1 is 10 times bigger than those with an **IndexDifference** of 1. This relation is inverted in the second scenario but the graph is not. This is due to the fact that the sum of all probabilities needs to be equal to the **TotalConnections** and there are more probabilities with lower **IndexDifferences,** $n_d = N - d$. Thus the area under the graph is lower for higher **ConnectionRatio**s.

Depending on the amount of nodes and the branching factor and mostly the connection ratio it is possible that the probabilities become higher than 1. Therefore the function outputs a message, that the connectionratio is too high or two low.

Next there are values sampled, the same size as C, from a uniform distribution between 0 and 1:

$$U_{ij} \varepsilon\ unif(0,1)$$

If the value is lower than the probability then there is a connection, else there is none.

$$C_{ij} = 1,\ U_{ij} \leq P(node_i,\ node_j)$$
$$C_{ij} = 0,\ U_{ij} > P(node_i,\ node_j)$$

Lastly a path to the goal is simulated to make sure that there is at least one connection from the start to the goal. Firstly a path is simulated of a length 5 to N-1, meaning that 5 to N-1 nodes are sampled in an arbitrary order. The matrix is filled such that there are connections between each of the consecutive nodes in the path to the goal.

There are three methods attached to the network:
Check_connection receives two nodes and returns if there is a connection
Return_connections receives one node and returns all connections to that node
Get_heuristic receives a node and returns the heuristic of that node. The heuristic that was chosen is:

$$h(node_i) = (1 - ConnectionRate)\ i$$

Because the idea of a heuristic is to give an indication of how close a node is to the goal. Generally the heuristic is higher when the node is far from the goal and lower when the node is close to the goal. When **ConnectionRate** is lower than 1, the probability increases with the the **IndexDifference** and the term $1 - ConnectionRate$ is positive and the heuristic increases as the IndexDifference increases and the opposite for when the ConnectionRate is greater than 1. If the **ConnectionRate** is 1, all probabilities are equal thus the heuristic is constant for each node.