IN3063 Programming and Mathematics for AI Coursework Task 1

By Tayo Ajibode-Folkes

<u>Table of Contents</u>

Implementation

The aim of task one was to create a grid game based on a grid of random numbers between 0 and 9. The objective was to get from the top left of the grid and find a path to the bottom right of the grid, the length of this path was to be calculated using a basic heuristic algorithm that would be better than random movement, and then using Dijkstra's shortest path algorithm to find the shortest length and then using the Ant colony optimisation Algorithm which finds the shortest path via graphs. In order to achieve the flexibility asked of me I allowed the player to change aspects of the grid such as height and width as well as the numbers that are randomised within. Changing these altered the parameters that the algorithms worked with.

Basic Heuristic Algorithm

For the basic heuristic algorithm I made a simple algorithm which moved the cell downwards and then to the right, it would do this while under the conditions that:

1. The cell was not located on the bottom row

2. The cell was not located on the right most row

If 1 was false then the algorithm would no longer move the cell downwards and would only move right

If 2 was false then the algorithm would no longer move the cell to the right and would only move downwards

If 1 and 2 were false then that would mean that the cell would be in the target destination and the loop could stop, ending the algorithm.

The path would have not backtracking, as the movement was only down and right with no left or up movement.

The end result is that the algorithm would perform better than a random path.

Dijkstra's Algorithm

For the Dijkstra algorithm, I knew this would be the more challenging section of task one so I made sure to give myself time for research and execution. I approached the task by transforming the grid into a node weighted graph making each cell would represent a vertex. I then built the Dijkstra algorithm by looping through all the cells of the grid and taking note of the neighbours of each cell and adding them to a queue. Then within the range of the vertices I would compare vertex points and plot the shortest path and calculate the distance, this was achieved with loops, conditional statements (if statements etc.), and queues. Because of some issues with my laptop testing took a long time but through different methods I was able to get it done.

Ant Colony Optimization (ACO) Algorithm

The Ant Colony Optimization (ACO) algorithm is used to find the optimal path through the grid from the top-left cell to the bottom-right cell. ACO is inspired by the foraging behaviour of ants and aims to solve optimization problems by simulating the process of ants finding the shortest path between their nest and a food source. In the context of this grid game, the ACO algorithm is applied to find the path that minimizes the total cost of traversal. A pheromone matrix is initialized with equal pheromone levels for each edge (vertex pair) in the graph representation of the grid. A specified number of ants (controlled by the num_ants parameter) are deployed to find paths through the grid. Each ant constructs a path by iteratively selecting the next vertex to move to based on pheromone levels and a heuristic value. The heuristic value in this case is the absolute difference in cost between the current cell and the neighbouring cell. After each iteration, the pheromone levels on each edge are updated. The more ants choose a particular edge, the higher the pheromone level on that edge becomes. Pheromone evaporation is also applied to reduce pheromone levels over time to encourage exploration. Once all ants have completed their paths, the algorithm selects the path with the lowest total cost as the optimal path.

Analysis

The shortest path length increases if the dimensions of the grid increase, this is because the algorithm will have more cells to cover overall. Even if the path from the top left to the bottom right was a straight path if the grid was 500x500 it would still have a larger path length than a 5x5 whose path is very curved. If the size of each of the cell numbers generated was greater, the sum of all the cell numbers in the shortest path would be greater exponentially. The runtime of each algorithm depends on the size of the maze (grid). The heuristic algorithms tend to run faster, with a time complexity roughly proportional to the number of cells in the grid. Dijkstra's algorithm, on the other hand, has a time complexity of O(n^2) using the provided grid-to-graph transformation. ACO's runtime depends on the number of iterations and the number of ants, making it harder to predict precisely, but it generally performs well for larger grids.

References

I used the lecture videos and notes about Dijkstra's algorithm to help me plan my methodology and build my Dijkstra algorithm code, I also watched the following videos to give me a better understanding of how to implement and structure my code:

https://www.youtube.com/watch?v=OrJ004Wid4o

https://medium.com/p/a514a9d31076

https://www.youtube.com/watch?v=EJKdmEbGre8

https://www.youtube.com/watch?v=X-iSQQgOd1A

https://www.youtube.com/watch?v=Sk9QQUGMdY8

I also learned more about using python classes, functions, syntax, etc. from the following websites:

https://www.programiz.com/python-programming/

https://www.programiz.com/python-programming/function

https://www.programiz.com/python-programming/class-object

https://www.w3schools.com/python/

https://www.w3schools.com/python/python_classes.asp https://www.w3schools.com/python/numpy/default.asp

https://www.w3schools.com/python/python_functions.asp

https://www.geeksforgeeks.org/queue-in-python/

https://induraj2020.medium.com/implementation-of-ant-colony-optimization-using-python-solve-traveling-salesman-problem-9c14d3114475#:~:text=Implementing%20Ant%20colony%20optimization%20in%20python%2D%20solving%20Traveling%20salesman%20problem,-Induraj&text=Ant%20colony%20optimization%20(ACO)%20is,by%20the%20behavior%20of%20ants.