Group 23 Report

Andrej Erdelsky — Dean Polimac — Mateja Zatezalo — Lucas Fatas

18 March 2021

1 Introduction

This report discusses path optimization using brute force methods. First part of the report deals with Ant Colony Optimization in which Swarm Intelligence is used to find the most optimal path in a maze. Second part of the report deals with the Travelling Salesman Problem (TSP) using a Genetic Algorithm.

2 Part 1 - Path finding through Ant Colony Optimization

2.1 Ant preparation

2.1.1 Features of the Maze

The maze used is a predetermined N by M matrix - N being the width and M being the height of it in our case - whose entries represent accessible tiles and inaccessible tiles, 1 being accessible and 0 being inaccessible, respectively. As in real life, our maze can have dead ends, loops, large accessible areas, and crossroads which are some of the features that make a maze harder to traverse. The way we deal with these will be discussed later in the report.

2.1.2 Amount and Purpose of Pheromone in the Algorithm

In this algorithm ants communicate via pheromone in order to navigate through the maze. They release the amount of pheromone proportional to the length of the path they took. The amount dropped on each tile of the path is calculated using equation 1. In this equation, $\Delta\tau_i^k$ stands for the pheromone left by k^{th} ant on i^{th} path, Q represents a constant amount of pheromone that each ant can drop over a path, while L_i represents the length of the path traversed.

The pheromones are used in order to increase the probability of an ant going to a matrix entry (a tile on the path) which was visited before. Of course, ants will sometimes still take a path that was not taken before, but we want them to have a larger probability of visiting a already visited path. The more popular a path is for ants to traverse, the more pheromone will be left on that path. Since

the amount of pheromones on a path dictates its probability to be chosen by future ants, the shortest, or at least the most optimal path will always be the most popular one.

$$\Delta\tau_i^k = \frac{Q}{L_i} \tag{1}$$

2.1.3 Pheromone Evaporation

Pheromone evaporation removes a fraction of pheromone trails that may misguide ants when a more optimal path is found. Evaporation constant ρ is set to 0.1 in our case for the easy maze, which means 90% of pheromone is left after each iteration, and the rest will vaporize. To this number, we add the sum of all pheromones left by ants this iteration, as seen in equation 2.

$$\tau_{ij} = (1-\rho) * \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k \tag{2}$$

2.2 Implementing Swarm Intelligence

2.2.1 Ant-Algorithm Pseudo-Code

The pseudo code explaining how a single ant traverses the maze can be can be seen in Algorithm 1.

direction_list is the list of possible directions an ant can go to.
direction direction in which the ant should go.
weights list of weighted probabilities of an and going into a certain direction.
current_position current position of the ant in the maze.
route route that the ant has taken.
visited list of visited positions.
start is the starting position of the ant.
end is the ending position of the ant.

Algorithm 1 find_route(self)

while *start* ≠ *end* **do**
  weights ← **if** direction in *visited* **then** 0 **else** possible pheromones
  direction ← randomly chosen direction from *direction_list*
  current_position ← coordinate in *direction*
  visited ← appends the *current_position*
  route ← appends the *current_position*
return *route*

2.3 Upgrading Ants with Intelligence

2.3.1 Improvements to the Ant-Algorithm

One glaring issue that might arise during the exploration of the maze is arriving at a position where all surrounding positions are either walls, maze edges or already visited positions. This is treated as a dead end for the ant, meaning the it has to find its way back to a previous position where it could choose a different path to explore. This is achieved by storing a set of visited positions for each ant in a map, but we will also store them in a stack that represents the final route the ant took. Every time an ant visits a position, we store it. In case an ant gets to a dead end it needs to backtracks, meaning it will keep return to previous positions in its route - whilst removing them from the stack - until there paths free to explore. A way to determine this is by checking if the sum of all surrounding pheromones is equal to 0, since when a position is already visited for an ant, its probabilistic weight - used when choosing directions - will be 0 for that specific ant.

2.4 Parameter Optimization

2.4.1 Varying parameters

First, let us define the varying parameters that we use. These parameters include the number of ants per generation, number of generations, variable Q which represents the number of pheromones that each ant in a generation drops, and rho which is the evaporation factor. Initially number of generations and number of ants per generation were set to one while testing if the code compiles, Q was set to 1600 and rho was 0.1. When we established that the algorithm works as it is supposed to we started adjusting the parameters for the easy maze. Shortly afterwards we concluded that 20 generations and 20 ants per generation with Q and rho having their initial values was enough for us to quickly converge to a distance of 38 units, which indeed is the shortest path. The quick convergence means that with these parameters the colony does not require a lot of generations to get to the optimal solution. Moving on to the medium maze, this took a bit more time to find somewhat optimal parameters, with Q and rho set to 2200 and 0.05, respectively. The number of generations required and ants per generation was equal to 20, and 100, respectively, and resulted in a distance of 173 units. Finally, for the hard maze it took significantly more computational time when compared to the previous two mazes. The best result of 873 units was achieved using 120 ants for 20 generations. As it was more time consuming than other mazes, we tried out a few values for Q and rho and settled for 2200 and 0.03, respectively.

2.4.2 Dependency of the Parameters

The relation between parameters and the size of maze is best reflected on the amount of pheromones the ants drop, and the evaporation factor. If we were

to use the same amount of pheromones for all three mazes, we would see that it does not scale well. This is because larger mazes will usually require the ant to cover a larger distance, and therefore less pheromone will be left on each tile along the path. This increases the probability of ants taking paths that are not visited. This increases the chances of optimal paths being lost due to the evaporation as the amount of pheromone on them will be smaller. The same applies if all the mazes use the same evaporation factor, as the amount of pheromones evaporates faster in a larger maze.

3 Part 2 - Travelling Robot Problem

3.1 Problem analysis

3.1.1 Definition of the TSP

The Travelling Salesman Problem is an algorithmic problem that deals with finding the shortest path between a given set of nodes, where the starting and the ending node are the same.

3.1.2 Difference from Classic TSP

The task is to solve a modified version of TSP where the starting and the ending nodes differ. In order to do so we use a Genetic Algorithms.

3.1.3 Solving TSP with Computational Intelligence

Computational Intelligence algorithms such as Ant Colony Optimization (ACO) and Genetic Algorithms (GA) solve large problems like TSP. With a number of generations, the algorithm will learn and select better routes in order to find the most optimal one. GA uses chromosomes and fitness which can easily be assigned to adapt to TSP constraints and conditions.

3.2 Genetic Algorithm

3.2.1 Genes and Encoding Chromosomes

The encoding of our chromosomes is quite simple. The encoding is a list of 18 values (representing 18 products), ranging from 0-17, with no duplicates. This encoding represents the order in which the products will be picked given the starting and ending points.

3.2.2 Fitness function

The fitness function we use is $100 * (1 - \frac{d - d_{min}}{d_{max} - d_{min}})$ where d is the distance between two consecutive products in the list, while d_{min} and d_{max} are minimal and maximal distances between all products.

3.2.3 Parent Selection

Using the fitness function, we calculate the fitness of each chromosome and then divide it by the sum of the fitness of the whole population. We use this number as the probability for picking a chromosome as a parent, because we want the chromosomes with the best fitness to have the highest probability of getting reproducing. We then randomly select two different parents using these probabilities.

3.2.4 Functions of genetic parameters

For our cross-over function we use order crossover (OX) which was relatively simple compared to other options, but was still quite effective. Regarding mutation, we pick two random products in the encoding for the chromosome and swapped their positions.

3.2.5 Preventing points from being visited twice

Both the cross-over and the mutation function we use ensured that there would be no duplicates, and when the initial population is encoded with random paths without duplicates as well.

3.2.6 Local Minima

The way we try to prevent our algorithm from reaching a local minima is with mutation. When our mutation probability is set to zero, our algorithm frequently gets stuck in local minima. When our mutation probability is set to 0.01, the algorithm almost always reaches the same level of fitness.

3.2.7 Elitism

Elitism in Generic Algorithms means choosing the best and most fit genes to be carried over onto the next generation. This allows our solution to either stagnate or converge to a optimum.

4 Discussion and Conclusion

We have concluded that Swarm Intelligence, more specifically the ant colony optimization algorithm is a valid and efficient way of finding optimal routes in complex mazes. Moreover, the Travelling Salesman Problem given during this assignment was able to be solved by a Genetic Algorithm of our own making, producing a satisfying result in adequate time.