ENG335
Computational Intelligence

Assignment 3 - Genetic Algorithms

Tyler Robards - 651790, Baley Eccles - 652137

Contents

1 Introduction

This assignment applies a Genetic Algorithm (GA) to optimise the placement of an Emergency Response Unit (ERU) with a 7×7 km city grid. Each grid sector represent the annual rate of emergency incidents, λ. The goal is to minimise the average weighted distance between the ERU and all city sectors, therefore reducing the average response time to emergencies.
Two cases are analysed:

1) Flat City - No obstacles in the city grid, the ERU can be placed anywhere.

2) River City - A river divides the city at $x = 5$ km with a single bridge at $(5.0, 5.5)$ km forcing detours across the bridge.

The Genetic Algorithm is developed in MATLAB to locate the global optimum ERU position, and is visualised through a progress plot. Simple testing of GA parameters and result visitation is implemented through an interactive GUI built with the MATLAB App Designer.

2 Domain Problem

This optimisation problem seeks to minimise total travel distance to a single ERU location.
The specification defines the objective function as the weighted total emergency response distance,

$$f(x_{eru}, y_{eru}) = \sum_{n=1}^{49} \lambda_n \sqrt{(x_n - x_{eru})^2 + (y_n - y_{eru})^2}$$

where,

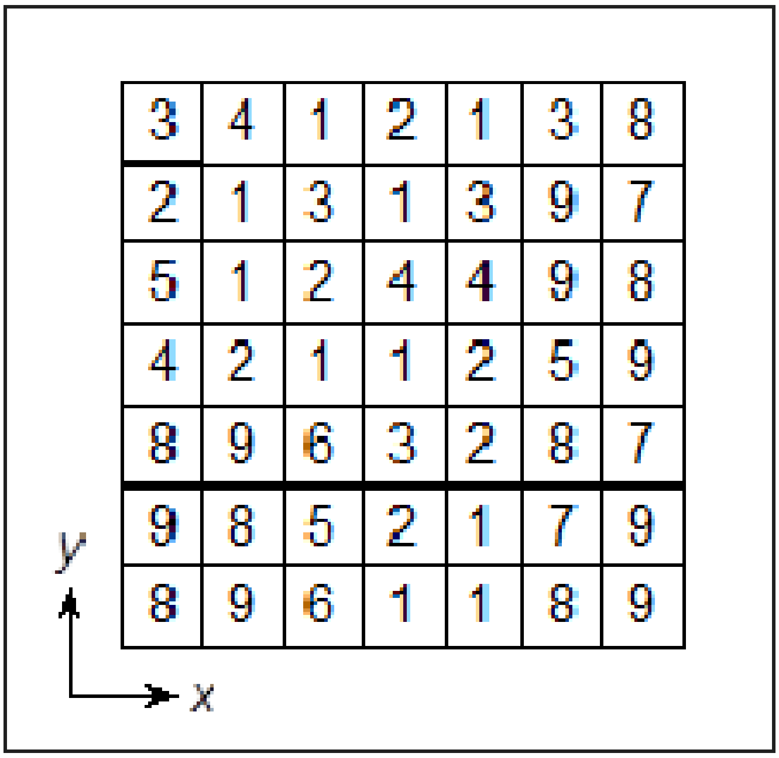
- λ_n is the number of emergencies per year in sector n,
- (x_n, y_n) are the coordinates of the center of sector n, and
- (x_{eru}, y_{eru}) are the coordinates of the ERU location

The goal is to minimise the this sum, $f(x, y)$, since a smaller value means a shorter response time. However, most genetic algorithms are designed for maximisation problems. To use the GA for this task, the reciprocal of the objective function is used as the fitness function,

$$f_{GA}(x, y) = \frac{1}{f(x, y) + \epsilon}$$

where ϵ is a small constant to avoid division by zero.
This allows the algorithm to maximise the fitness while minimising the original objective.

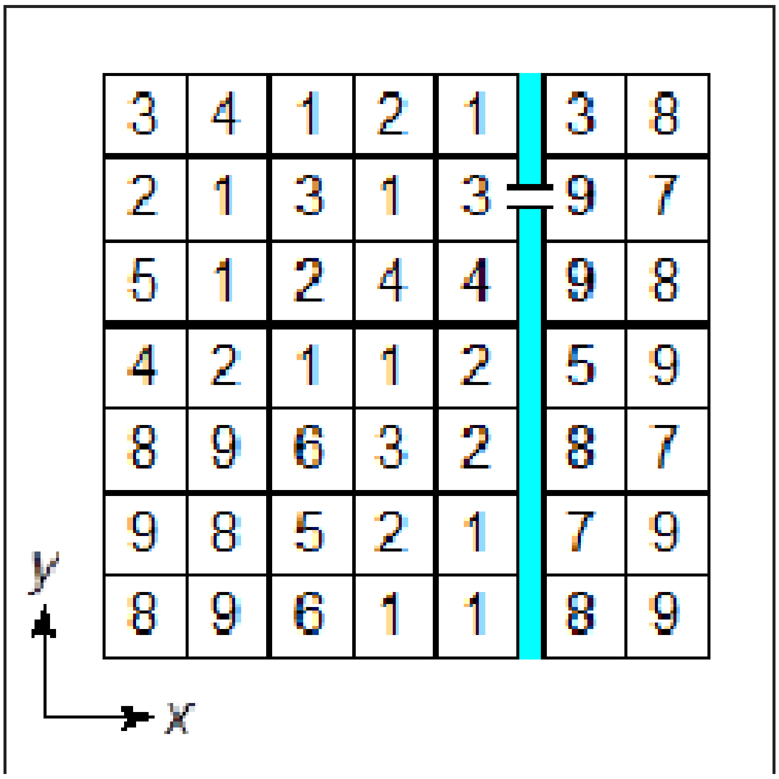
This project employs two distinct city layouts. The first configuration, the Flat City (Figure 1), represents a simple environment without geographical barriers. The ERU can travel directly to all sectors in a straight line.



3	4	1	2	1	3	8
2	1	3	1	3	9	7
5	1	2	4	4	9	8
4	2	1	1	2	5	9
8	9	6	3	2	8	7
9	8	5	2	1	7	9
8	9	6	1	1	8	9

Figure 1: Flat City Grid

The second configuration, the River City (Figure 2), includes a realistic constraint, a river running through the city at $x = 5$ and a bridge at $(x, y) = (5.0, 5.5)$. When the ERU and the emergency are on opposite sides of the river the distance must be calculated differently, as the sum of the distance for the ERU to the bridge and the distance from the bridge to the relevant sector.



3	4	1	2	1	3	8
2	1	3	1	3	9	7
5	1	2	4	4	9	8
4	2	1	1	2	5	9
8	9	6	3	2	8	7
9	8	5	2	1	7	9
8	9	6	1	1	8	9

Figure 2: River City Grid

By comparing the Flat City and River City configurations the impact of realistic geographical constraints can be observed.

2.1 The Travelling Salesman Problem

The travelling salesman problem (TSP) asks the following question "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" The TSP is a similar optimisation problem to that in this assignment, both aim to minimise the total travel distance, however this task has a single ERU location rather than a closed route.

3 Genetic Algorithm Description

The Genetic Algorithm (GA) for this project is designed to find the best possible location for the Emergency Response Unit (ERU) within a 7×7 km city grid. The objective is to minimise the total weighted distance between the ERU and all 49 city sectors, where each sector has a specific emergency rate. This ensures that areas with higher emergency rates have higher weighting on the final ERU location. Each possible ERU location is represented by integer coordinates corresponding to the center of each sector in the city. The algorithm begins by randomly generating an initial population of ERU positions, allowing it to explore the entire search space. This spread increase the chances of finding the global solution, rather than getting stuck in a local maximum.

Once the initial population is generated, each candidate is evaluated using the fitness function that measures how effective each candidate is at minimising emergency response distance to each city sector.

$$f(x,y) = \frac{1}{\sum_{n=1}^{49} \lambda_n \sqrt{(x_n - x_{eru})^2 + (y_n - y_{eru})^2}}$$

For sectors on the same side of the river as the ERU the direct euclidean distance is used, however sectors across the river must be calculated differently. The total distance will be the sum of two segments, from the ERU to the bridge at $(5.0, 5.5)$ and the bridge to the relevant sector. These distances are multiplied by each sectors weighting, λ_n, producing a weighted total distance that represents the ERU's overall effectiveness at the current candidates position. As the GA is designed to maximise the fitness, rather than minimise it as needed for this task, the reciprocal of this weighted sum is used, allowing the GA to maximise fitness as which corresponds to minimising distance.

The optimisation process then proceeds through several generations using biologically inspired operations, selection, crossover and mutations. During selection, the algorithm uses the tournament method, where small groups of candidates are randomly chosen and the one with the highest fitness becomes a parent. This ensures that the best solutions have a higher chance of reproducing whilst retaining genetic diversity. The crossover stage combines information form two parents to produce offspring, using a uniform crossover approach where each coordinate has an equal probability of being inherited from either parent. This allows good characteristics from different solutions to mix and form potentially stronger candidates. The mutation step introduces random changes by modifying one or both coordinates to a new grid location. This prevents stagnation and helps the GA escape local solutions. Elitism is also used to ensure that the best performing candidates from one generation are carried over, completely unchanged, to the next, preventing the loss of high quality solutions.

To ensure the algorithm is performing well and to validate the final solution an exhaustive search is used alongside to GA. This brute force method calculates the fitness for all possible ERU locations and identifies the best solution. The results from the GA are compared to this exhaustive optimum to check the GA converges to the true global solution.

3.1 Algorithm Testing

To test the genetic algorithm the GUI shown in figure 3 was developed. This allows the user to easily change the GA parameters and display the results.

Table 1: GA Test Configurations

Test Case	Population	Generations	p_c	p_m	Elitism	Tournament Size	Fitness	Solution Position
Test 1	5	50	0.90	0.15	1	2	0.001313	(4.5,5.5)
Test 2	10	75	0.85	0.25	1	3	0.001313	(4.5,5.5)
Test 3	8	100	0.80	0.50	0	2	0.001313	(4.5,5.5)
Test 4	12	100	0.90	0.05	3	4	0.001313	(4.5,5.5)
Test 5	15	100	0.90	0.20	2	3	0.001313	(4.5,5.5)

To evaluate the performance and robustness of the system five tests were conducted using different GA parameters, shown in Table 1. Despite significant variations in parameters all test successfully converged to the same solution at $(x, y) = (2.5, 5.5)$, with a fitness value of 0.001313. This result matches the exhaustive search solutions, confirming the reliability of the algorithm. The rapid convergence, typically within 50 generations, indicates the search space is relatively simple and that the fitness surface contains a singular optimal solution. Tests with lower mutation probabilities and higher elitism (Tests 1 and 4) converged slightly faster, suggesting a limited need for exploration once the population reaches the correct region. Configurations with higher mutation rates (Test 3) were slower to converge as they explored the region more, however they still settled to the same global solution. Overall, these results display that the GA implementation is both stable and efficient for this task, requiring minimal tuning to achieve consistent and accurate solutions.

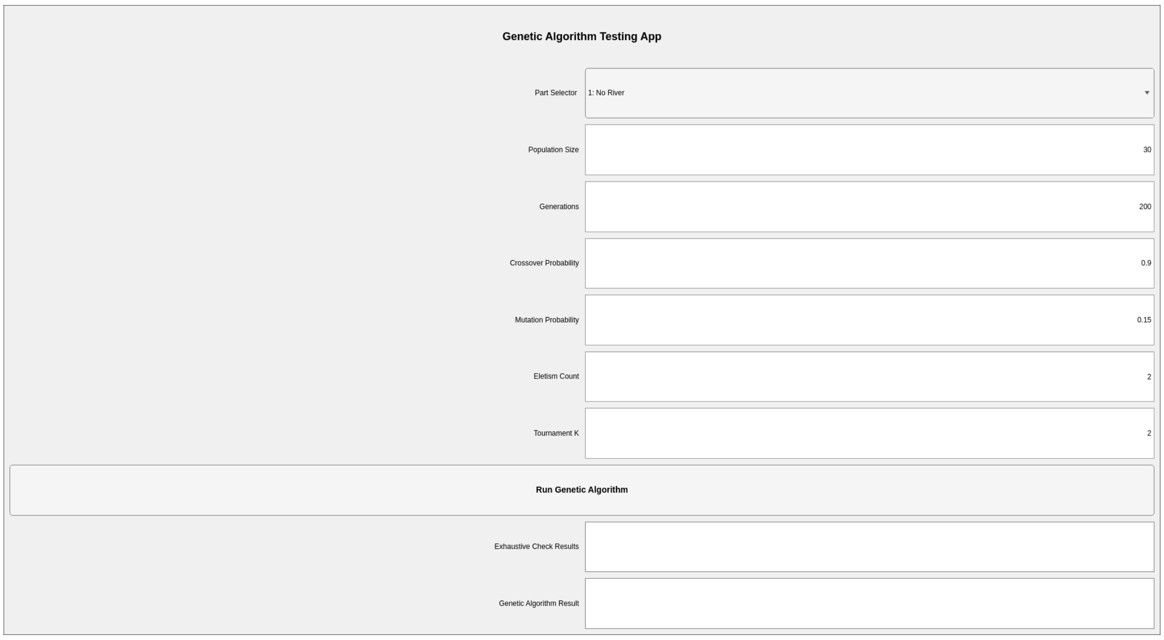
3.2 Final Parameter Selection

Based on the testing conducted, a middle ground between Tests 1, 3 and 4 was selected. This balances speed and stability, whilst allowing the population to explore the region.

Table 2: Final Parameter Selection

Parameter	Symbol	Value
Population Size	N_p	8
Generations	N_g	75
Crossover Probability	p_c	0.85
Mutation Probability	p_m	0.2
Elitism Count	-	1
Tournament Size	K	2

3.2.1 GUI User Guide


Genetic Algorithm Testing App

Part Selector: 1: No River

Population Size: 30

Generations: 200

Crossover Probability: 0.9

Mutation Probability: 0.15

Eletism Count: 2

Tournament K: 2

Figure 3: GUI

Features:

- Drop down selection to choose if the river is enabled
- Editable fields for population, number of generations, crossover and mutation probabilities, elitism count and tournament size.
- Run GA button executes the algorithm with the current settings
- Emergency rate heatmap, showing river/bridge, and markers for GA and exhaustive solutions

Steps:

1) Extract `Simulation Files.zip` and open `Main.m`
2) Run `Main.m`. This will open the GUI
3) Enter desired GA parameters
4) Click `Run GA` to execute
5) Results will be displayed in the bottom text areas. The plot progess plot will be displayed, showing the result from both the Genetic Algorithm (red circle) and Exhaustive Check (star)

4 Conclusions

This assignment successfully applied a Genetic Algorithm to optimise the placement of an Emergeny Response Unity within a simplified city grid. The algorithm was able to minimise the total weighted emergency distance for both city layouts, the Flat City and the River

City. The algorithm consistently located the same optimal solution as the exhaustive check. This confirmed the algorithm implementation was accurate, efficient and well suited for this optimisation problem.

Through parameter testing, the algorithm demonstrated reliable convergence behaviour across all configurations. Lower mutation rates accelerated convergence, while higher mutation rates promoted exploration at the cost of slower convergence. The final selected parameters (Table 2) provided a balance between exploration and speed, achieving consistent and accurate results.

Overall, the Genetic Algorithm proved to be an effective optimisation tool. It successfully modelled realistic urban challenges and provided insight into parameter tuning and geographical features influence over performance. The final MATLAB implementation and GUI also proved a useful platform for visualising behaviour and exploring different parameter configurations.

End of Assignment 3