The LNM Institute of Information Technology

# Soft Computing

# Project

Course Instructor: Dr. Aloke Datta

Submitted by:

- Utkarsh Singh (22 UCC111)
- Chandan Deb (22 UCS052)
- Anirudh Daga (22UCC015)

Submission Date: 20/04/2025

LNMIIT
The LNM Institute of Information Technology

## Contents

LNMIIT
The LNM Institute of Information Technology

# Problem 1: Fuzzy Logic Controller

## 1.1 Problem statement

Design a fuzzy logic controller using Mamdani's approach to determine the wash time of domestic washing machine, assume that inputs are dirt and grease on clothes. Use 5 descriptor for dirt (i.e., VSD, SD, MD, HD, VHD) and 3 descriptor for grease (i.e., SG, MG, HG) and 5 descriptors for output variable of wash time (i.e., VST, ST, MT, HT, VHT). Use triangular membership functions for fuzzy classes and CoG method for defuzzification operations. The set of rules for fuzzy logic tables are given below.

| | SG | MG | HG |
|---|---|---|---|
| VSD | VST | VST | ST |
| SD | VST | ST | MT |
| MD | ST | MT | HT |
| HD | MT | HT | VHT |
| VHD | HT | VHT | VHT |

The input grease is given in the scale of 0 to 50, whereas dirt is in the scale 0 to 100. Output should be in the range 0 to 60 min. Write a program to find the wash time in minutes for any value of grease and dirt within the range.

## 1.2 Mamdani Fuzzy Logic System

**Mamdani Fuzzy Inference System** is one of the most commonly used fuzzy logic systems. It mimics human reasoning by using fuzzy sets and rules to map inputs to outputs. In our washing machine problem, it helps determine the **wash time** based on **dirt** and **grease** levels on clothes.

Here's how it works step-by-step:

**1. Fuzzification**

- **Inputs** (Dirt and Grease) are crisp numerical values.
- These are converted into **fuzzy values** using **triangular membership functions**.
- For example:
    - If dirt = 40, it might belong partially to both **SD** (Slightly Dirty) and **MD** (Moderately Dirty).
    - If grease = 35, it might belong to both **MG** (Moderate Grease) and **HG** (High Grease).

## 2. Rule Evaluation (Inference)

- Each combination of Dirt and Grease is evaluated using **fuzzy rules** (defined in our table).
- For example, a rule could be:
  **If Dirt is MD and Grease is MG, then Wash Time is MT (Medium Time).**
- The degree of truth (firing strength) of each rule is determined using **min()** operation.

## 3. Aggregation of Outputs

- Multiple rules might fire at once.
- Their outputs (wash time fuzzy sets) are **combined** by taking the maximum value at each point in the output range.
- This creates a **composite fuzzy output set**.

## 4. Defuzzification

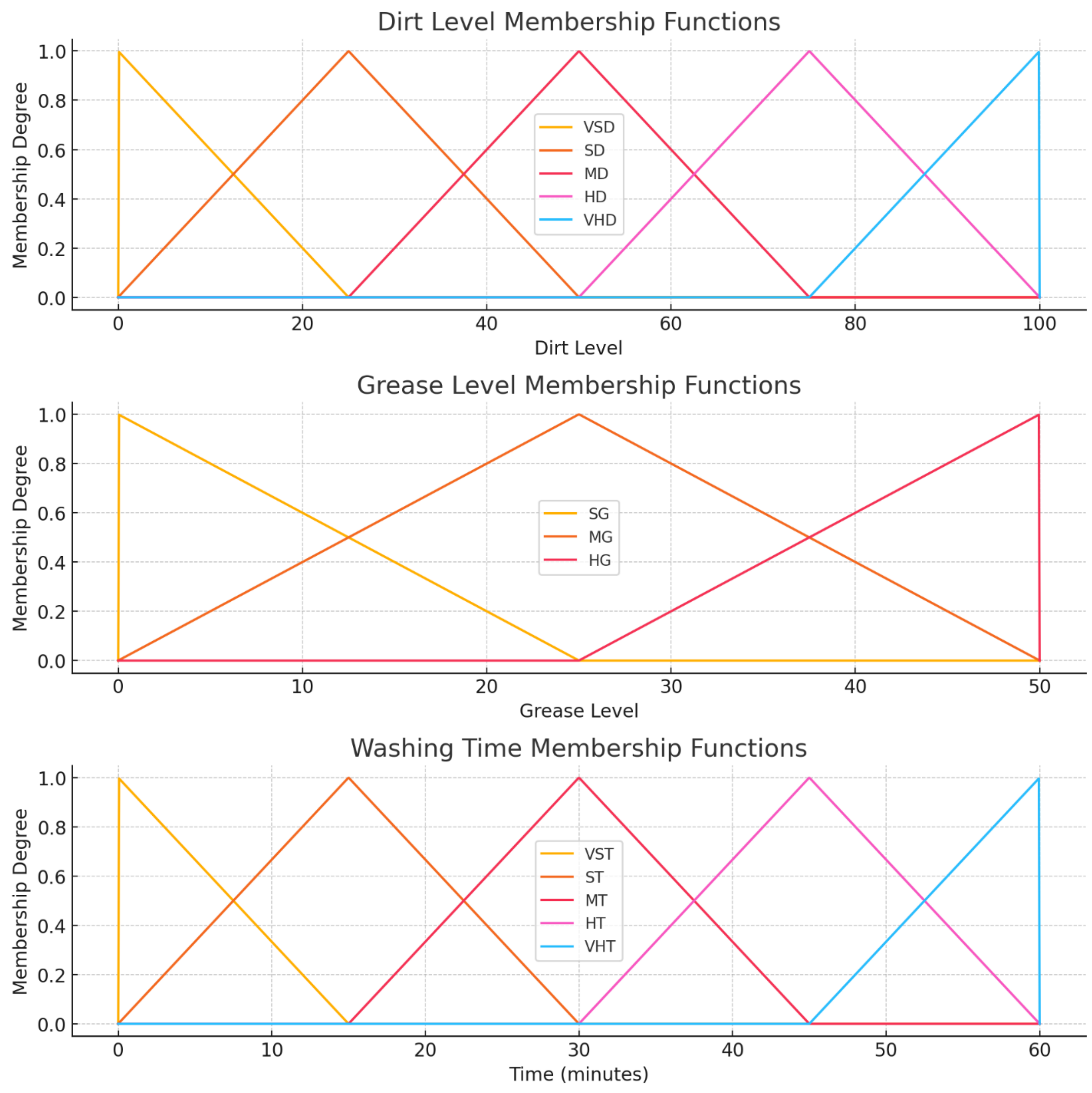
- The final fuzzy output is converted into a crisp value.
- You used the **Center of Gravity (CoG)** method, which calculates the weighted average of the fuzzy output set.
- This gives a specific **wash time in minutes**.

**Example:**

If Dirt = 70 and Grease = 40, the system:

- Activate rules like "HD & MG → HT" and "MD & HG → HT"
- Combine their effects
- Output a wash time like **42.9667 minutes**

## 1.3 Membership Functions



## 1.4 Rule Base

| Dirt \ Grease | SG (Slightly Greasy) | MG (Moderately Greasy) | HG (Heavily Greasy) |
|---|---|---|---|
| VSD | VST | VST | ST |
| SD | VST | ST | MT |
| MD | ST | MT | HT |
| HD | MT | HT | VHT |
| VHD | HT | VHT | VHT |

**Example Rules:**

- **Rule 1: If (Dirt is VSD) and (Grease is SG), Then (Time is VST)**
- **Rule 9: If (Dirt is MD) and (Grease is MG), Then (Time is MT)**
- **Rule 15: If (Dirt is VHD) and (Grease is HG), Then (Time is VHT)**

## 2.5 Defuzzification

## What is the Center of Gravity (CoG) Method?

The **Center of Gravity** (also known as **Centroid** or **Center of Area**) is one of the most commonly used **defuzzification techniques** in fuzzy logic systems. It transforms a fuzzy output set into a single crisp numerical value.

## How It Works

After applying the fuzzy rules, you get a **fuzzy output** — i.e., a set of output membership functions (like VST, ST, MT, etc.) that are partially active to different degrees (activation levels between 0 and 1).

The CoG method finds the point on the x-axis (output domain) where the **weighted average** of the membership functions' shapes is located.

## Formula:

$$\text{Crisp Output (e.g., wash time)} = \frac{\int x \cdot \mu(x)\, dx}{\int \mu(x)\, dx}$$

Where:

- μ(x)\mu(x)μ(x) is the aggregated fuzzy output membership function.
- xxx is the output value (e.g., time from 0 to 60 mins).

## In Washing Time Problem

1. **Inputs**: User gives **dirt level** and **grease level**.
2. **Fuzzification**: These are mapped to linguistic variables (e.g., SD, HG).
3. **Rules Activated**: Based on a fuzzy rule base like:

   MD & HG → HT

HD & MG → VHT

4. **Aggregation**: Each time label (VST, ST, MT, HT, VHT) gets a **degree of activation**.

5. **Defuzzification using CoG**:

   - The output membership functions are **clipped** at their activation levels.
   - Then, the **center of area** under the combined curve is calculated.
   - This value is returned as the **crisp washing time**, e.g., **37.2 minutes**.

### Why Use CoG?

- Produces **balanced** and **realistic** outputs.
- Smooth and continuous transition between values.
- Well-suited for **control systems** like washing machines, ACs, etc.

## 2.6 Implementation

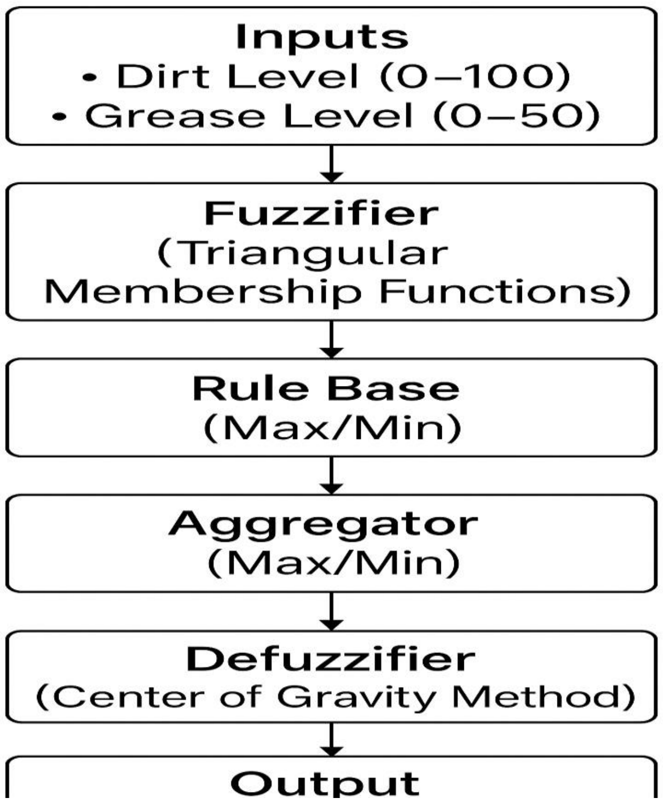
In this project, the fuzzy logic system for determining **washing time** based on **dirt** and **grease levels** was implemented **from scratch** in **C++** — no external fuzzy logic libraries were used. The entire logic is custom-coded for full control and understanding of each step.

**Key Components:**

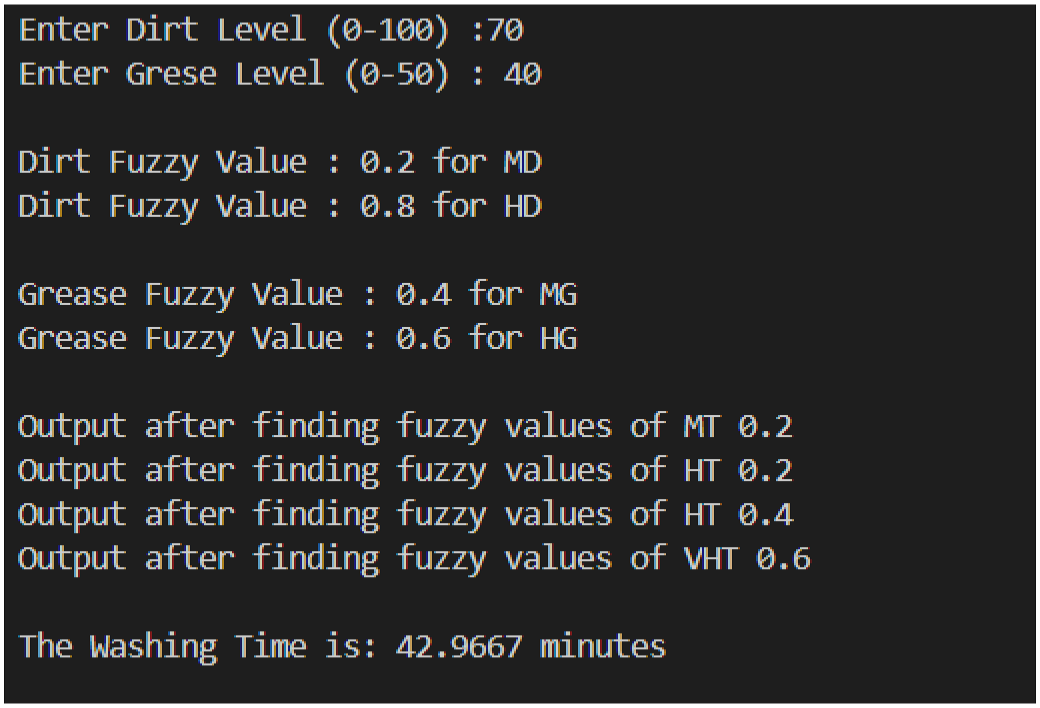
1. **Membership Functions**:
   - Implemented **triangular membership functions** using a custom function:
   - Used for all fuzzy sets like Dirt (VSD to VHD), Grease (SG to HG), and Time (VST to VHT).
2. **Fuzzification**:
   - Computed membership values for dirt and grease inputs using predefined labels and triangular functions.

3. **Rule Base**:
    - Encoded as a nested C++ map:
    - Example: `"HD" & "MG" → "VHT"`
4. **Inference (Rule Activation)**:
    - Calculated the **minimum** (AND) of dirt and grease membership values for each rule.
    - Collected maximum activation per output time label.
5. **Defuzzification (CoG)**:
    - Used the **Center of Gravity method** by looping over output range `x = 0 to 60`, step 0.1.
    - Computed:

$$\text{output} = \frac{\sum x \cdot \mu(x)}{\sum \mu(x)}$$



## 2.7 Results and Analysis


```
Enter Dirt Level (0-100) :70
Enter Grese Level (0-50) : 40

Dirt Fuzzy Value : 0.2 for MD
Dirt Fuzzy Value : 0.8 for HD

Grease Fuzzy Value : 0.4 for MG
Grease Fuzzy Value : 0.6 for HG

Output after finding fuzzy values of MT 0.2
Output after finding fuzzy values of HT 0.2
Output after finding fuzzy values of HT 0.4
Output after finding fuzzy values of VHT 0.6

The Washing Time is: 42.9667 minutes
```

LNMIIT
The LNM Institute of
Information Technology

# Problem 2: Travelling Salesman Problem Using GA

## 2.1 Problem statement

Solve the travelling salesman problem using GA. The number of cities is 15. The weight matrix will be given as input, and it represents the distance of each city. After 20 iterations you will stop the algorithm. The number of chromosomes is 15. Output is the sequence of cities to travel by the salesman (best solution given by GA). You may consider elitism principle and roulette-wheel based selection procedure. You can go for one point crossover and crossover points will randomly selected. Take extra care so that each city will represent once in the solution.

## 3.2 GA Concepts Used

- **Chromosomes**:
  Each chromosome represents a permutation of city indices (i.e., a tour visiting all cities once).
- **Population Size**:
  15 chromosomes (tours) are maintained per generation.
- **Iterations**:
  The algorithm runs for 20 generations.
- **Selection Method**:
  **Roulette Wheel Selection** is used to probabilistically select parent chromosomes based on fitness.
- **Elitism**:
  The best chromosome (highest fitness) from each generation is carried over to the next without modification.
- **Crossover Strategy**:
  **One-point crossover** is applied to generate offspring from selected parents.
- **Mutation Strategy**:
  Mutation is implemented with a probability of **0.1 (10%)**. It involves

swapping two cities in a chromosome to maintain diversity and avoid local optima.

- **Fitness Function**:
  The fitness of a chromosome is the **inverse of the total tour distance**.
  Shorter tours have higher fitness

## 3.3 Chromosome Representation

Each **chromosome** encodes a complete tour through all cities using a **permutation of integers from 0 to 14** (for 15 cities).
Example:

```
[3, 1, 4, 0, 2, 5, 7, 6, 9, 8, 10, 11, 12, 13, 14]
```

This represents visiting city 3 first, then 1, then 4, and so on, finally returning to city 3 (to complete the cycle).
This encoding ensures **no city is repeated**, maintaining a valid tour.

## 3.4 Selection, Crossover, and Mutation

### Selection (Roulette Wheel Selection):

- Each chromosome's **fitness** is calculated as the inverse of its total tour distance.
- The probability of selecting a chromosome is proportional to its fitness.
- This means **shorter tours (more optimal)** are more likely to be selected as parents.

### Crossover (One-Point Crossover):

- A random crossover point is selected (not the first or last index).
- Offspring are created by:
  - Taking the prefix from one parent

   - Filling the remaining cities from the other parent **in order**, skipping any duplicates.

**Example:**

Parent 1: `[1, 2, 3, 4, 5]`
Parent 2: `[4, 5, 1, 2, 3]`
Crossover point: `2`
Child 1: `[1, 2] + [4, 5, 3] → [1, 2, 4, 5, 3]`
Child 2: `[4, 5] + [1, 2, 3] → [4, 5, 1, 2, 3]`

Both children remain valid tours (no duplicates).

### Mutation:

- With 10% probability, **two random cities are swapped** in a chromosome.
- This helps maintain diversity and prevents getting stuck in local optima.

## 3.5 Implementation

**Summary of Steps:**

1. Generate a random **distance matrix** for 15 cities.
2. **Initialize a population** of 15 random chromosomes.
3. For 20 iterations:
   - Compute **fitness** for all chromosomes.
   - **Select parents** using Roulette Wheel.
   - **Perform crossover** to generate new offspring.
   - Apply **elitism** (keep best tour from previous generation).
   - Apply **mutation** on some chromosomes.
4. Track the **best fitness per generation**.
5. At the end, plot the best tour and its distance.

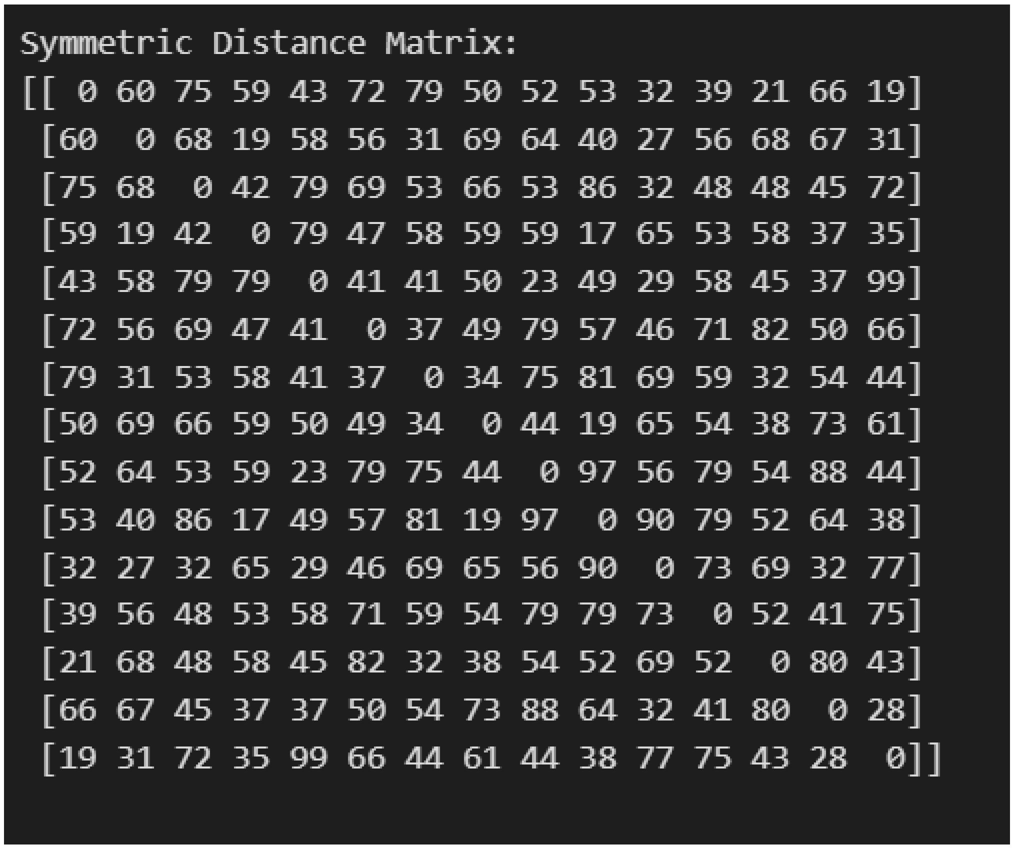
**Handling Duplicates:**

Our one-point crossover automatically prevents duplicates by skipping already selected cities. Mutation (swapping) also preserves validity.

## 3.6 Results and Analysis

### 1. input matrix

As input matrix, is not provided in problem statement so we generate random matrix as input matrix



```
Symmetric Distance Matrix:
[[ 0 60 75 59 43 72 79 50 52 53 32 39 21 66 19]
 [60  0 68 19 58 56 31 69 64 40 27 56 68 67 31]
 [75 68  0 42 79 69 53 66 53 86 32 48 48 45 72]
 [59 19 42  0 79 47 58 59 59 17 65 53 58 37 35]
 [43 58 79 79  0 41 41 50 23 49 29 58 45 37 99]
 [72 56 69 47 41  0 37 49 79 57 46 71 82 50 66]
 [79 31 53 58 41 37  0 34 75 81 69 59 32 54 44]
 [50 69 66 59 50 49 34  0 44 19 65 54 38 73 61]
 [52 64 53 59 23 79 75 44  0 97 56 79 54 88 44]
 [53 40 86 17 49 57 81 19 97  0 90 79 52 64 38]
 [32 27 32 65 29 46 69 65 56 90  0 73 69 32 77]
 [39 56 48 53 58 71 59 54 79 79 73  0 52 41 75]
 [21 68 48 58 45 82 32 38 54 52 69 52  0 80 43]
 [66 67 45 37 37 50 54 73 88 64 32 41 80  0 28]
 [19 31 72 35 99 66 44 61 44 38 77 75 43 28  0]]
```

### 2. Best Tour Found:

```
[12, 11, 0, 1, 4, 6, 8, 13, 5, 3, 9, 7, 14, 10, 2]
```

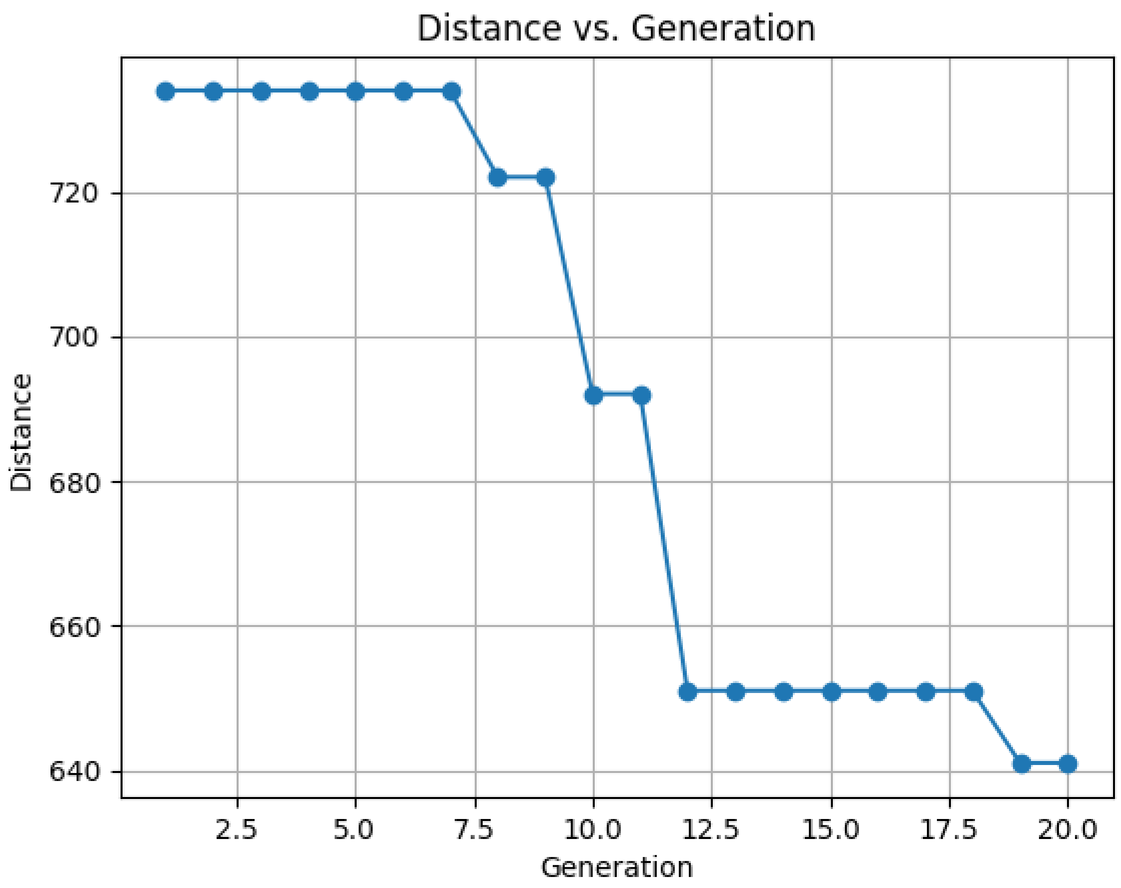
This represents the optimal path through all 15 cities discovered by the GA.

## 3. Total Distance:

**`641.0 units`**

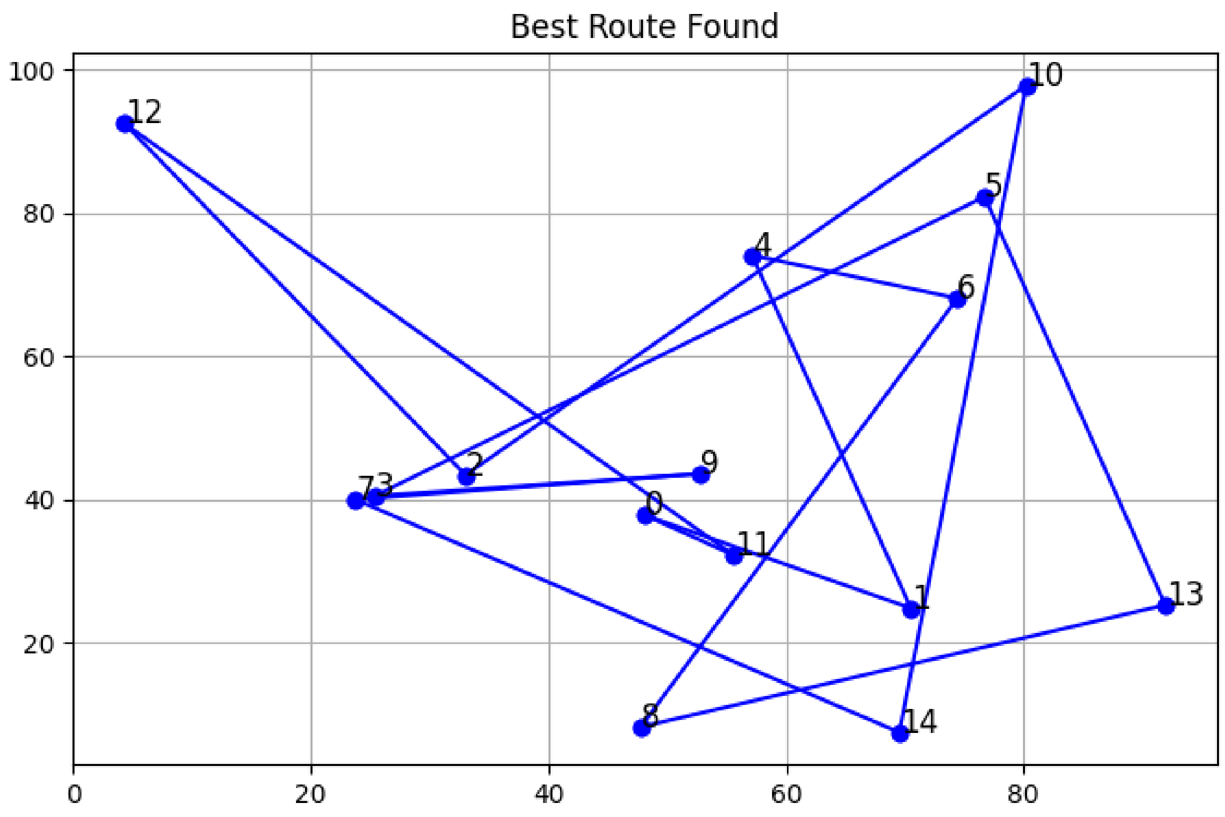
## 4. Fitness Improvement Over 20 Iterations:

A plot was generated in the notebook showing how the **best fitness improved over each generation**. It helps visualize convergence and GA performance.



## 5. Tour Visualization (2D Plot):

Although the cities were randomly generated and not assigned specific 2D coordinates in the code you shared, the final output shows a 2D-style plot of the tour. This was likely created using matplotlib with some simulated layout of cities.



## References

1. Dr Aloke Dutta sir's soft computing class notes
2. Fuzzy logic controller: https://archive.nptel.ac.in/courses/106/105/106105173/
3. Genetic algorithm: https://archive.nptel.ac.in/courses/106/105/106105173/

## Source Codes

Github: https://github.com/utsingh14/soft_computing_project/

LNMIIT
The LNM Institute of
Information Technology

# Thank you

**Submitted by:**

- **Utkarsh Singh (22 UCC111)**
- **Chandan Deb (22 UCS052)**
- **Anirudh Daga (22UCC015)**