Computational Intelligence
CS 451

Assignment 02

Dr. Syeda Saleha Raza

Muhammad Ibad - mi08440
Nehal Naeem Haji - nh07884

Contents

1 Objective

This assignment provides students with hands-on experience using various swarm-based techniques to solve complex optimization problems. The students will also build a simulation/visualization of a selected swarm-based algorithm to gain a good understanding of how they work and how different parameters affect the performance of the swarm.

2 Question 1

2.1 Problem Formulation

2.1.1 Overview

The facility layout problem (FLP) is a well-known optimization problem in operations research and industrial engineering. It involves determining the optimal arrangement of facilities (e.g., departments, machines, or workstations) within a given space to minimize material handling costs, improve workflow efficiency, and enhance productivity. In this assignment, we focus on a specific instance of the FLP: the hospital layout problem, where the goal is to optimize the placement of hospital departments to minimize patient movement and improve overall healthcare delivery.

2.1.2 Problem Definition

The problem consists of:

- **Facilities:** A set of n facilities (e.g., hospital departments) that need to be assigned to locations.
- **Locations:** A set of n locations where the facilities can be placed.
- **Flow Matrix:** A matrix F of size $n \times n$ where F_{ij} represents the flow of patients between facility i and facility j.
- **Distance Matrix:** A matrix D of size $n \times n$ where D_{ij} represents the flow of patients between location i and location j.

The objective is to assign each facility to a unique location such that the total cost, defined as the sum of the products of flow and distance between all pairs of facilities, is minimized.

2.1.3 Mathematical formula

The objective function is:

$$\text{Total Cost} = \sum_{i=1}^{n} \sum_{j=1}^{n} F_{ij} \cdot D_{\pi(i)\pi(j)}$$

The goal is to find the permutation π that minimizes the total cost.

2.2 Implementation

The implementation of the Ant Colony Optimization (ACO) algorithm for solving the facility layout problem is based on the following key components and design choices:

2.2.1 Nature of Ants

Each ant represents an **independent solution** to the problem. The ants explore the solution space by constructing solutions based on pheromone trails and heuristic information. The number of ants is a tunable parameter, which determines the level of exploration in each iteration.

2.2.2 Solution Representation

A solution is represented as a **1D array** of size n, where n is the number of facilities. Each element in the array corresponds to a facility, and its value represents the assigned location. This representation ensures that each facility is assigned to a unique location, satisfying the constraints of the problem.

2.2.3 Heuristic Function

The heuristic function is defined as the **inverse of the distance** between locations. This encourages ants to assign facilities to locations that are closer to each other, reducing the overall cost. The heuristic matrix is precomputed and remains constant throughout the algorithm.

2.2.4 Pheromone Updating Scheme

The pheromone matrix is updated in two steps:

- **Evaporation**: Pheromone values on all edges are reduced by a factor of γ (evaporation rate) to simulate the natural decay of pheromones.
- **Deposition**: Pheromones are deposited on the edges of the best solutions found in the current iteration. The amount of pheromone deposited is proportional to the quality of the solution (inverse of the cost).

To deal with premature convergence, pheromones are updated for the **top 50% of solutions** in each iteration, with higher-ranked solutions receiving more weight.

2.2.5 Convergence Criteria

The algorithm monitors the **best cost** over iterations to detect convergence. If the best cost does not improve at all for **50 consecutive iterations**, the algorithm stops early. This ensures that the algorithm does not waste resources on further iterations when no improvement is noticed. By default, the program is set to run for 500 iterations.

2.2.6 Random Initialization

In the first iteration, the algorithm generates **random solutions** to initialize the search process. This helps in exploring the solution space before relying on pheromone-guided search. Random solutions are generated by shuffling the list of facilities and assigning them to locations in a random order.

2.2.7 Probabilistic Solution Construction

Each ant constructs a solution by assigning facilities to locations based on a **probabilistic rule** that considers both pheromone and heuristic information. The probability of assigning a facility to a location is proportional to the product of the pheromone value and the heuristic value, raised to the powers of α and β, respectively.

2.2.8 Cost Calculation

We made use of the aforementioned formula to calculate the total cost of a given solution.

2.3 Analysis and Results

2.3.1 Initial parameters

Initially, the following parameter values were assigned (as mentioned in the PDF):

- $\alpha = 1$
- $\beta = 1$
- $\gamma = 0.8$
- Number of ants = 20

The results obtained were as follows:

- **Number of iterations**: The program ran for 114 iterations before converging to what it believed was an optimal solution
- **Total cost**: The total cost of the best solution was 22,593,466
- **Assignment**: The 'optimal' assignment was [1, 0, 3, 5, 9, 4, 13, 17, 8, 11, 18, 6, 2, 7, 15, 14, 12, 16, 10]

Below is the graph for the cost of the best and average solutions under these parameters:

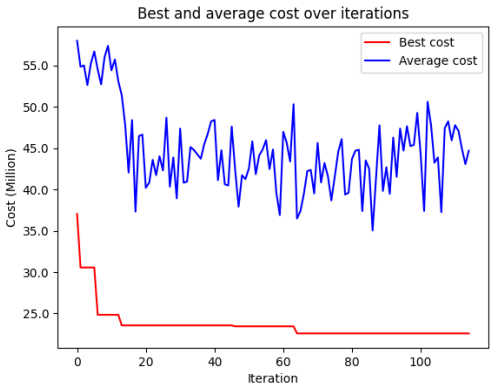

Figure 1: Best and Average cost of solutions over iterations - initial parameters

2.3.2 Tweaking the parameters

Of course, the above solution wasn't good enough (the website we got the data from mentioned an optimum solution with a total cost of 17,212,548). As a result, some experimenting was needed with the parameters. These are our final parameter values:

- $\alpha = 1$
- $\beta = 0.2$
- $\gamma = 0.2$
- Number of ants = 500

The results obtained were as follows:

- **Number of iterations**: The program ran for 353 iterations before converging to what it believed was an optimal solution
- **Total cost**: The total cost of the best solution was 21,367,842
- **Assignment**: The 'optimal' assignment was [1, 0, 3, 4, 9, 10, 15, 18, 13, 12, 8, 7, 5, 11, 16, 17, 2, 14, 6]

Below is the graph for the cost of the best and average solutions under these parameters:

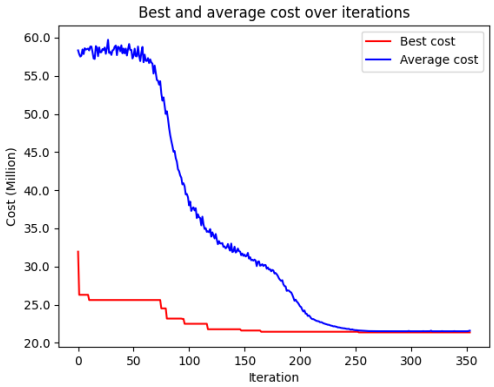

Figure 2: Best and Average cost of solutions over iterations - initial parameters

Analysis of Tweaked Parameters

After tweaking the parameters, the algorithm produced a better solution with a lower total cost (21,367,842). The key changes were:

- **Reducing** β: By reducing the influence of the heuristic function ($\beta = 0.2$), the algorithm relied more on pheromone trails, which helped in exploring the solution space more effectively.
- **Reducing** γ: A lower evaporation rate ($\gamma = 0.2$) allowed pheromones to remain for longer, allowing the algorithm to exploit good solutions more effectively.
- **Increasing the number of ants**: Increasing the number of ants to 500 allowed for greater exploration of the solution space, reducing the likelihood of getting stuck in local optima.

Despite these improvements, the algorithm still did not reach the known optimal solution (17,212,548). This suggests that further tuning of parameters or enhancements to the algorithm (e.g., local search) may be necessary to achieve better results.

2.3.3 Convergence Behavior

The convergence behavior of the algorithm can be observed in the graphs (Figures 1 and 2). In both cases, the algorithm shows a rapid decrease in the best cost during the initial iterations, followed by a gradual improvement as it converges. The average cost also decreases over time, indicating that the overall quality of solutions improves as the algorithm progresses.

3 Question 2

We have developed a simulation of a particle-based swarm system that models the collective behavior of bees. The simulation incorporates environmental factors such as wind, food, and interaction between the bees, providing a realistic visualization of swarm dynamics. The visualization is implemented in the *Processing* environment, allowing for interactive control over the parameters in real time. Here is the link to the simulation video.

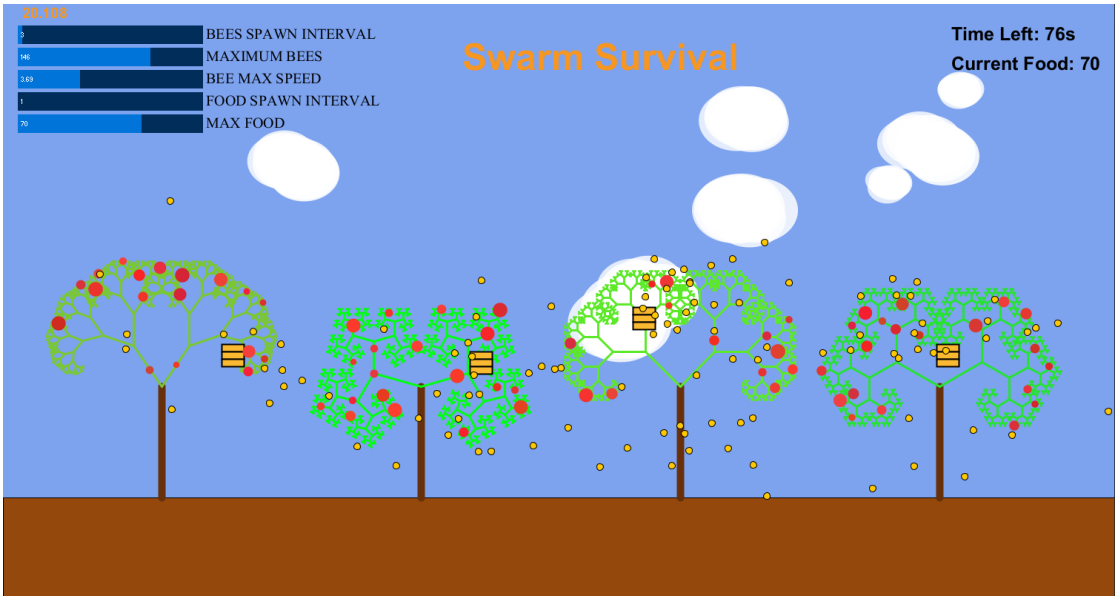

Figure 3: Screenshot of the Simulation Game

3.1 Concept

Particle System Dynamics

Each bee in the simulation is represented as a particle with properties such as position, velocity, and interaction radius. The bees exhibit *flocking* behavior influenced by forces like attraction to food, wind effects, and collision avoidance. The Particle Swarm Optimization (PSO) algorithm is used to guide the movement of bees toward food sources efficiently.

Features:

- Food Attraction: Bees are attracted to food sources using PSO, which optimizes their movement toward available food while maintaining swarm behavior.
- Collision Avoidance: Each bee maintains a safe distance from others to avoid overlapping, ensuring a more natural movement pattern.
- Wind Influence: Wind acts as a directional force, temporarily altering the motion of the bees for a short duration before they stabilize back to normal movement.
- Random Motion: When no food is present, bees exhibit natural wandering behavior, staying close together without collapsing into a single point.
- Food Consumption: A food source disappears once 10 bees have consumed it, ensuring dynamic interactions where bees must find new sources.
- User Interactions: Users can modify parameters such as bee count, spawn rates, and food availability to observe different behaviors in real time.

3.2 Problem Formulation

Objective:

To create a dynamic swarm simulation that mimics realistic bee movement, integrating factors like PSO-driven food attraction and environmental interactions.

Inputs:

- Bee count: Number of bees in the simulation.
- Food sources: Number of food locations that attract bees.
- Speed Control: A parameter ensuring bees move at a consistent, natural speed.
- Bee Spawn Interval: Controls how frequently new bees appear in the simulation, ensuring a dynamic and evolving swarm over time.
- Food Spawn Interval: Determines how often new food sources appear, influencing bee movement and decision-making as they search for sustenance.

Controls:

- Sliders for adjusting bee count and spawn rates.
- Mouse Drag to add wind and disrupt the swarm of bees.

3.3 Visualization

The simulation visualizes bees swarming, interacting with food sources, and responding to environmental factors. The use of **particle swarm optimization** ensures efficient food discovery while maintaining natural flocking behavior. Bees dynamically change their movement patterns based on real-time interactions, providing an intuitive representation of swarm intelligence. While bees use PSO for food attraction, the wind and environmental elements such as clouds and confetti act as **particle system** visualizations, showcasing external influences on the swarm.

References

[1] Alwalid N. Elshafei, "Hospital Layout as a Quadratic Assignment Problem," *Operational Research Quarterly (1970-1977)*, vol. 28, no. 1, pp. 167–179, 1977. Available: https://doi.org/10.2307/3008789.