Intelligent systems
Assignment 1: Graph Pathfinding with Genetic Algorithms

October 12, 2025

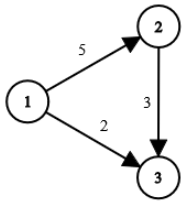
1 Introduction

In this seminar assignment, your goal is to use genetic algorithms to find the shortest path between two points in a directed graph. You will be tasked with running suitable genetic algorithms, preparing the data to test your algorithms and writing a report that will show your work and present the results in a clear and concise manner.

2 Task 1 - Shortest path between 2 nodes (15%)

A common way to represent graphs is by first specifying the number of points (nodes) and then listing the connections (edges) between the nodes in the form of $Node_A, Node_B, Distance$. For example, a simple graph can be represented as:

```
3           # Number of nodes
1 2 5       # Edge between nodes 1 and 2 with distance 5
1 3 2       # Edge between nodes 1 and 3 with distance 2
2 3 3       # Edge between nodes 2 and 3 with distance 3
```



Use this format to create several example graphs (ranging from small to large) that you will use to test the subsequent tasks.

The goal of the first task is to find the shortest distance between two nodes. In addition to the graph itself, the input data should also include the start and end nodes. Choose and implement a suitable fitness function and run a genetic algorithm to find the shortest path in a given graph provided as a .txt file.

Experiment with different mutation/crossover functions and other GA parameters (population size, selection, etc.) and try to find the optimal configuration of the algorithm.

3 Task 2 - Shortest path between multiple nodes (20%)

The goal of the second task is to update the genetic algorithm from Task 2 with the ability to follow a path consisting of multiple nodes. Rather than simply finding the shortest path between nodes A and B, your genetic algorithm should receive a sequence of N target nodes $A, B, C, ...N$ and return the shortest path that follows the given node in order. Paths that do not visit every target node or paths that visit the target nodes in the incorrect order are no longer valid.

Note that your solutions may now need to visit a single node multiple times (for example, to backtrack through a graph).

In addition, improve your genetic algorithm using custom crossover and mutation functions [1]). Since the task now contains strict requirements (for example, not visiting the target nodes out of order), you can implement these constraints into crossover and mutation functions and into the starting population.

4 Task 3 - Shortest path with multiple agents (35%)

The goal of the third task is to update the genetic algorithm from Task 2 to work with multiple agents moving through the graph simultaneously. Instead of finding a single path, assume that you have a m different agents that would all like to complete the path in the shortest possible time, Your algorithm should still be given a sequence of target nodes $A, B, C, ...N$ but should now also be given a sequence of starting nodes for agents $S_1, S_2, ...S_m$. Instead of returning a single path, it should return m paths (one for each agent) so that the agents complete the target path in the shortest time.

This task also introduces several new restrictions:

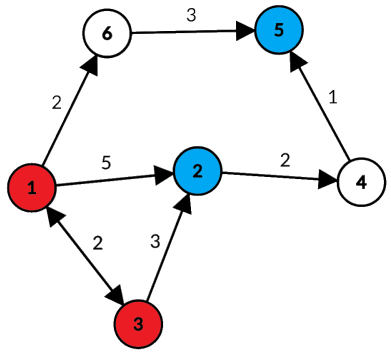
- Moving through an edge with distance d takes d time units.
- After moving, each agent will wait at a node for an additional 10 time units.

[1] User-Defined Crossover, Mutation, and Parent Selection Operators

- Each node and edge on the graph can contain only one agent at a time.
- Instead of moving, agents can wait at a node for 20 time units.

This means that agents will sometimes have to wait at a node instead of moving. In the example below, the path goes from node 2 to 5 and two agents start at nodes 1 and 3. For both agents, the shortest path is $[2, 4, 5]$, but if they both followed this path simultaneously, they would collide at node 2. Instead, one agent has to wait at the starting node. The correct solution could be represented as:

```
Agent 1: [1, 2, 4, 5]
Agent 2: [3, wait, 2, 4, 5]
```
or
```
Agent 1: [1, 2, 4, 5]
Agent 2: [3, 3, 2, 4, 5]
```



Similarly, the following solution would be invalid because both agents would cross the edge between nodes 1 and 3 at the same time:

```
Agent 1: [1, 3, 2, 4, 5]
Agent 2: [3, 1, 1, 2, 4, 5]
```

As in Tasks 2 and 3, experiment with different (custom) crossover/mutation functions and other GA parameters. Try to enforce the rules of this task using custom crossover mutation and crossover functions. In order to fully test this task, you will likely need to create large graphs with long target paths and many agents.

5 Task 4 - Evaluation and report (30%)

The final task of the assignment is to perform a thorough evaluation of your approach. For each task, run your GA with different parameters, starting populations, and crossover/mutation functions, and compare your results on **multiple** graphs of different sizes

Compile a comprehensive report (jupyter notebook) detailing your approach, showcasing code highlights, and presenting the results. Your report should include a **results** section that compares all of your approaches. Make sure to include:

- Tables and graphs comparing the performance of different genetic algorithms.
- Tables and graphs comparing the effect of various parameters and mutation/crossover functions on the results.
- Tables and graphs comparing the effect of different graph sizes.
- Discussion of results. Which approaches worked and which did not?

Make sure that the graphs are readable and have appropriate titles and labels.

6 Submission

- **Deadline: 30. 11. 2025**
- **Format:** Jupyter Notebook
- **Group Work:** Maximum of two people per group.
- **Use of Generative AI:** Using generative language models to initiate the solution is permitted. Please attach the conversation as a single `.txt` file along with your Jupyter Notebook.