LARGE ROUTE OPTIMIZATION

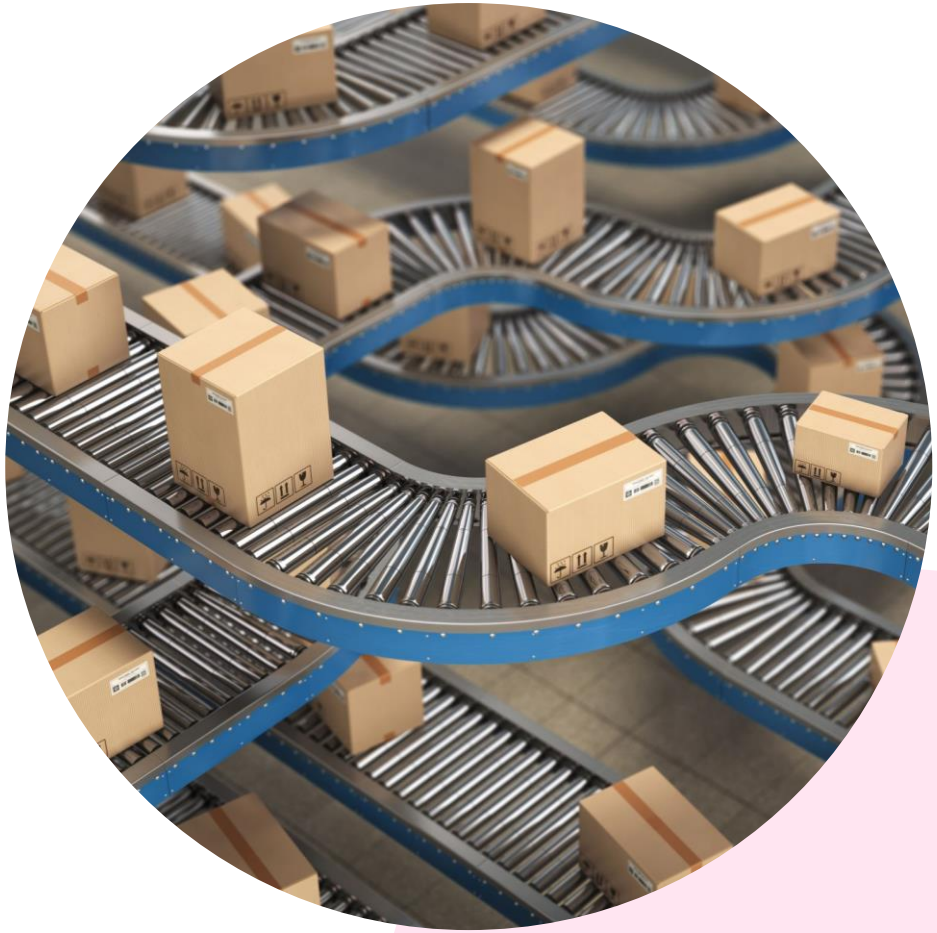
GENETIC ALGORITHM APPROACH

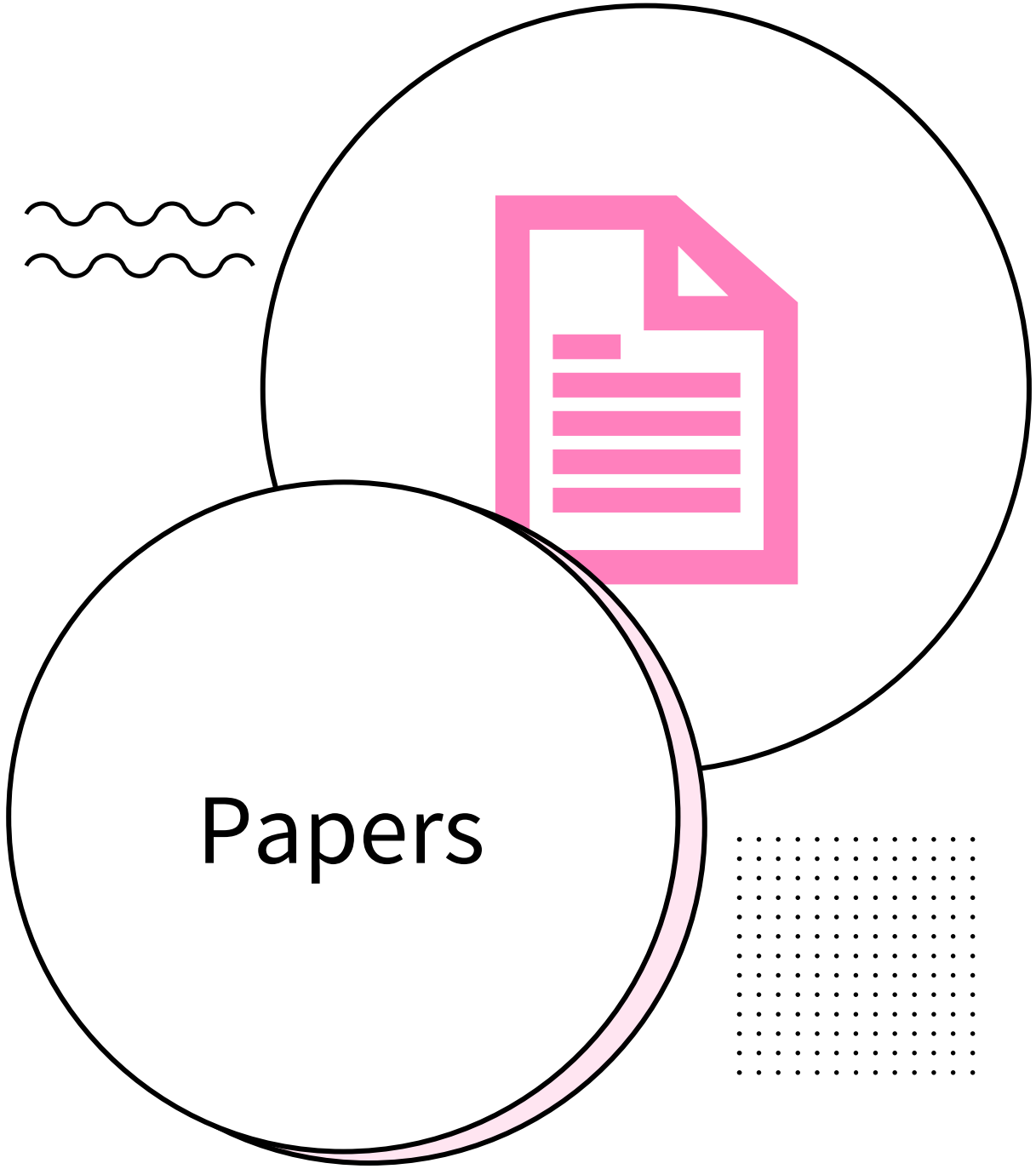
Agenda

- Project Overview
- Papers
- Datasets
- Algorithm Implementation
- User Interface

Project Overview

The main idea of the project is to assign orders to different trucks and then find the best possible route that minimizes the overall delivery cost of some provided constrains e.g. least distance and orders should arrive before deadline





- (PDF) Route planning by evolutionary computing: an approach based on genetic algorithms (researchgate.net)
- Electronics | Free Full-Text | Selected Genetic Algorithms for Vehicle Routing Problem Solving (mdpi.com)
- Dynamic vehicle routing using genetic algorithms | Applied Intelligence (springer.com)
- Optimization of Multiple Traveling Salesmen Problem by a Novel Representation Based Genetic Algorithm | SpringerLink
- Solving the Vehicle Routing Problem using Genetic Algorithm (thesai.org)
- Vehicle Routing Problem Using Genetic Algorithm with Multi Compartment on Vegetable Distribution - IOPscience

Datasets Used

Order_ID	Material_ID	Item_ID	Source	Destination	Available_Time	Deadline	Danger_Type	Area (m^2)	Weight (kg)
A140109	B-6128	P01-79c46a02-e12f-41c4-9ec9-25e48597ebfe	City_61	City_54	2022-04-05 23:59:59	2022-04-11 23:59:59	type_1	3.888	3092
A140112	B-6128	P01-84ac394c-9f34-48e7-bd15-76f92120b624	City_61	City_54	2022-04-07 23:59:59	2022-04-13 23:59:59	type_1	3.888	3092
A140112	B-6128	P01-b70c94db-630a-497b-bb63-b0ad86a7dce6	City_61	City_54	2022-04-07 23:59:59	2022-04-13 23:59:59	type_1	3.888	3092

Datasets Used

Truck Type (length in m)	Inner Size (m^2)	Weight Capacity (kg)	Cost Per KM	Speed (km/h)
16.5	16.1×2.5	10000	3	40
12.5	12.1×2.5	5000	2	40
9.6	9.1×2.3	2000	1	40

Datasets Used

Source	Destination	Distance(M)
City_24	City_47	1114251
City_24	City_31	97187
City_24	City_54	1716028
City_24	City_53	1729925
City_24	City_19	1594107
City_24	City_12	774894

Algorithm Implementation

Steps taken

01
Pre-Processing the data to gain information from it that can be used in the project

02
Identify how this data can be represented to feed it into the algorithm

03
Identify the constraints

04
Find a formula to calculate the fitness of an individual

05
Implementing a function to randomly initialize a population

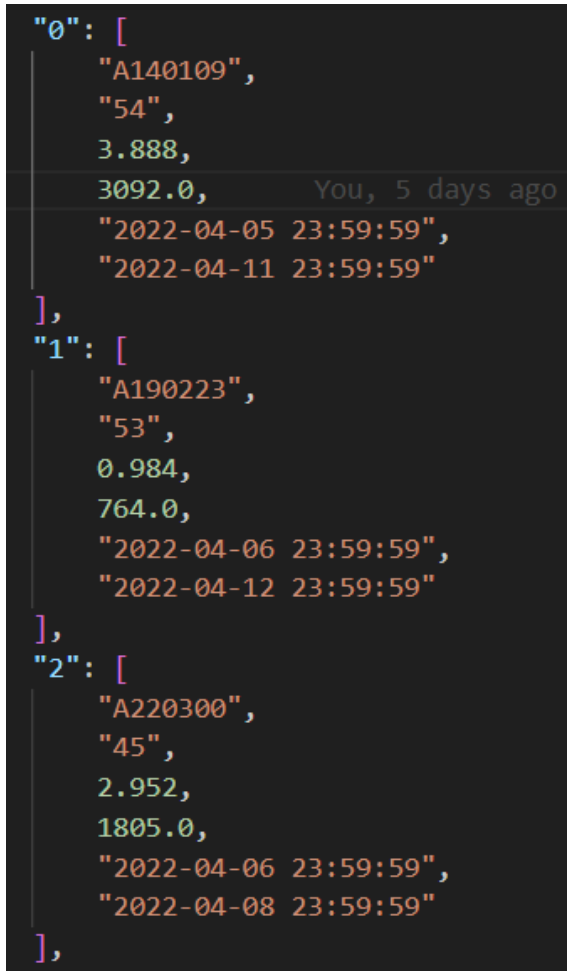
06
Choosing suitable crossover, mutation and selection functions for the representation

07
Implementing a survivor mechanism

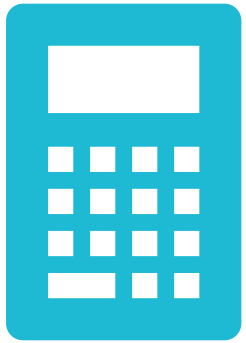
08
Combining all the functions into the genetic algorithm

Representation Used

- The representation used is a combined representation
- Each number here represents a different order with all information needed about that order
- A single chromosome is a unique list of these numbers



Fitness Function



As this project is classified as a Multi-Objective Optimization, the fitness formula used is a weighted summation of all constraint values we have.

The objective here is to minimize this value, this indicates a fit individual

Initialization Function

- This function is used to initialize the first population randomly

```
def initialize_population(population_size, candidate_len):
    for _ in range(population_size):
        truck = Truck(max_stops=max_stops)
        candidate = Individual(candidate_len).individual
        fit = Fitness(candidate, truck)
        fit.get_fitness()
        population.append(candidate)
        fitness_values.append(fit.fitness)
```

Crossover, Mutation and Selection

Crossover Functions Used:

- Edge Crossover
- Order Crossover

Mutation Functions Used:

- Inversion
- Insertion
- Scramble
- Swap
- Random Resetting

Parent Selection Functions Used:

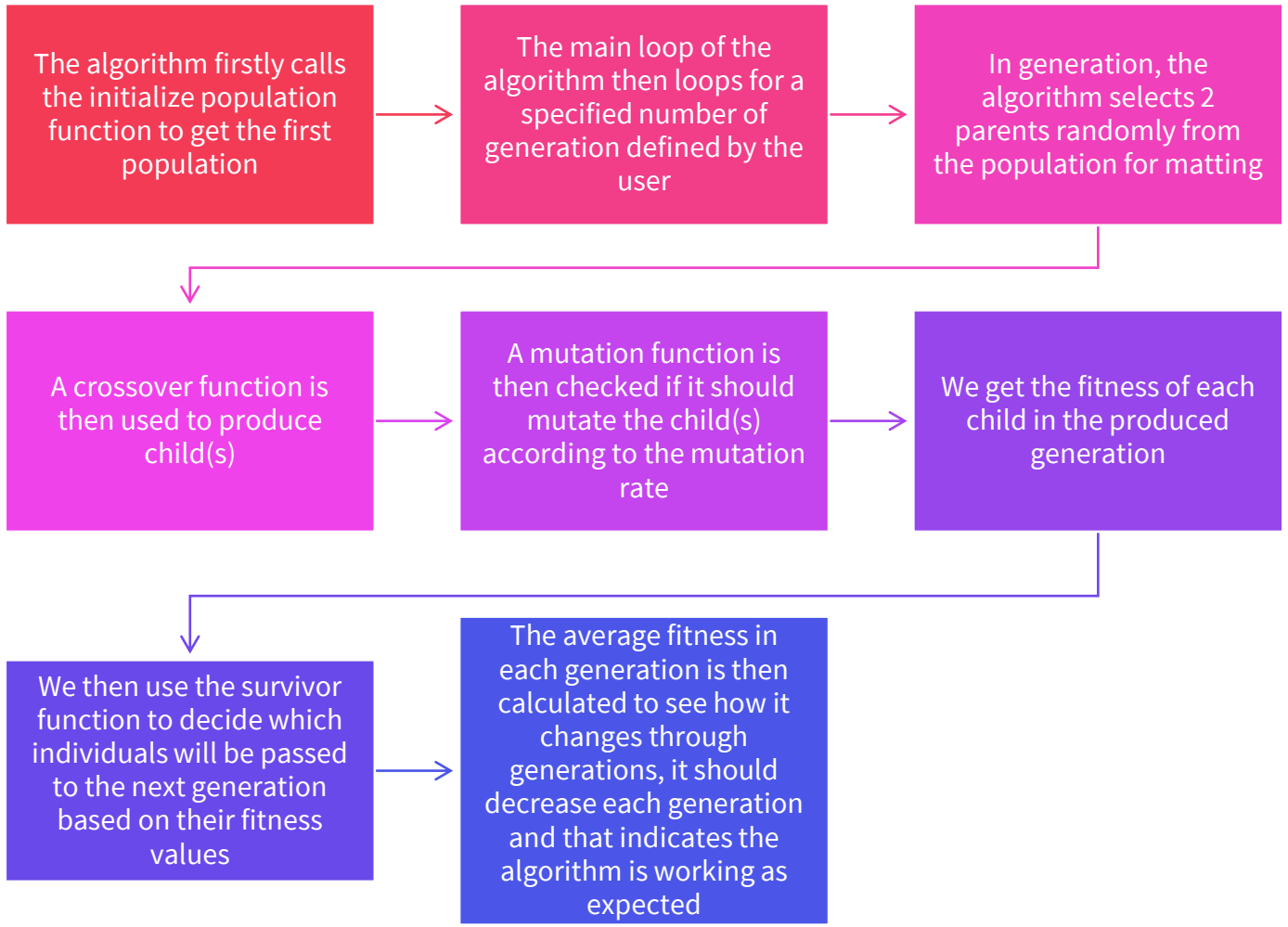
- Tournament Selection
- Exponential Rank Based Selection

Survivor Mechanism

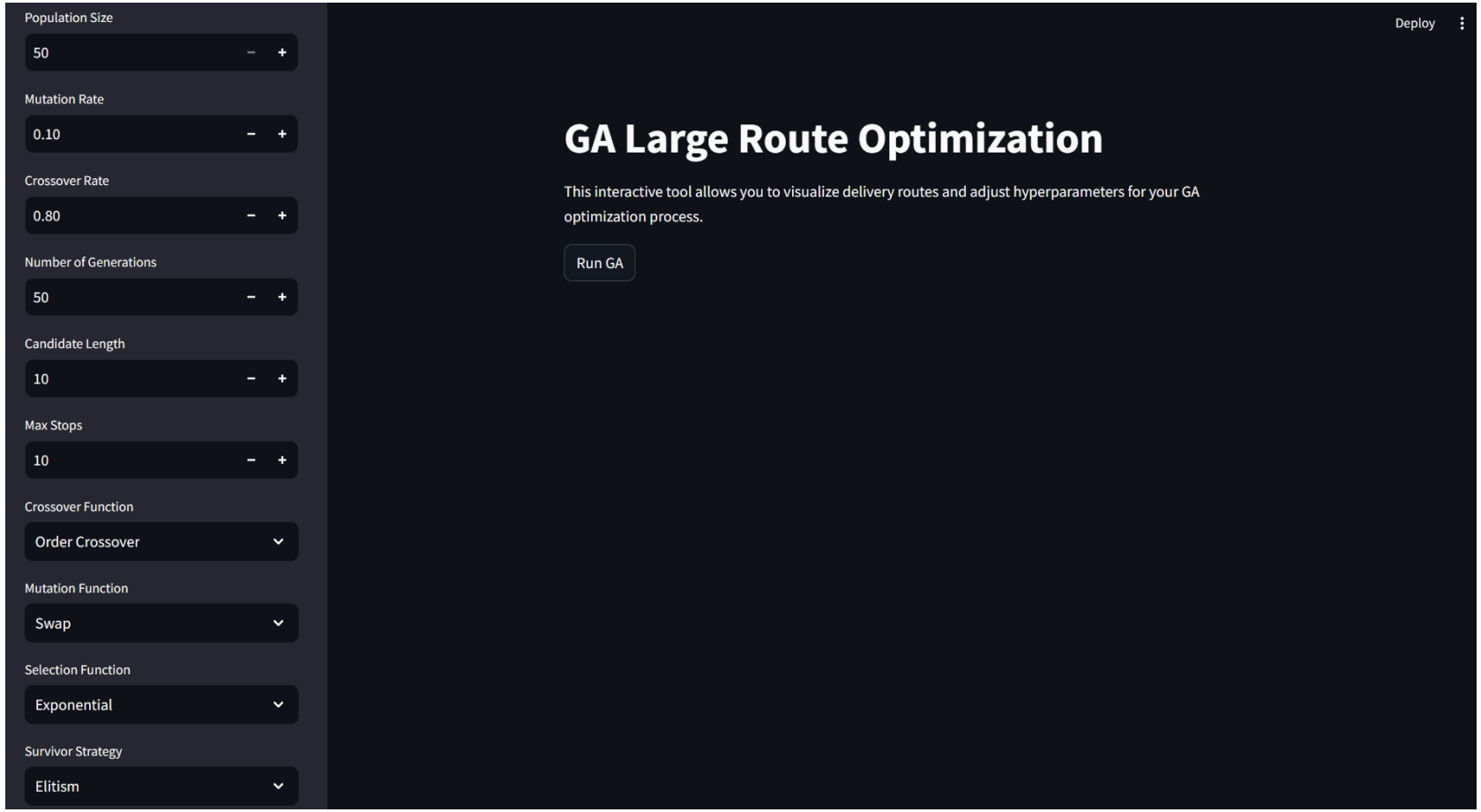
Survivor mechanisms are used to decide which individuals should be passed to the next generation, usually the most fit individuals are passed to the next generation

We use the Elitism survivor function, it returns the top 30 individuals in a population

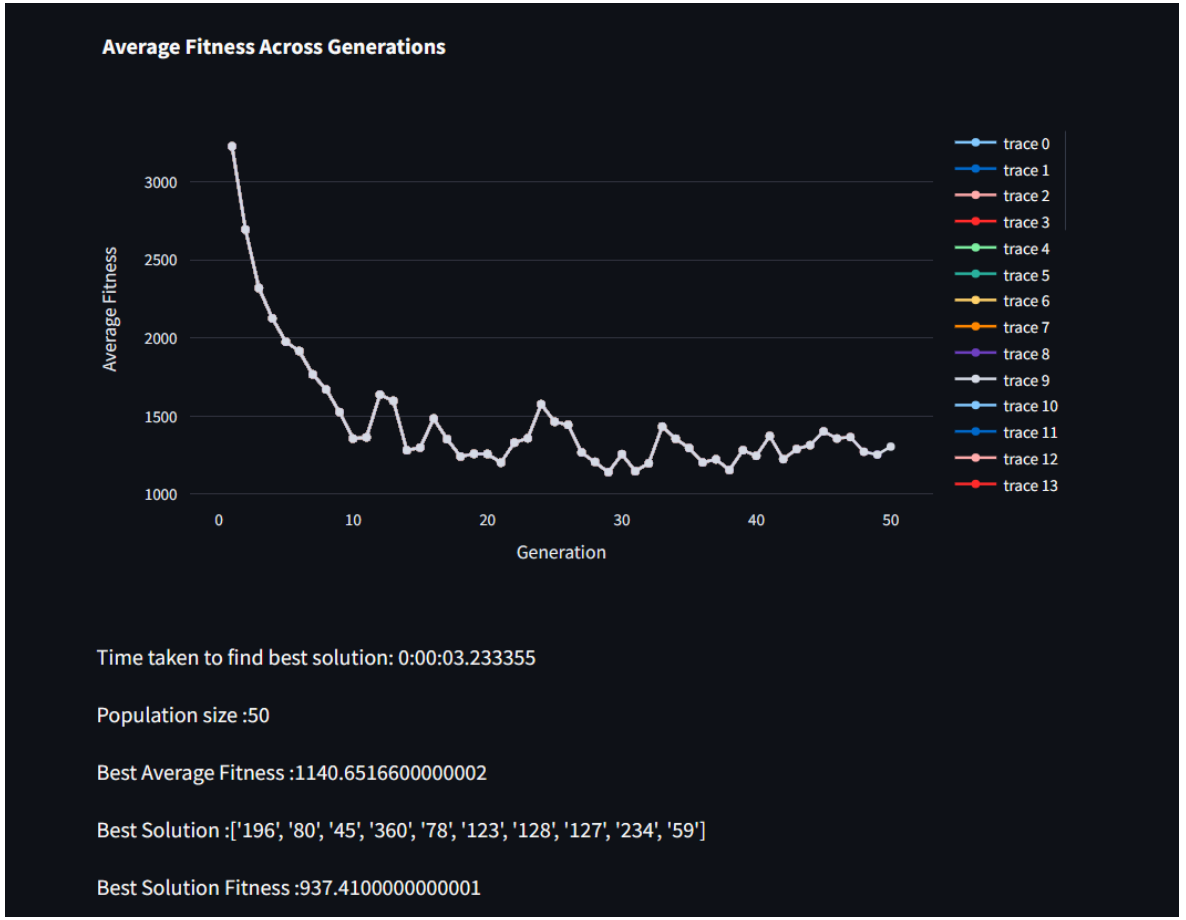
Genetic Algorithm



User Interface



User Interface

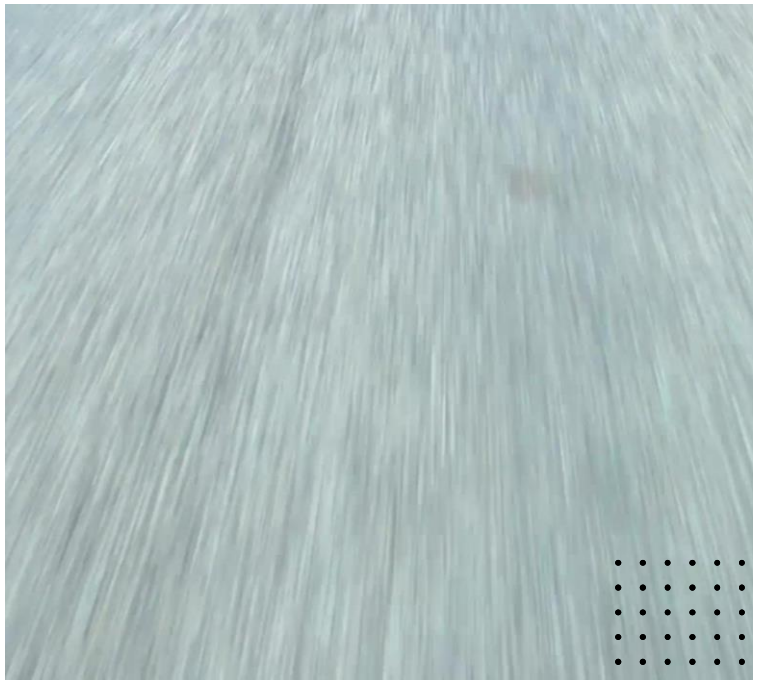


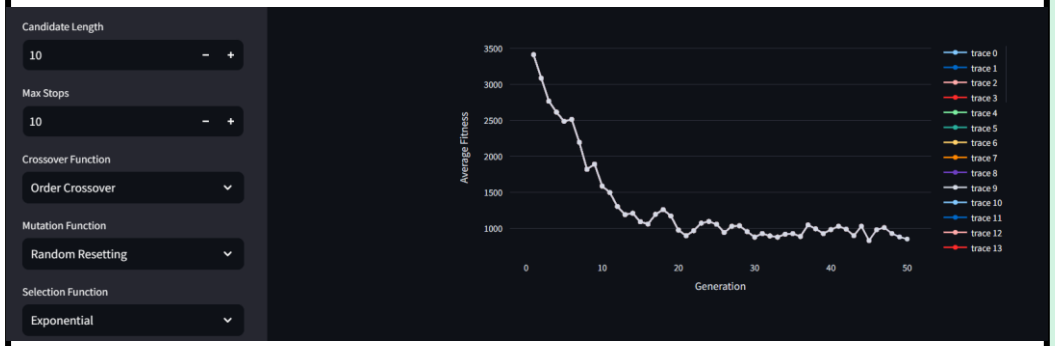
Testing Different configurations

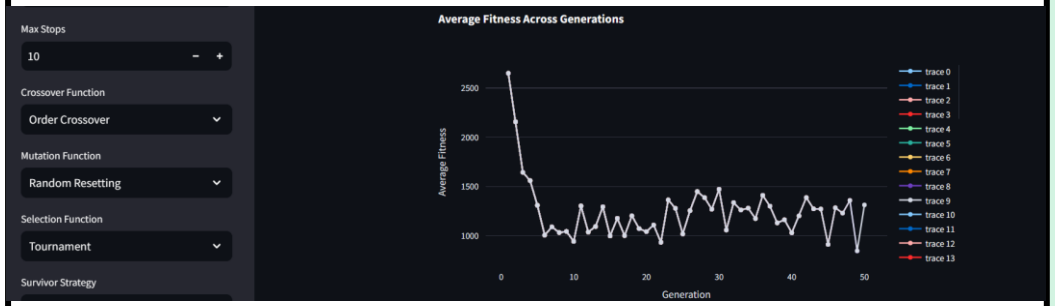
- The average time taken by the algorithm if we are using Order Crossover is about 2.8 seconds which is faster than the Edge Crossover by 0.6 less seconds when using the same configurations

Testing Different configurations

- The Random Resetting Mutation allows more diverse population and leads to finding the best solution faster than other mutation functions







Testing Different configurations

- The first photo is the curve of the average fitness when using the Exponential Rank Based Selection
- The second photo is the same curve but with Tournament Selection
- We can see that Exponential is more consistent than Tournament Selection as the curve is smoother

THANKS