Report

**Part 1**

The algorithm for solving the TSP problem using a genetic algorithm starts by creating an initial population of randomly selected cities as chromosomes. The size of the initial population is determined by the number of chromosomes and should be larger for larger chromosomes to increase accuracy. Each genome in the initial population has a fitness value, which indicates its strength.

The next generation is created by selecting two parent genomes from the initial population using a randomized tournament selection approach. The algorithm takes 20 random samples from the initial population and returns the fittest genome. The "reproduction" function then chooses random genomes to create a new child using a one-point crossover approach. The algorithm chooses a random crossover point within the chromosomes and the child's chromosomes are inherited from the first parent up to the crossover point. The remaining chromosomes are inherited from the second parent in a loop. The child's chromosome is then evaluated using the "eval chromosome" function to determine its fitness value.

Like natural processes, mutation occurs randomly in the algorithm with a mutation rate of 0.2, though the maximum mutation rate can be set to 0.3. The mutation used is a "scramble mutation" or "single point mutation." The algorithm randomizes the values within a specific range and assigns the new chromosome to the genome. The fitness value of the genome is then re-calculated using the "eval chromosome" function.

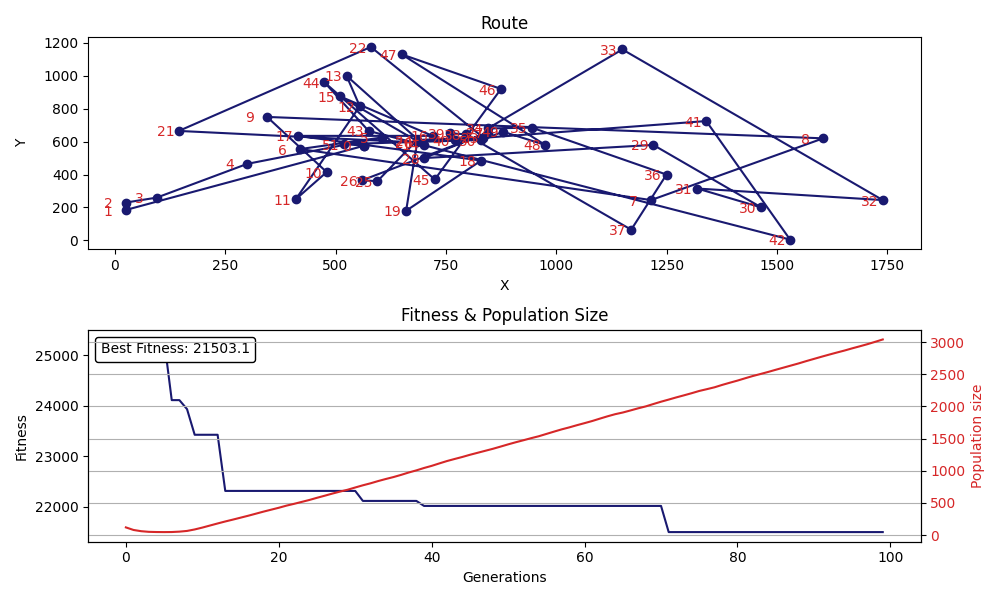
Finally, the new child is added to the child's population for the next generation, completing the process.

**Part 2**

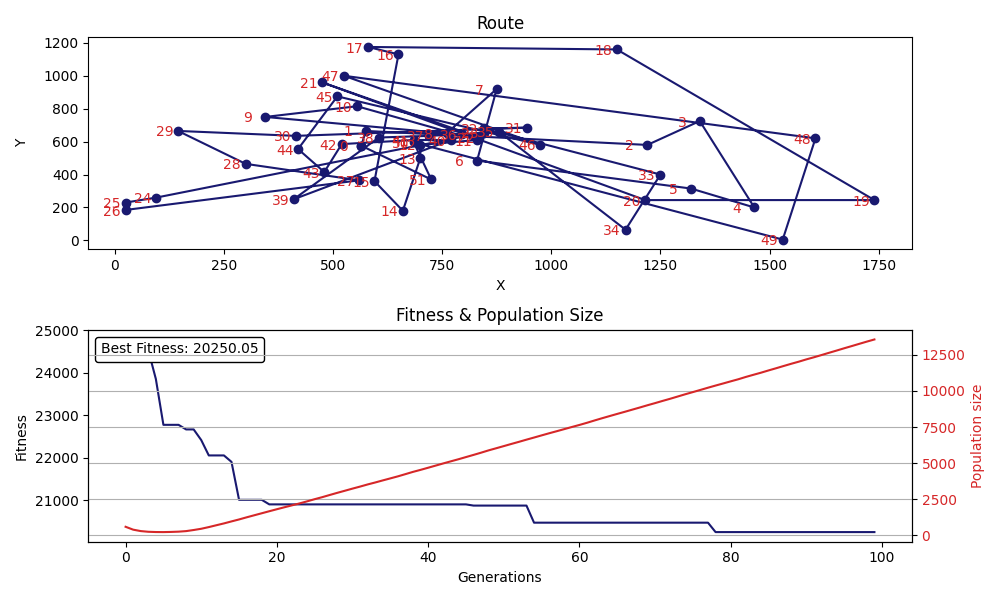
**Size of Initial Population** The size of the initial population in a genetic algorithm is important because it determines the diversity and the exploration of the solution space.

* 1. Diversity: A larger initial population size provides more genetic diversity, which helps the algorithm explore a wider range of solutions and avoid getting stuck in a suboptimal local minimum.
  2. Exploration: A larger initial population size allows for a broader exploration of the solution space, which can increase the chances of finding a global optimal solution.
  3. Computational Cost: However, a larger initial population size also increases the computational cost, as more individuals need to be evaluated and processed in each iteration.

The size of the initial population must be chosen such that it provides enough diversity and exploration, while being computationally feasible. The appropriate size of the initial population depends on the specifics of the problem being solved and the desired outcome and can be determined through experimentation or prior knowledge about the problem.



Initial Population 200 People

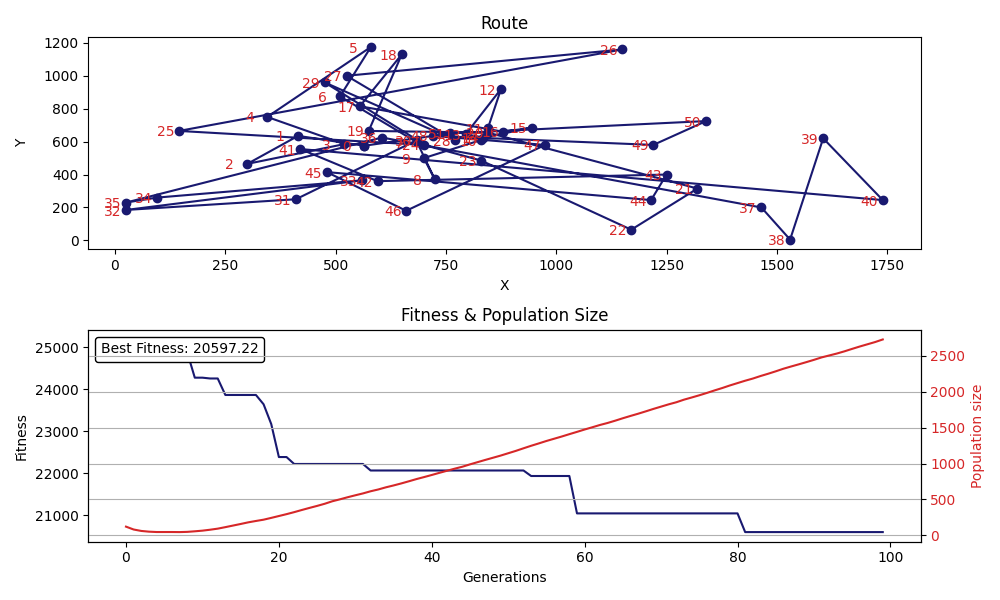


Initial Population 1000 People

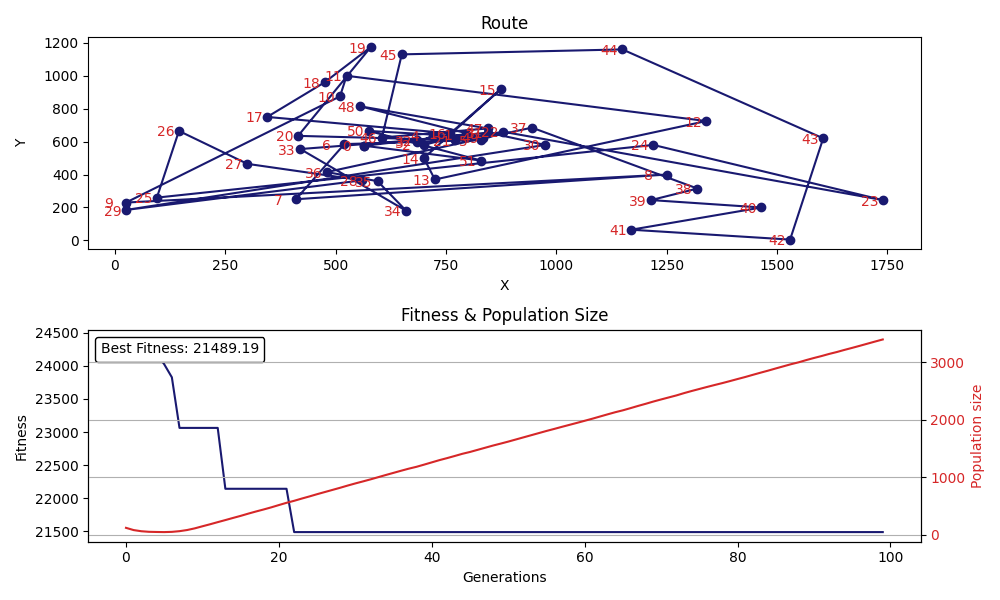
**Mutation Probability:** The mutation probability in a genetic algorithm is important because it determines the rate at which new genetic information is introduced into the population.

1. Exploration: By introducing new genetic information through mutation, the algorithm can explore new parts of the solution space and avoid getting stuck in a suboptimal local minimum.
2. Preservation of Diversity: Mutation helps to preserve genetic diversity in the population, which can improve the chances of finding a global optimal solution.
3. Fine-Tuning: Mutation can also help fine-tune the individuals in the population by making small adjustments to their genetic information.

The mutation rate is a probability value used in genetic algorithms to determine the likelihood that a particular mutation will occur in each generation. In the context of solving the Traveling Salesman Problem (TSP) using a genetic algorithm, the mutation rate is typically set as a small value, such as 0.1 or 0.2, in order to maintain diversity in the population. A higher mutation probability allows for greater exploration and preservation of diversity but can also lead to a loss of genetic information and decreased convergence to an optimal solution. A lower mutation probability allows for more fine-tuning but can limit the exploration of the solution space.



Mutation Rate is 0.2



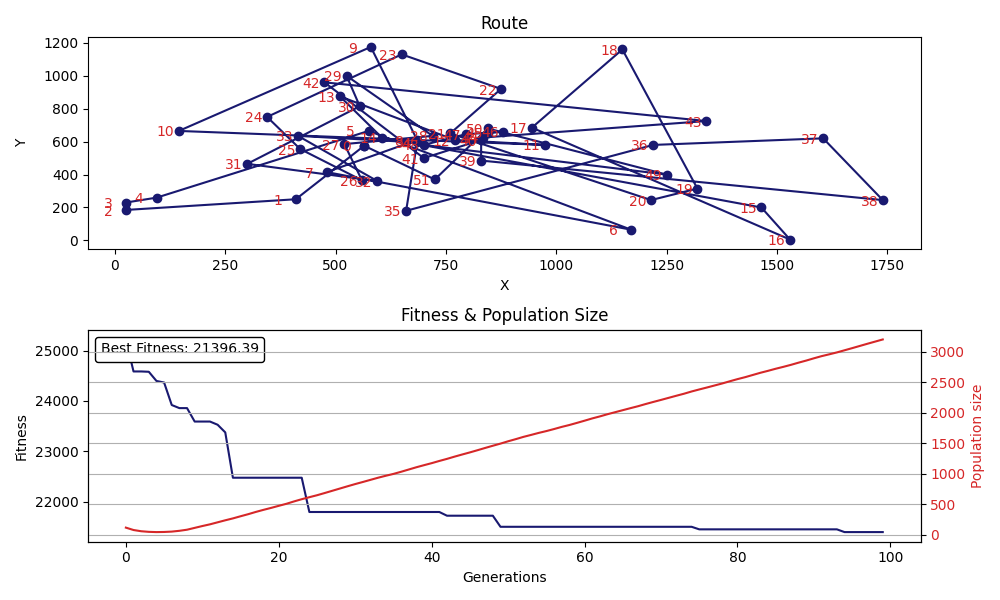
Mutation Rate 0.5

**Selection Parameters:** The tournament size in tournament selection is an important parameter because it determines the degree of exploration vs exploitation in the algorithm.

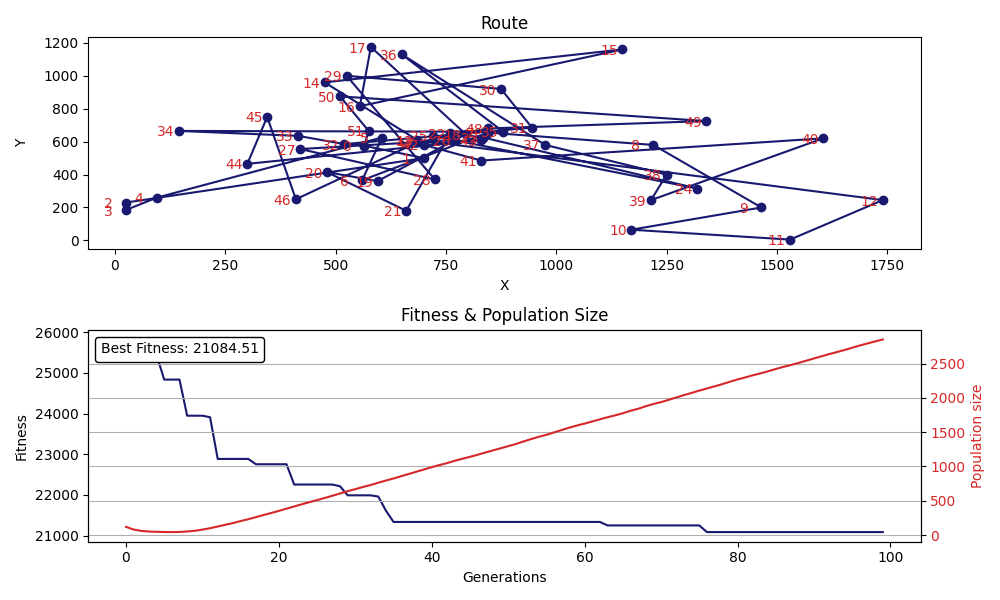
A larger tournament size increases the exploration of the solution space, as it provides a larger pool of individuals from which to select the fittest. This can help prevent premature convergence to a suboptimal solution.

However, a smaller tournament size increases the exploitation of the algorithm, as it focuses more on the individuals with the highest fitness values. This can help the algorithm converge more quickly to an optimal solution but can also lead to loss of genetic diversity and decreased exploration of the solution space.

The choice of tournament size is a trade-off between exploration and exploitation, and the appropriate value depends on the specifics of the problem being solved and the desired outcome. An appropriate value for the tournament size must be found through experimentation or prior knowledge about the problem.



Selection Parameter is 20



Selection Parameter 30

**Part 3**

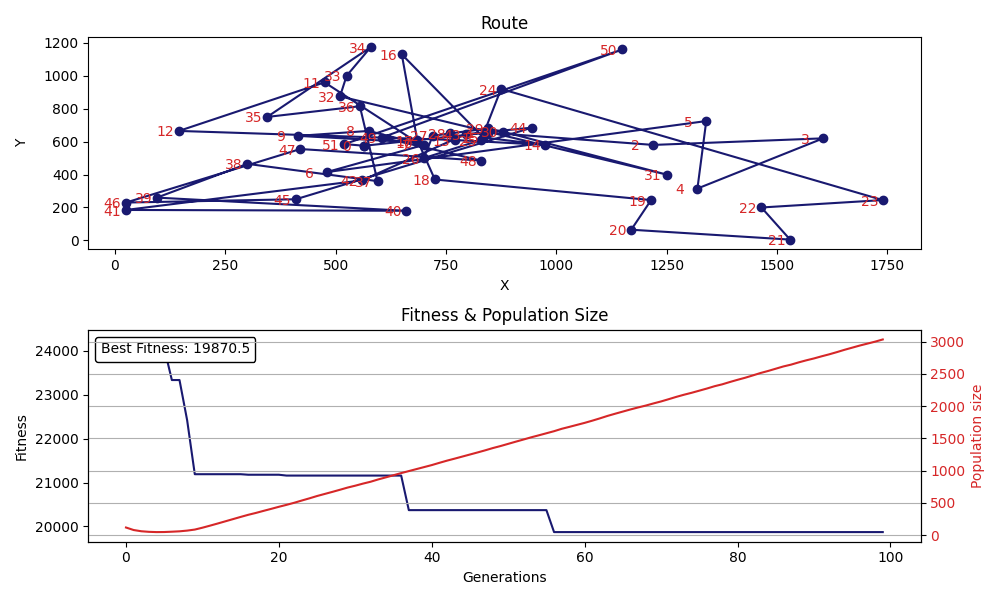
Pass

**Part 4**

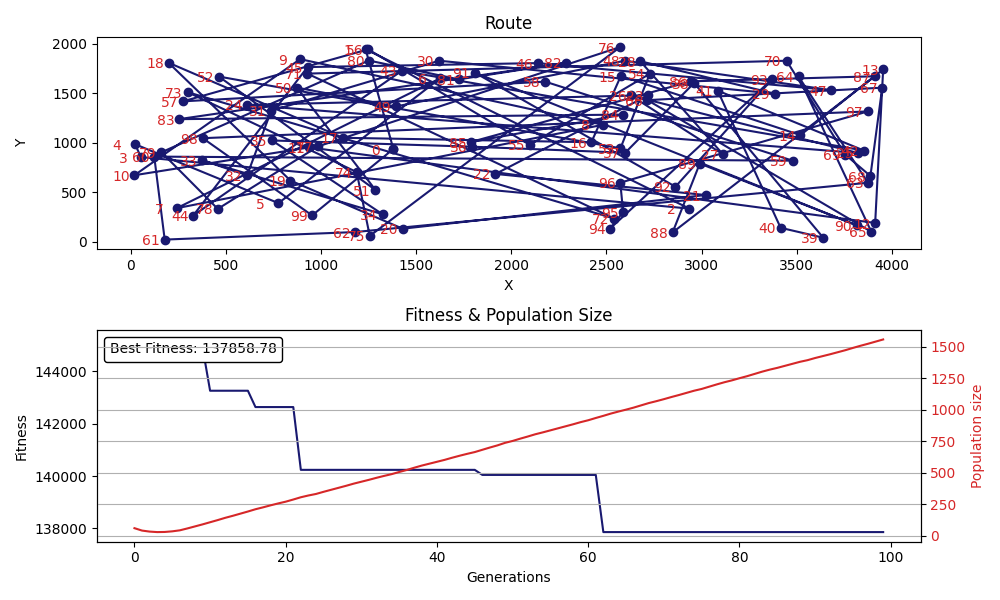
Chart, line chart

Description automatically generated

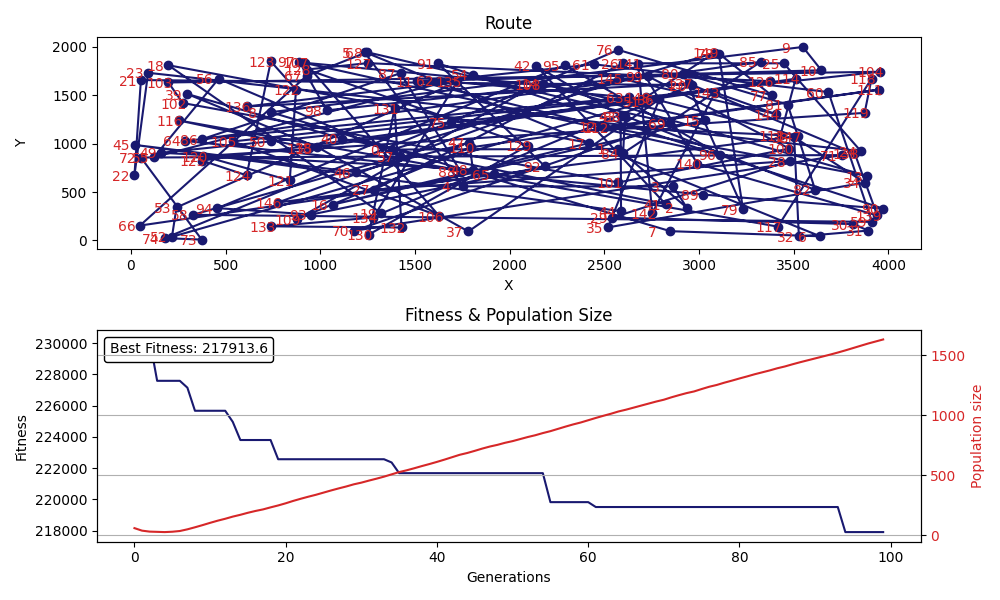
Berlin11.tsp



Berlin52.tsp



kroA100.tsp

kroA150.tsp

**Part 4**

In conclusion, the use of Genetic Algorithm to solve the Traveling Salesman Problem has proven to be an efficient method to find near-optimal solutions for large problem sizes. It mimics the natural process of evolution to generate a population of solutions and gradually improve upon them through generations, allowing for a more comprehensive search of the solution space. The GA approach has been shown to perform well in comparison to other optimization methods, such as brute force and greedy algorithms. It is also a versatile technique that can be easily adapted to incorporate problem-specific constraints.