Mateo W. Racca

UNIVERSITAT ROVIRA i VIRGILI
Escola Tècnica
Superior d'Enginyeria

NEURAL AND EVOLUTIONARY COMPUTING

(MESIIA): Assignment #4: Optimization with Genetic Algorithms

Abstract:

This report explores the application of genetic algorithms (GAs) to the Traveling Salesperson Problem (TSP), a classical problem in combinatorial optimization.

The main objective is to find the shortest possible route that visits a set of cities and returns to the origin point.

The methodology involves translating the TSP into a GA framework, where routes are chromosomes and the total travel distance is the fitness value to be minimized. The result was reached after experimenting with various selection, crossover, and mutation techniques to optimize performance.

The script, datasets and results are available at: https://github.com/raccamateo/NEC_A4

Introduction:

As mentioned, the Traveling Salesperson Problem (TSP) is a well-known problem in the field of optimization and computational mathematics. It involves finding the most efficient route for a salesperson who must visit a list of cities and return to the starting city, with the constraint that each city is visited exactly once. It has significant applications in logistics, planning, and the organization of networks.

The genetic algorithm approach to the TSP is motivated by the need for heuristic methods that can provide good solutions within reasonable timeframes, especially for large datasets where exact algorithms become impractical.

By simulating the process of natural evolution, GAs adaptively search through the space of potential solutions, leveraging mechanisms such as selection, crossover, and mutation to evolve routes that approximate the shortest possible path.

The objective of this report is to demonstrate the effectiveness of GAs in finding near-optimal solutions to the TSP and to analyze the impact of different algorithmic parameters on the quality of the solutions obtained.

Problem Representation:

In the context of genetic algorithms, the TSP is represented as a chromosome, which is essentially a specific sequence of genes. Here, each gene represents a city, and the chromosome as a whole represents a complete tour. The permutation encoding is crucial because it naturally represents the path through each city. In this encoding scheme, each city is listed once, and the order of cities represents the sequence in which they are visited. This direct approach maintains the integrity of the problem by ensuring that each city is visited exactly once before returning to the starting point.

Genetic Algorithm workover:

The script has been designed for solving the Traveling Salesman Problem (TSP) using a genetic algorithm (GA), it approaches the problem in the following way:

Chromosome Representation: Each chromosome represents a possible solution to the TSP, meaning a route visiting all cities implying that it starts and ends in the same one. The chromosome is a sequence (permutation) of numbers, where each number represents a city.

The order of numbers (cities) in the chromosome determines the route: the city represented by the first number is the starting point, and the route follows the sequence of cities as listed in the chromosome.

This representation ensures that each city is visited exactly once, as each city number appears only once in the chromosome.

Selection Techniques Elaboration:

- ➔ Tournament Selection: It involves randomly choosing a subset of the population, the "tournament", and then selecting the best individual from this subset.

  This method balances exploration and exploitation by sometimes selecting sub-optimal solutions for the next generation, preserving diversity.

- ➔ Roulette Wheel Selection: Here the probability of selection is proportional to the chromosome's fitness. Chromosomes with higher fitness have a greater chance of being selected.
- ➔ Rank-Based Selection: Individuals are ranked based on fitness, and selection probability is based on this rank. It reduces the chances of premature convergence by preventing the fittest individuals from dominating selection.

Crossover Techniques:

Partially Mapped Crossover (PMX): it is implemented to mantain the relative order of cities. A segment of one parent's route is copied to the child, and then missing cities are filled from the second parent. This respects the TSP constraint that each city must be visited exactly once.

Order Crossover: It is similar to PMX, but it directly copies a sequential segment from one parent to the child. The rest of the route is filled with cities from the second parent, following their sequence in the second parent but skipping those already in the child.

Mutation Techniques:

Swap Mutation: Because of this implementation, two cities in a chromosome are randomly chosen and swapped, introducing small changes, allowing the algorithm to explore the solution space and avoid local optima.

Scramble Mutation: A subset of the route is chosen and the cities in this subset are shuffled randomly, introducing more significant variability, helping to explore new regions of the solution space.

Elitism: A certain number of the best individuals (with the shortest routes) are carried over to the next generation unchanged. This is implemented because it ensures that the genetic material of the best solutions is not lost, aiding the convergence of the GA towards an optimal or near-optimal solution.

Code Implementation/Solution: It approaches the problem by creating a population of random routes (permutations of city indices), calculating fitness as the total route distance, and applying selection, crossover, and mutation techniques to evolve better solutions over generations. Elitism is used to retain the best solutions.

The mutation rate dynamically changes over generations, starting more explorative and becoming more exploitative. The algorithm is executed with different parameters, and the results are analyzed by plotting the evolution of the best solution's total distance over generations.

Settings:

The genetic algorithm is evaluated using a series of test cases with varying parameters. The parameters include: population size, number of generations, mutation rate, and elite size.

Each parameter set is chosen to examine the algorithm's performance under different conditions. A smaller population size (50) with fewer generations (10) and a higher mutation rate (0.01) can test the algorithm's efficiency in quickly converging to a solution. In contrast, a larger population size (600) over more generations (250) with a very low mutation rate

(0.0001) and a significant number of elites (35) tests the algorithm's ability to thoroughly explore the solution space and fine-tune towards the best solution.

The parameter sets are fixed for all testing cases in order to reach a general and extrapolable approach.

The universal parameter used parameters for each test case are:

Test Case	Population Size	Generations	Mutation Rate	Elite Size
1	50	10	0.01	1
2	100	20	0.01	2
3	200	60	0.01	5
4	200	100	0.005	5
5	600	250	0.0001	35

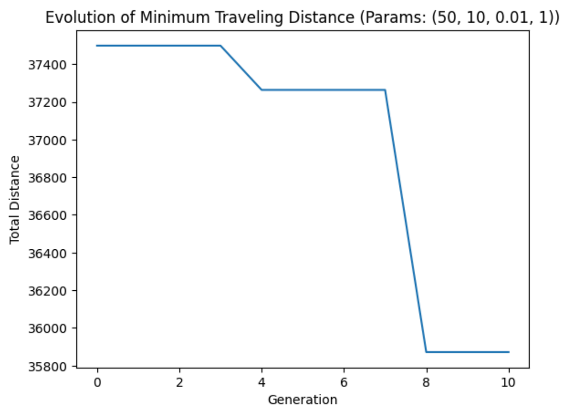
The variety in the test cases, ranging from quick exploration to deep extensive searches allows for a comprehensive assessment of the genetic algorithm's adaptability and effectiveness in solving the TSP across different scales and complexities.

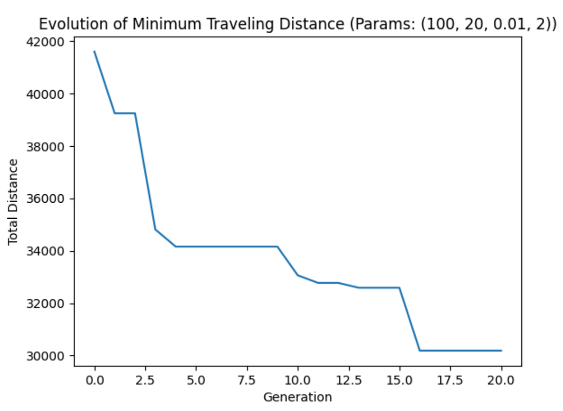
These test cases play a crucial role in evaluating the robustness and flexibility of the genetic algorithm, providing insights into how different parameter configurations can influence the outcome and efficiency of the solution process for the Traveling Salesman Problem.

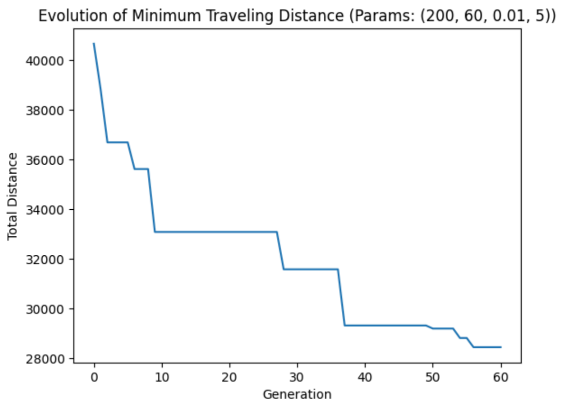
Testing and Analysis:

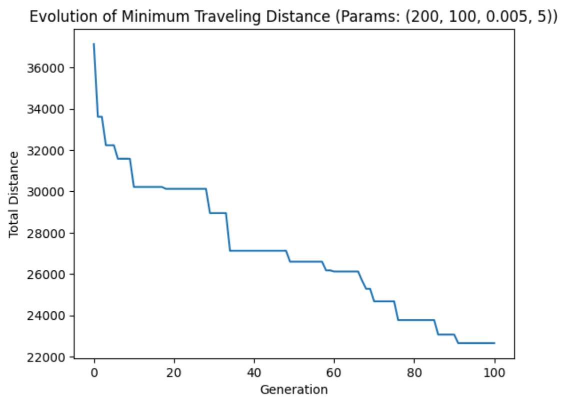
For each case, five tests were conducted. Data is available on http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/ and the results from http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/STSP.html.

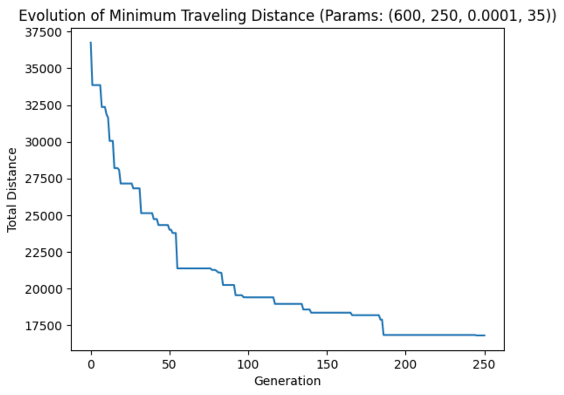
att48:





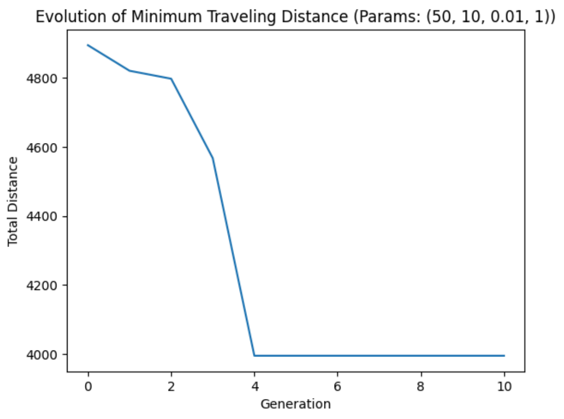


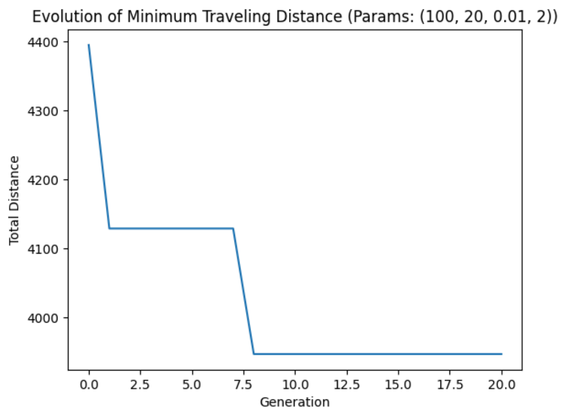


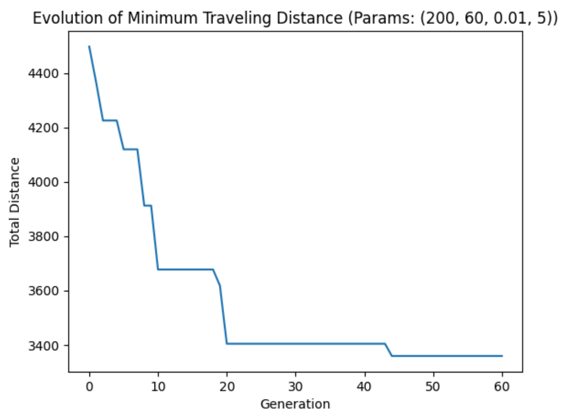


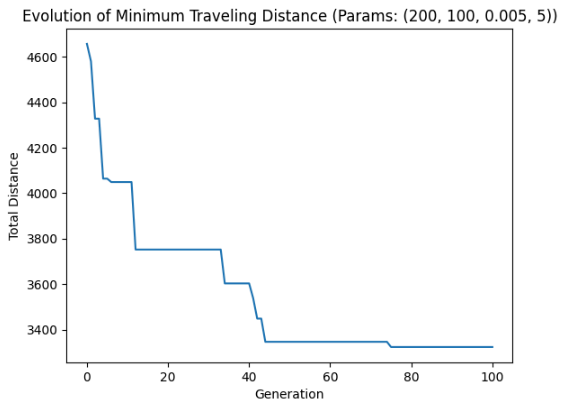
Improvements were made as the population size and number of generations increased, indicating the algorithm benefits from a larger pool of potential solutions and more time to evolve these solutions. The smallest gap from the best known solution was achieved with the largest population and generation count, highlighting the importance of extensive search and optimization periods for complex problems like att48.

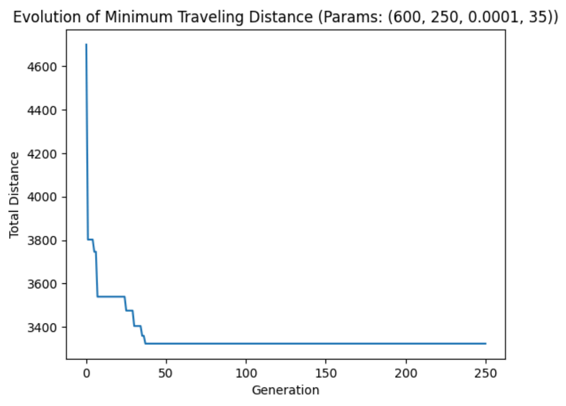
<u>burma14:</u>





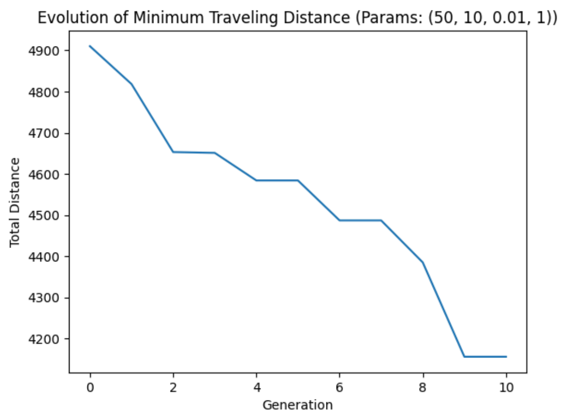


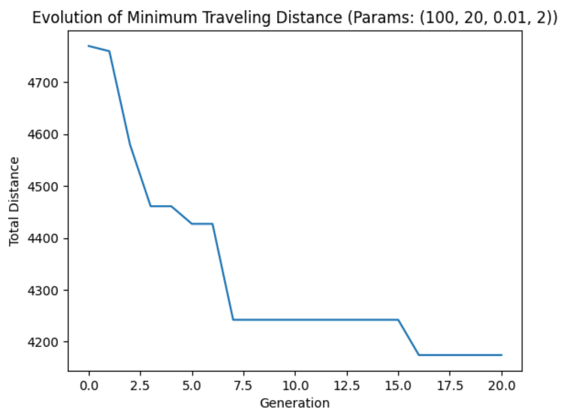


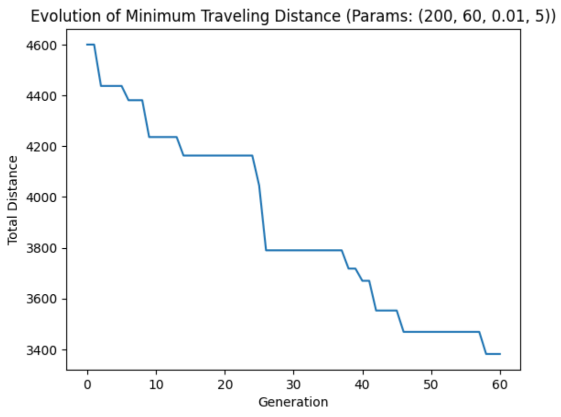


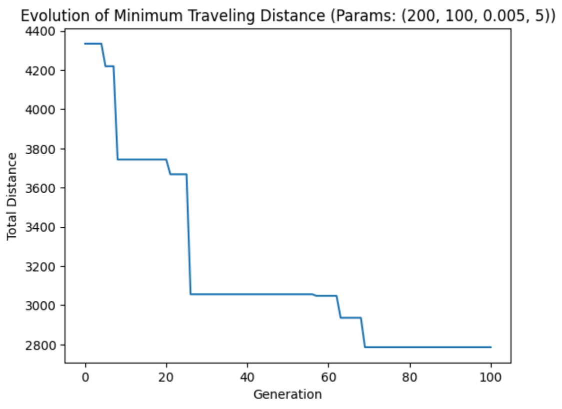
The algorithm consistently found the known best solution across different parameter sets, suggesting that the problem's search space is well mapped by the GA. It implies that burma14 may not require intense computational resources to yield optimal solutions and that the GA parameters are robust for this instance.

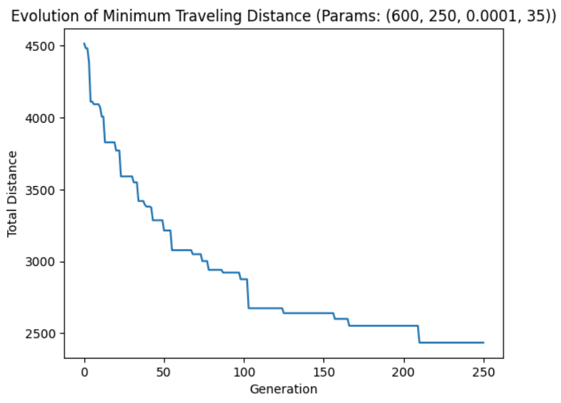
<u>bays29:</u>





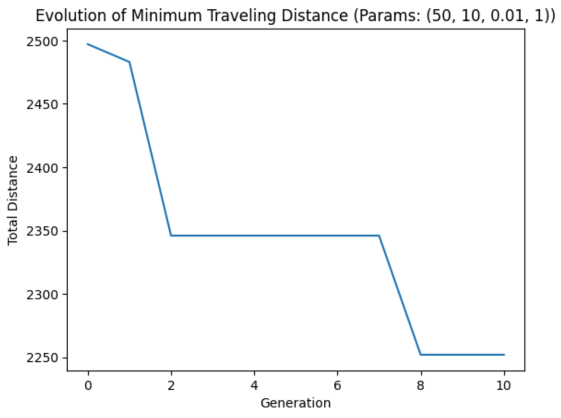


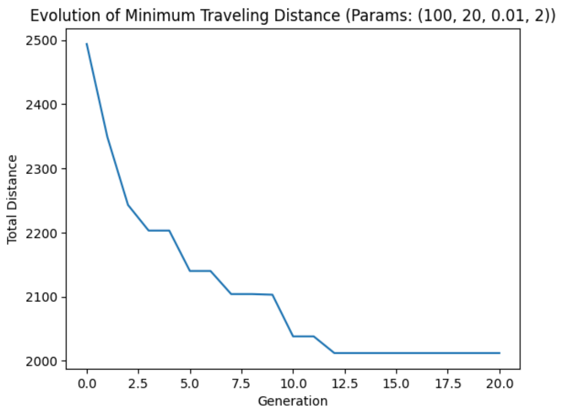


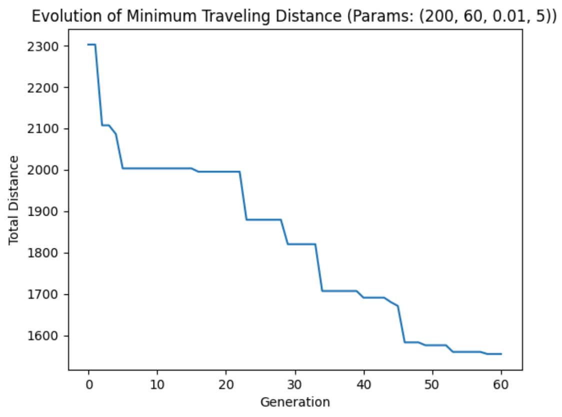


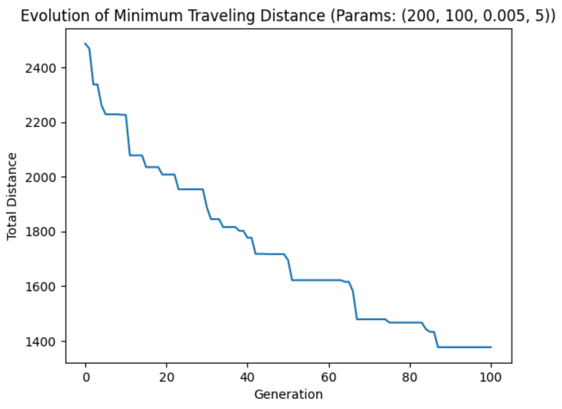
Despite not achieving the best known solution, the GA showed progressive improvements, particularly with an increased number of generations and a larger population. This suggests that the exploration and exploitation mechanisms are functioning correctly but may need further refinement or a longer runtime to converge on the optimal solution.

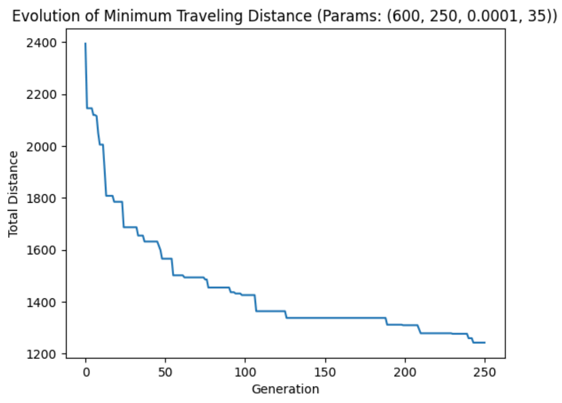
dantzig42:





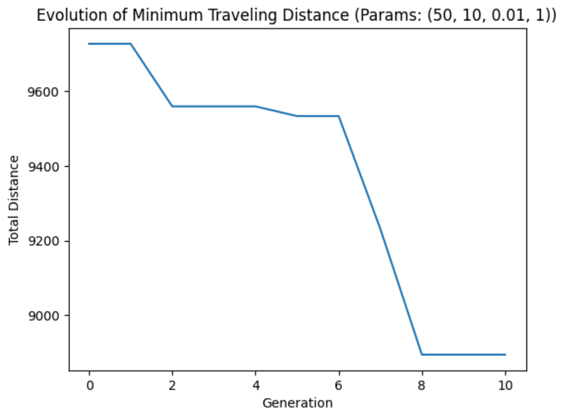


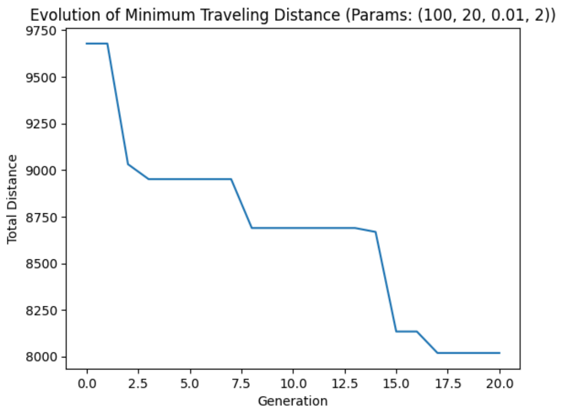


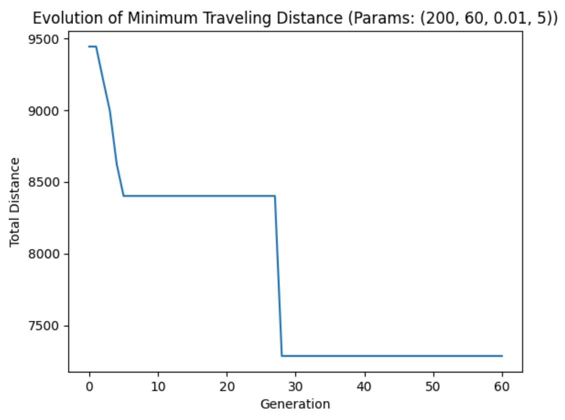


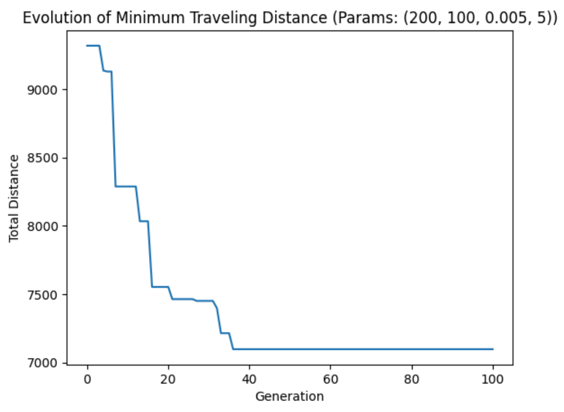
The gap between the GA's results and the best known solution suggests that the problem might be more complex or that the GA requires additional tuning. The improvement as the parameters are scaled up indicates that the algorithm benefits from more diversity and a broader search, but the mutation rate and crossover strategies may also play a significant role in finding better solutions.

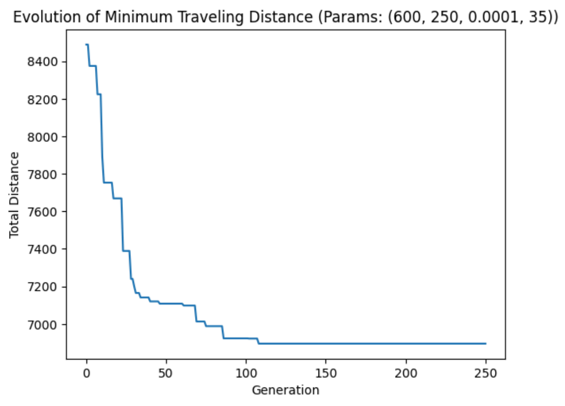
<u>ulysses16:</u>











The GA came close to the known best solution, particularly with the largest population and generation count. This near-optimal performance suggests that the algorithm parameters are almost ideal for this problem size and complexity. However, the slight discrepancy also implies there may be room for improvement, possibly through fine-tuning the mutation rate or crossover method.

Conclusion

The comprehensive work over for the Traveling Salesperson Problem (TSP) illustrates a deep exploration of the genetic algorithm (GA), parameters and their impact on solving various TSP instances. The document concludes with several key insights:

The research indicates a general trend where increasing the population size and the number of generations tends to enhance the GA's ability to find better solutions. This suggests that a broad search space, combined with ample time for evolutionary processes to work, is beneficial. This finding reinforces the principle of exploration and exploitation in GA dynamics—larger populations allow for greater exploration, while more generations enable the refinement of solutions through exploitation.

Secondly, the performance of the GA varied significantly with the complexity of the TSP instance. For the burma14 problem, the GA consistently found the best-known solution, which implies that the GA's approach is well-suited to the problem's complexity and that it falls within a solvable domain for the algorithm with the given parameters.

However, for more challenging problems like att48 and dantzig42, there was a notable gap between the GA's solutions and the best-known solutions. This gap indicates the need for more sophisticated GA strategies or a more granular approach to parameter tuning.

The GA demonstrated the ability to perform reasonably well across different TSP instances, highlighting the adaptability of the algorithm.

Lastly, the study concludes that while the genetic algorithm is a powerful tool for solving TSP instances, there is room for improvement, particularly in complex problem spaces.

Future work may involve fine-tuning the mutation rates or crossover methods, investigating the use of hybrid algorithms that combine the strengths of different optimization techniques, or applying machine learning to intelligently adapt GA parameters in real-time.