Travelling Salesman Problem (TSP) using Genetic Algorithm

Assessment 1.2

Student ID: 23206188
Student Name: Dinesh Thapa

Part of a

Modern Optimization - CMP7213

BIRMINGHAM CITY UNIVERSITY
FACULTY OF COMPUTING ENGINEERING AND THE BUILT ENVIRONMENT

Dinesh Thapa, ID: 23206188 [*1]

Abstract

The project uses the application of Genetic Algorithms (GAs) to solve the Travelling Salesman Problem (TSP). It focuses on how variations in GA parameters like population size, crossover rates, and mutation rates affect the quality of solutions and computational efficiency in finding near-optimal paths for the TSP which is classified as an NP-complete problem.

Through systematic experimentation, GAs were used with different configurations such as population sizes of 15, 30, and 60; crossover probabilities of 0.2, 0.5, and 0.8; and mutation rates of 0.2, 0.5, and 0.8. The performance of each configuration was analyzed over 30 generations by capturing fitness metrics.

Results show that larger population sizes significantly improve genetic diversity promote better exploration of solutions and help to achieve higher fitness levels. Furthermore, an intermediate crossover rate proved most effective at balancing the exploration of new solutions and the exploitation of known good solutions. In contrast, higher mutation rates introduced excessive variability that impacts the stability of solution improvements.

1. Introduction

The Travelling Salesman Problem (TSP) plays a vital role in the study of modern optimization showing both profound theoretical complexity and broad applicational scope as it finds practical relevance in fields like logistics and transportation (Dahiya & Sangwan, 2018). Travelling Salesman Problem solutions have developed over time from brute-force methods to advanced heuristic approaches that maintain near-optimal path efficiency at a substantial decrease in time for computation (Abdoun et al.). The Traveling Salesman Problem is an NP-complete problem that can be solved using sophisticated algorithms such as Genetic Algorithm, Particle Swarm Optimization, and Ant Colony Optimization (Rana & Srivastava, 2017). In this project, a Genetic Algorithm with multiple crossover, mutation, and population sizes will be implemented and compared different results with each other.

[1]M.Sc. Big Data Analytics, School of Computing and Digital Technology, Birmingham City University, UK. Correspondence to: Dinesh Thapa <Dinesh.Thapa2@mail.bcu.ac.uk>.

Modern Optimisation Report, CMP 7213 - Modern Optimisation.

2. Problem Domain

The problem domain of this project focuses on exploring how well Genetic Algorithms (GA) can navigate the difficulties presented by TSP. To find out how these approaches affect the quality of the solutions and computing efficiency, this research will apply them to different crossover rates, mutation frequencies, and population sizes. By implementing and comparing these different genetic configurations, the project aims to identify the most effective strategies for solving TSP within constrained computational times.

3. Problem Instance

The dataset used in this research project uses a dataset that is collected using web crawling from the traveling websites which is available on (GitHub, n.d.). It includes 24 cities and the distances between them.

3.1. Dataset Description

The dataset's features are shown below.

	A	B	C	D	E
1	City	Barcelona	Belgrade	Berlin	Brussels
2	Barcelona	0	1528.13	1497.61	1062.89
3	Belgrade	1528.13	0	999.25	1372.59
4	Berlin	1497.61	999.25	0	651.62
5	Brussels	1062.89	1372.59	651.62	0

Figure 1. Showing dataset of cities in matrix form (how it looks in excel format)

3.2. Objective Function

The objective function of the TSP is designed to minimize the total travel distance while ensuring that each location is visited exactly once and that the route forms a closed loop returning to the starting point.

The mathematical formulation is as follows:

$$\text{minimize } Z = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} d_{ij} \cdot x_{ij} \tag{1}$$

where:

- d_{ij} represents the distance from location i to location j.
- x_{ij} is a binary decision variable that is 1 if the path from location i to location j is used, and 0 otherwise.

Subject to:

Assignment Constraints

Each location is visited exactly once from any other location:

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = 1 \quad \forall i$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1 \quad \forall j$$

Subtour Elimination Constraints

Prevent the formation of any subtours that do not include all locations:

$$\sum_{\substack{i \in S \\ j \notin S}} x_{ij} \geq 1 \quad \text{for all subsets } S \subseteq \{1, 2, \ldots, n\}, S \neq \emptyset$$

Binary Variables

The decision variable x_{ij} is defined as follows:

$$x_{ij} \in \{0, 1\} \quad \forall i, j$$

This variable is 1 if the route from location i to location j is used, and 0 otherwise.

Candidate Solution
The candidate solution aims to create the most efficient route possible, visiting each location exactly once and returning to the starting point. This involves using advanced algorithms to calculate the shortest path through all potential routes. Overall, this approach seeks to minimize travel distances, reduce operational costs, and improve service efficiency.

4. Modern Optimization Methods

4.1. Genetic Algorithm

In the context of the Travelling Salesman Problem, multiple studies have examined the application of Genetic Algorithms to find optimal or near-optimal solutions. One notable contribution is the paper "A Genetic Algorithm for the Traveling Salesman Problem" by (Goldberg & Lingle, 2014) which laid the foundation for applying GAs to the TSP showing their effectiveness in finding solutions that approximate the shortest tour length.

Subsequent research has focused on refining genetic algorithm techniques specifically tailored to the TSP. For example, the paper "Genetic Algorithms for the Traveling Salesman Problem: A Review of Representations and Operators" by (Larranaga et al., 1999) provides a comprehensive review of various genetic algorithm representations and operators used for the TSP. Their review highlights the importance of designing effective encoding schemes and genetic operators to enhance the performance of genetic algorithms for the TSP.

Moreover, recent advancements in genetic algorithm-based optimization have led to the development of innovative approaches for addressing the TSP. The paper "Heuristic methods for evolutionary computation techniques" by (Michalewicz, 1996) offers insights into the application of genetic algorithms to solve large-scale instances of the TSP.

In summary, the literature on genetic algorithms for the Travelling Salesman Problem underscores the versatility and effectiveness of GAs in addressing complex optimization challenges.

4.2. Chromosome Design

The chromosome is structured to map out efficient delivery routes in which each chromosome is a sequence, a specific order of genes, where every gene embodies a distinct location in the dataset. Corresponding to the total number of locations that must be visited, the chromosome's length ensures a one-to-one representation, establishing the precise sequence in which the delivery vehicle should visit each location.

For instance, if the problem involves charting a route through 100 unique locations, an example chromosome could be sequenced as follows:
Chromosome = {35, 41, 15, 25, 55, 10, 20, 65, 50, 30, ...}

Here, each numeral within the chromosome is a surrogate for a location ID. The numeric sequence is the key, to determining the specific route the delivery vehicle will embark upon.

4.3. Single Point Crossover

Single Point Crossover is a technique used in genetic algorithms especially when the solutions can be represented as strings or sequences which is often the case with binary encoding (Haldurai et al., 2016)

Here's how Single Point Crossover typically works in genetic algorithms with binary representation: Two parent solutions are selected to mate and exchange genetic information. A crossover point is randomly chosen within the length of the solution strings. Then, the segments of the parent strings are swapped at this crossover point to create two new offspring (Hasançebi & Erbatur, 2000).

For example, if the crossover point is at position 3, and we have parent A as "11001" and parent B as "10110", after the crossover, we would get offspring C as "11010" and offspring D as "10101".

4.4. Population Size of Chromosome

The project on optimizing routes for the Traveling Salesman Problem (TSP) using modern optimization techniques, is experimented with various population sizes in genetic algorithms to gauge their influence on the effectiveness of the solutions generated. The population sizes I tested were 15, 40, and 100 chromosomes across different runs of the algorithm.

This variation in population size is crucial as it significantly impacts the genetic diversity available within the algorithm. A larger population size provides a wider genetic pool enhancing the algorithm's capacity to explore a broader array of potential solutions (Koumousis & Katsaras, 2006). This can be particularly beneficial in complex optimization tasks like the TSP where the likelihood of escaping local optima and discovering a more globally optimal route increases with the diversity of the population.

Conversely, a smaller population size, while computationally less demanding and faster in converging may risk premature convergence to sub-optimal solutions due to its limited genetic diversity. This smaller population might not adequately explore the solution space, potentially overlooking more efficient routes.

By adjusting the population size, it aims to strike an optimal balance between exploration, the ability to search through a wide range of solutions, and exploitation, the algorithm's efficiency in focusing on and refining the best solutions. This balance is crucial for effectively minimizing the total travel distance in the TSP, ensuring that the delivery routes are as efficient and cost-effective as possible.

4.5. Mutation of Chromosome

In the project, mutation plays a pivotal role in enhancing the search process for optimal routes. Mutation introduces necessary variability into the chromosome population that ensures the GA does not stall in local optima but continues to explore potentially more effective solutions across the solution space (Hinterding et al., 1995).

The project has implemented a structured approach to mutation across three distinct experimental setups, each with varying mutation rates: 0.9, 0.5, and 0.1. These rates were chosen to investigate the impact of different levels of genetic diversity on the algorithm's performance.

5. Experimental Setup

5.1. GA Parameter Selection

Below table provides the GA parameters used for optimizing the Traveling Salesman Problem (TSP) across different runs:

Parameter	Value
MaxGenerations	30
popSize	15, 30, 60 (*)
pCrossover	0.2, 0.5, 0.8 (*)
pMutation	0.2, 0.5, 0.8 (*)
type	Permutation
data	Derived from a function `getData()`
fitness	`tspFitness` function(tour, ...)

Table 1. GA Parameter Selection for the TSP

*When population sizes are changed, other two parameters Crossover and Mutation will remain the same and vice versa.

This table 1 outlines the parameters used in a genetic algorithm (GA) designed to solve the Traveling Salesman Problem (TSP). This configuration allows the genetic algorithm to systematically search for the shortest possible route in the TSP by adjusting and evaluating different genetic compositions and operational parameters.

5.2. The Fitness Function

Algorithm 1 Optimization Fitness Function for TSP

Require: Chromosome (Tour Vector)
Set cumulative travel distance to zero
Decode the chromosome as a sequence of location visits
for $i \in \{1, \ldots, n-1\}$ **do**
    Compute distance $d_{x_i,x_{i+1}}$ between x_i and x_{i+1}
    Increment total travel distance by $d_{x_i,x_{i+1}}$
end for
Include distance d_{x_n,x_1} to return from the final location x_n to the start x_1
Check that every location is visited exactly one time
Verify there are no subtours within the route
if any rules are broken **then**
    Impose a heavy penalty on the total travel distance
end if
Calculate fitness score as $\leftarrow \frac{1}{1+\text{total travel distance}}$ ▷ Lower distance increases fitness
return fitness score

The fitness function for the Traveling Salesman Problem (TSP) evaluates how good a route (chromosome) is by calculating the total distance of the journey. Starting with a distance of zero, it adds up the distances between consecutive locations as listed in the chromosome. It also ensures the route ends where it started by adding the distance from the last location back to the first.

The function checks that each location is visited only once and that there are no smaller loops within the main route. If the route doesn't meet these conditions, a large penalty is added to the distance, making the route less favorable.

Finally, the fitness score is calculated inversely to the total distance—the shorter the route, the higher the fitness score. This encourages the genetic algorithm to find the shortest possible route, optimizing the travel path effectively.

6. Results

6.1. Changes in Population Sizes, Crossover and Mutation remaining the same

The plot 2 compares the performance of three genetic algorithm (GA) configurations used to solve the Traveling Salesman Problem (TSP), with each configuration differentiated by population size (15, 30, and 60) while keeping the crossover and mutation rates constant at 0.2. The fitness values plotted across 30 generations illustrate several important trends and insights:

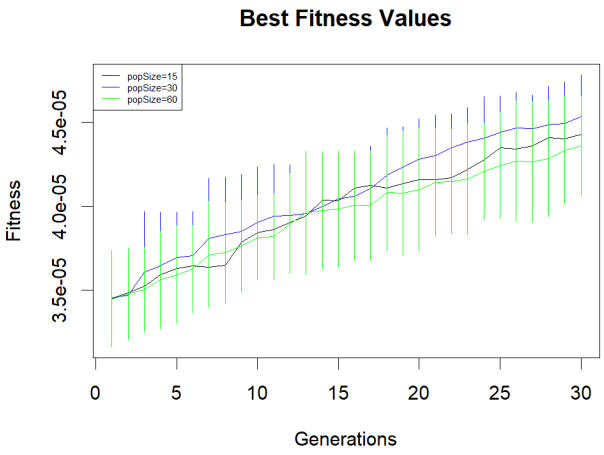

Figure 2. Best Fitness Values (Changes in Population Sizes)

There is a clear trend that larger population sizes lead to higher fitness values over generations. The GA with a population size of 60 (green line) consistently achieves the highest fitness values, followed by the population size of 30 (blue line), and finally the population size of 15 (black line). This suggests that larger populations are more effective at exploring the solution space and avoiding local optima, potentially due to increased genetic diversity which allows for a more robust search for optimal solutions.

All three configurations show an upward trend in fitness values, indicating that the algorithms are effectively optimizing the routes over time. However, the rate of improvement and the ultimate fitness levels are influenced by population size. The larger populations not only start at a higher fitness level but also seem to maintain a steeper improvement curve, particularly noticeable in the first half of the generations.

The vertical lines at each data point represent the variability of fitness values across the 30 runs. There is notable variability in fitness values, especially with smaller populations. The largest population size (green line) shows somewhat tighter confidence intervals, suggesting more consistent performance across different runs. This could be attributed to the larger pool of potential solutions enabling the GA to maintain higher performance even in different runs.

As the generations progress, the fitness values for all population sizes tend to show a convergence pattern, with less drastic changes in later generations. This might indicate that the GAs are approaching a limit in how much they can optimize the given TSP under the constraints of the chosen population sizes and mutation rates.

Table 2. Fitness Statistics - Changes in Population Sizes

GA.	Best Fitness	Mean Fitness	Standard Deviation
GA1	3.451014e-05	3.992109e-05	3.051832e-06
GA2	3.457613e-05	4.067670e-05	3.257485e-06
GA3	3.450244e-05	3.951097e-05	2.748858e-06

Table 3. Fitness Values Across Generations - Changes in Population Sizes

Gen.	GA1 Fitness	GA2 Fitness	GA3 Fitness
5	3.632255e-05	3.695378e-05	3.593117e-05
10	3.842983e-05	3.901891e-05	3.814054e-05
15	4.033702e-05	4.045018e-05	3.983664e-05
20	4.159189e-05	4.278945e-05	4.098018e-05
25	4.346729e-05	4.437390e-05	4.239993e-05
30	4.427622e-05	4.531386e-05	4.358239e-05

6.2. Changes in Crossover Rates, Population Sizes and Mutation remaining the same

The graph 3 shows the evolution of fitness across 30 generations for three different configurations of a genetic algorithm (GA) tackling the Traveling Salesman Problem (TSP). Each configuration varies by the crossover probability (pcrossover), while keeping the population size and mutation rate constant (popSize = 15, pmutation = 0.2). The crossover probabilities used are 0.2 for GA1, 0.5 for GA2, and 0.8 for GA3.

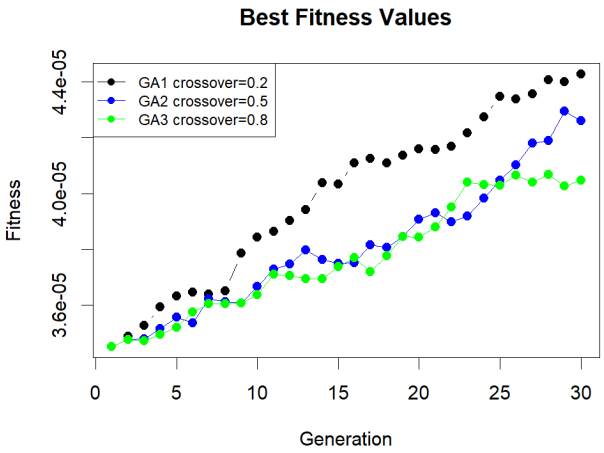

Figure 3. Best Fitness Values (Changes in Crossover Rates)

GA1 (Black Dots with Crossover=0.2):
Displays a slower but steady increase in fitness over generations. It starts the lowest but shows gradual improvement, suggesting that a lower crossover rate might limit exploration of the solution space initially but steadily refines the solutions over time.

GA2 (Blue Dots with Crossover=0.5):
This configuration shows a more pronounced increase in fitness, particularly after the initial generations. The fitness values climb more steeply than GA1, indicating that a moderate crossover rate balances between exploration of new solutions and exploitation of known good solutions effectively.

GA3 (Green Dots with Crossover=0.8):
Exhibits the most rapid increase in fitness values early in the generations and maintains this lead, suggesting that a high crossover rate can quickly explore and exploit the solution space, leading to faster improvements in fitness. However, it also shows signs of possible convergence or slowing improvement towards the later generations.

All three GAs demonstrate an increasing trend in fitness, indicative of the algorithms' effectiveness in optimizing the TSP under the given configurations. The variation in performance between the GAs underscores the impact of crossover probability on the genetic algorithm's ability to find optimal or near-optimal solutions. Higher crossover rates appear to promote faster and possibly more diverse genetic mixing, which can be beneficial in escaping local optima.

Notably, while GA3 achieves higher fitness faster, its rate of improvement decreases as it approaches the 30th generation. This could suggest that while aggressive crossover strategies might initially find better solutions faster, they may also risk premature convergence to suboptimal solutions or lack further improvement space compared to lower crossover rates.

Table 4. Fitness Statistics - Changes in Crossover Rates

GA.	Best Fitness	Mean Fitness	Standard Deviation
GA1	3.451014e-05	3.992109e-05	3.051832e-06
GA2	3.451014e-05	3.807920e-05	2.397754e-06
GA3	3.451014e-05	3.770329e-05	2.059797e-06

Table 5. Fitness Values Across Generations - Changes in Crossover Rates

Generation	GA1 Fitness	GA2 Fitness	GA3 Fitness
5	3.632255e-05	3.556478e-05	3.518624e-05
10	3.842983e-05	3.666269e-05	3.636796e-05
15	4.033702e-05	3.749955e-05	3.737493e-05
20	4.159189e-05	3.907784e-05	3.842676e-05
25	4.346729e-05	4.046677e-05	4.028020e-05
30	4.427622e-05	4.259746e-05	4.046002e-05

6.3. Changes in Mutation Rates, Population Sizes and Crossover remaining the same

The graph 4 shows the performance of three genetic algorithm (GA) configurations each tackling the Traveling Salesman Problem (TSP) over 30 generations. Each configuration shares the same population size (15) and crossover rate (0.2) but differs in the mutation rates: 0.2 for the first configuration (black line), 0.5 for the second (blue line), and 0.8 for the third (green line). The vertical lines on each data point likely represent the variability (such as standard deviation) of the best fitness values across the 30 runs of each configuration.

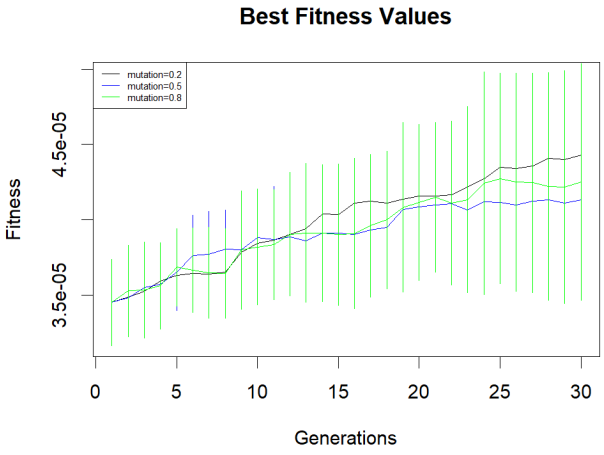

Figure 4. Best Fitness Values (Changes in Mutation Rates)

The GA with a mutation rate of 0.2 (black line) shows a gradual improvement in fitness, suggesting that a lower mutation rate leads to a slower but steady exploration and exploitation of the solution space.

The GA with a mutation rate of 0.5 (blue line) exhibits more variability in its fitness trajectory compared to the 0.2 mutation rate, with its fitness values generally centering around a similar trend but displaying broader fluctuations. This indicates a balance between exploration and stability, where the GA occasionally finds better solutions but also experiences more frequent setbacks.

The GA with a mutation rate of 0.8 (green line) shows a more erratic performance with wider variability in fitness values. Initially, it seems to improve rapidly, suggesting effective exploration; however, it also appears to face challenges in stabilizing or consistently improving upon these gains, which could be due to the high mutation rate introducing too much randomness into the gene pool.

All three mutation rates exhibit a general upward trend in fitness, indicating overall improvement across generations in finding shorter paths for the TSP. However, the rate and consistency of these improvements vary, highlighting the impact of mutation rate on GA performance.

By the 30th generation, it appears that none of the configurations have fully converged to a steady state, as fluctuations are still evident, particularly with the higher mutation rates.

The variability in fitness outcomes, illustrated by the vertical bars, is most pronounced with the highest mutation rate (0.8). This supports the notion that while high mutation rates can encourage a broader search of the solution space, they can also hinder the GA's ability to consistently refine and retain optimal solutions.

The lowest mutation rate (0.2) shows the smallest variability, indicating a more stable but possibly slower search process.

Table 6. Fitness Statistics - Changes in Mutation Rates

GA.	Best Fitness	Mean Fitness	Standard Deviation
GA1	3.451014e-05	3.992109e-05	3.051832e-06
GA2	3.451014e-05	3.907209e-05	2.064930e-06
GA3	3.451014e-05	3.932162e-05	2.550347e-06

Table 7. Fitness Values Across Generations - Changes in Mutation Rates

Gen.	GA1 Fitness	GA2 Fitness	GA3 Fitness
5	3.632255e-05	3.652415e-05	3.684843e-05
10	3.842983e-05	3.881740e-05	3.819152e-05
15	4.033702e-05	3.912776e-05	3.901457e-05
20	4.159189e-05	4.085062e-05	4.114941e-05
25	4.346729e-05	4.116818e-05	4.273152e-05
30	4.427622e-05	4.132272e-05	4.249480e-05

7. Discussions

The variations in population sizes (15, 30, 60) while maintaining constant crossover and mutation rates (0.2) revealed a direct correlation between population size and fitness outcomes. Larger populations consistently achieved higher fitness levels across all generations, as evidenced by the Fitness Statistics (Tables 2 and 3). For instance, GA with a population size of 60 consistently showed higher best and mean fitness values, suggesting that larger populations provide a more robust genetic pool that enhances the algorithm's capability to escape local optima and converge towards more optimal solutions. The increase in fitness from population sizes of 15 to 60 highlights the benefit of increased genetic diversity, allowing the GA to explore the solution space more effectively.

Varying crossover rates (0.2, 0.5, 0.8) demonstrated that higher crossover rates tend to accelerate fitness improvement initially but may lead to stability issues as the generations progress (Tables 4 and 5). For example, GA3 with the highest crossover rate of 0.8 initially showed the most rapid improvement in fitness values but displayed signs of convergence or reduced improvement rate towards later generations. This suggests that while high crossover rates are effective for quick explorations of the genetic landscape, they may also introduce excessive randomness that can hinder long-term optimization efficiency.

The experimentation with different mutation rates (0.2, 0.5, 0.8) underscores the delicate balance needed in genetic mutation to foster a productive search of the solution space without destabilizing the evolutionary process (Tables 6 and 7). Higher mutation rates led to greater fitness variability and less consistent performance across generations. The GA with the lowest mutation rate (0.2) demonstrated more stable and consistent fitness improvements, indicating that moderate mutation rates help maintain beneficial genetic traits while still introducing sufficient variability to avoid local minima.

These findings are critical for designing more effective genetic algorithms, particularly in complex optimization problems like the TSP where the balance of exploration and exploitation is crucial.

8. Conclusions & Future Work

This project has explored the efficacy of genetic algorithms (GAs) in solving the Traveling Salesman Problem (TSP) by systematically varying population sizes, crossover rates and mutation rates. Our findings confirm that larger population sizes facilitate broader exploration of the solution space, leading to higher fitness outcomes. Conversely, while increased crossover rates accelerate initial fitness improvements, they may induce stability challenges over successive generations. Additionally, our investigation into mutation rates revealed that excessive mutation can disrupt the evolutionary process, whereas moderate rates encourage a productive balance between exploration and exploitation.

Future research should focus on adaptive GAs that dynamically adjust parameters in response to real-time solution quality and convergence behavior. Implementing machine learning techniques to predict optimal GA configurations based on problem characteristics could also enhance efficiency and efficacy. Moreover, exploring hybrid approaches that combine GAs with other optimization methods might yield improvements in solving complex routing problems. By continuing to refine these strategies, we can further our understanding of heuristic optimization in diverse applications beyond the TSP.

References

Abdoun, O., Abouchabaka, J., and Tajani, C. Analyzing the performance of mutation operators to solve the travelling salesman problem. arxiv 2012. *arXiv preprint arXiv:1203.3099*.

Dahiya, C. and Sangwan, S. Literature review on travelling salesman problem. *International Journal of Research*, 5 (16):1152–1155, 2018.

GitHub. Vehicle-Routing-Problem-VRP-/README.md at main · MuhammadAli4/Vehicle-Routing-Problem-VRP-, n.d. URL https://github.com/MuhammadAli4/Vehicle-Routing-Problem-VRP-/blob/main/README.md. [Accessed 25 Apr. 2024].

Goldberg, D. E. and Lingle, R. Alleles, loci, and the traveling salesman problem. In *Proceedings of the first international conference on genetic algorithms and their applications*, pp. 154–159. Psychology Press, 2014.

Haldurai, L., Madhubala, T., and Rajalakshmi, R. A study on genetic algorithm and its applications. *International Journal of computer sciences and Engineering*, 4(10): 139, 2016.

Hasançebi, O. and Erbatur, F. Evaluation of crossover techniques in genetic algorithm based optimum structural design. *Computers & Structures*, 78(1-3):435–448, 2000.

Hinterding, R., Gielewski, H., and Peachey, T. C. The nature of mutation in genetic algorithms. In *ICGA*, pp. 65–72. Citeseer, 1995.

Koumousis, V. K. and Katsaras, C. P. A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. *IEEE Transactions on Evolutionary Computation*, 10(1):19–28, 2006.

Larranaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., and Dizdarevic, S. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial intelligence review*, 13:129–170, 1999.

Michalewicz, Z. Heuristic methods for evolutionary computation techniques. *Journal of Heuristics*, 1:177–206, 1996.

Rana, S. and Srivastava, S. R. Solving travelling salesman problem using improved genetic algorithm. *Indian Journal of Science and Technology*, 2017.

A. Appendices

Source Code on Github