# CSCI 4511W: Final Project Report

May 8, 2023

Sol Kim
University of Minnesota
Minneapolis, Minnesota, USA
kim01540@umn.edu

## Abstract

A genetic algorithm is a heuristic search algorithm inspired by natural selection. It is used to find optimal solutions to optimization and search problems. The Travelling Salesman Problem is a well-known NP-hard problem that can be very time-consuming to solve for large problem sizes. While various artificial intelligence algorithms can be applied to solve the Travelling Salesman Problem, the genetic algorithm stands out as a superior choice due to its ability to handle large-scale problems with numerous variables and constraints, and its parallel computing capability that allows for faster convergence to optimal solutions. In this study, the genetic algorithm is utilized as a powerful optimization technique to find optimal routes for the Travelling Salesman Problem by using fitness evaluation and genetic operators such as selection, crossover, and mutation.

## 1 Introduction

The Travelling Salesman Problem is a problem that seeks to find the shortest possible route. A salesman must visit a given set of cities, visiting each city exactly once and returning to the origin city. The primary objective of the Travelling Salesman Problem is to find efficient routes that can minimize travel time and costs [2]. While the Travelling Salesman Problem sounds simple and easy, it is a well-known NP-hard problem, implying no polynomial time algorithm for solving the problem, and as more destinations are added, the complexity of solving the problem increases exponentially. Despite considerable effort by researchers to find effective solutions to the Travelling Salesman Problem, a tractable algorithm for solving the problem has not been discovered yet [3].

The genetic algorithm is a heuristic search algorithm that simulates the process of natural selection to solve optimization problems. The genetic algorithm, similar to other heuristic algorithms, does not guarantee finding the best solution to an optimization problem. Instead, its objective is to identify a good solution within a reasonable time frame [18]. Despite this, the genetic algorithm is frequently used for optimization problems because it is much faster and much more efficient in finding optimal or near-optimal solutions to complex problems with large search spaces due to its ability to explore a large number of candidate solutions simultaneously and maintain diversity in the population [5].

The Travelling Salesman Problem can be solved using various artificial intelligence algorithms, such as Simulated Annealing and Ant Colony Optimization, which have been shown to be effective. However, these algorithms have their limitations. The simulated annealing algorithm can get stuck in local optima, while the ant colony optimization algorithm can be sensitive to parameter settings and require a large number of iterations to converge to near-optimal solutions [4] [1]. In comparison, the genetic algorithm stands out as the more suitable and flexible option for finding good solutions to the Travelling Salesman Problem, as it can simultaneously explore a large search space and maintain diversity in the population through the use of genetic operators such as selection, crossover, and mutation [13] [18].

## 2 Travelling Salesman Problem

The purpose of the Travelling Salesman Problem is to find the shortest route for a travelling salesman who starts from one city, visits each city exactly once, and returns to the origin city [2]. The challenge of the problem lies in the vast number of possible routes that can be taken. The Travelling Salesman Problem is considered an NP-hard combinatorial optimization problem, meaning that it is impossible to find a practical and efficient solution for large instances of the problem. The complexity of the problem grows exponentially with the number of cities to be visited, resulting in an enormous number of possible routes that must be evaluated to determine the optimal solution. As a result, as the number of cities to be visited increases, the problem becomes more challenging to solve. In practice, this means that finding the optimal solution for the Travelling Salesman Problem is extremely challenging except for the smallest cases, and heuristic algorithms such as the genetic algorithm are often employed to find near-optimal solutions in a practical time frame [3] [10].

Despite its complexity, the Travelling Salesman Problem remains an active area of research, and researchers are constantly studying new algorithms and techniques to solve the problem efficiently and effectively. The Travelling Salesman Problem's popularity and complexity have made it a benchmark problem for evaluating the performance of optimization algorithms and for comparing different algorithmic approaches [2].

Moreover, in addition to its use in traveling between cities or countries, the Travelling Salesman Problem has diverse applications in various fields such as logistics, DNA sequencing, manufacturing, transportation, and astronomy for telescope scheduling. For example, in logistics, the Travelling Salesman Problem is used to optimize the routes and sequence of deliveries for vehicles, reducing transportation costs and time while ensuring that all destinations are visited. In DNA sequencing, the Travelling Salesman Problem is used to determine the shortest possible path to visit all the required DNA fragments for sequencing, reducing the cost and time needed to sequence a genome [15].

## 3 Genetic Algorithm

The genetic algorithm was originally developed by John Holland and his colleagues in the 1960s and 1970s. It is an optimization technique that is inspired by the principles of evolution, specifically Charles Darwin's theory of natural selection [18]. The genetic algorithm is commonly used to solve complex problems that deterministic or traditional algorithms find too costly in terms of time and processing. Over the years, the genetic algorithm has been able to solve problems that were previously considered difficult or impossible to solve using other methods [5].

It works by mimicking the process of natural selection to generate new solutions to a problem. The algorithm starts with a population of possible solutions, and through a process of the genetic operators, including selection, crossover, and mutation, it generates a new population of solutions that are potentially better than the previous population. The selection operator is implemented after the initial population is initialized. The purpose of selection is to choose the fittest individuals, which have a higher chance of passing on their favorable traits to the next generation [5] [16]. After the selection operator is applied, the crossover operator selects the two or more selected individuals as parent individuals, chooses a random point in their genetic material, and exchanges the genetic information to create new offspring individuals [8]. The crossover operator allows the genetic algorithm to explore various regions of the search space and create new solutions by combining the characteristics of multiple individuals. After the crossover operator, the mutation operator is performed by randomly selecting a point to mutate and swapping that point [8]. The mutation operator is essential to discover novel solutions and prevent the algorithm from getting stuck in local optima [11]. As a result of implementing the genetic operators, the average quality of the population increases. This process is repeated until an optimal or satisfactory solution is found, or a termination criterion is met.

The genetic algorithm is particularly well suited for optimization problems with large search spaces and complex constraints and objectives, such as the Travelling Salesman Problem. It has two primary advantages over traditional optimization methods. Firstly, since the genetic algorithm evolves a population of individuals over multiple generations, it can explore a broader range of the search space and find better solutions. Secondly, the genetic algorithm is highly flexible and can be tailored to fit various problem objectives and constraints. The fitness function which is used during the evaluation of individuals can be designed to reflect the problem's specific objectives and constraints. Additionally, the genetic operators, such as selection, crossover, and mutation, can be adjusted to meet the problem's requirements. By this approach, the genetic algorithm can handle a wide range of problem types [5] [18].

It is important to carefully select the function and setting formulation when implementing the genetic algorithm to guarantee meaningful outcomes since inappropriate choices can significantly decrease the algorithm's performance [18]. Despite this limitation, the genetic algorithm is one of the most widely used optimization algorithms.

## 4 Relevant work

The research paper titled "Exploring Travelling Salesman Problem Using Genetic Algorithm" [13] solved the Travelling Salesman Problem using the genetic algorithm in various ways. The authors discussed the performance of the different operators used in the genetic algorithm, such as mutation, crossover, and selection, and emphasize the importance of parameter settings to obtain optimal outcomes. Through case studies, the authors demonstrated the effectiveness of the proposed approach and provide evidence of its performance in solving the Travelling Salesman Problem. Overall, the paper provided the application of the genetic algorithm to the Travelling Salesman Problem, and also presented practical evidence of its efficiency in addressing real-world issues.

The research paper "Travelling Salesman Problem Optimization Using Genetic Algorithm" [7] proposes a genetic algorithm approach to solving the Travelling Salesman Problem. The authors implemented the genetic algorithm with selection, crossover, and mutation operators to find optimized solutions. They tested the genetic algorithm on a set of 50 state capital cities in the United States and experimented with different population sizes, mutation rates, and crossover rates to evaluate the impact on the algorithm's performance. The results showed that the genetic algorithm is an effective approach for solving the Travelling Salesman Problem, and the optimal solution is found within a reasonable amount of time.

## 5 Different approach

In the research article titled "Study of Variation in TSP using Genetic Algorithm and Its Operator Comparison" [9], the authors compared the implementation of the Travelling

Salesman Problem using the genetic algorithm and the nearest neighbor algorithm. The nearest neighbor algorithm is a simple heuristic algorithm that is similar to the greedy algorithm. In contrast to the genetic algorithm, it selects a random starting city and travels to the city closest to the previous city, repeating the process until all cities are visited. Although it has the advantages of fast execution and ease of implementation, it often fails to find shorter routes that are easily discernible to humans because it does not consider the overall structure of the tour. Consequently, the optimal solution is unlikely to be found using the nearest neighbor algorithm. The authors conducted experiments to compare the performance of the genetic algorithm and the nearest neighbor algorithm, and as expected, the genetic algorithm produced much better results for the travelling salesman problem. The total length of the solution route from the nearest neighbor algorithm was around 1.2 to 1.3 times longer than the solution obtained from the genetic algorithm.

## 6 Approach

The travelling salesman problem can be approached and solved using various representations and genetic operators such as binary, matrix, or path representations, and additional genetic or mutation operators [11]. The common approach to solve the problem using the genetic algorithm is to represent each city in a given set of cities as a gene, where the goal is to identify the shortest route that visits each city only once before returning to the starting city. Chromosomes refer to any routes that adhere to this criterion. The fitness function evaluates each chromosome and assigns it a score based on a specific criterion, like the total length of the route. Elite chromosomes with higher scores are chosen as parents for the next generation in the mating pool via crossover. After crossover, the mutation operator is performed by randomly selecting a chromosome to mutate. The algorithm repeats this process until a solution converges.

This approach incorporates three operators: selection, crossover, and mutation. The selection operator chooses chromosomes based on their fitness, while the crossover operator generates new chromosomes using the chosen parent chromosomes as parents. At this stage, the population size usually decreases since unfit individuals are removed. Additionally, the average quality of the population improves owing to the crossover operator's ability to generate better chromosomes. The mutation operator then allows for the creation of new individuals by altering genes. Its function is to explore new states, maintain diversity, and avoid premature convergence. [8] [14]

To test the Travelling Salesman Problem, the genetic algorithm is utilized based on the previous work of Eric Stoltz [16] and Lee Jacobson [6], along with certain modifications. For a realistic simulation, a real data set of cities from the GNU LESSER GENERAL PUBLIC LICENSE [17] is used. A set of 50 state capitals in the United States serves as a group of cities that must be traveled.

During the evolution process, three genetic operators, including selection, crossover, and mutation, are employed to create better solutions in each generation of the algorithm.First, an initial population of possible routes is randomly created based on a certain number of population size. Then, the fitness function evaluates the fitness of each route based on its total distance. The best solutions, those with higher fitness scores, are selected as parents to create the next generation. The crossover and mutation operators are applied to the selected parent individuals to create the next generation. The crossover operator involves exchanging genetic information between two parent routes to produce offspring solutions, while the mutation operator introduces random changes in an offspring route to explore new search areas and introduce diversity. This process is repeated iteratively until a satisfactory solution is obtained or a termination criterion is met.

## 7 Implementation

### 7.1 Initialization

The genetic algorithm is tested on a set of 50 state capital cities in the United States, such as Saint Paul, Sacramento, Salt Lake City, Albany, Springfield, Tallahassee, Denver, Little Rock, etc. Each city's location is represented by x and y coordinates, which are used to calculate the distance between cities. The distance between two cities is calculated by using the distance formula derived from the Pythagorean Theorem. For example, the location of Sacramento is (-8392.976246048636, 2664.025175599511), and the location of Madison is (-6176.077618882252, 2976.2765709408 56). The distance from Sacramento to Madison is 2238.781020 3378126.

### 7.2 Evaluation

The initial population size is set to 100, generating 100 possible routes that satisfy the constraints of the Travelling Salesman Problem. The fitness function evaluates each of the 100 chromosomes, which represent possible routes, based on the total distance required to travel to all cities and ranks them accordingly. Out of these 100 initial routes, the fittest one with the shortest total distance is the following route:

> [ Topeka > Tallahassee > Jackson > Pierre > Honolulu > Phoenix > Salem > Cheyenne > Olympia > Jefferson City > Austin > Charleston > Lansing > Baton Rouge > Hartford > Providence > Lincoln > Helena > Juneau > Columbus > Little Rock > Carson City > Montgomery > Annapolis > Des Moines > Sacramento > Boston > Bismarck > Trenton > Raleigh > Richmond > Concord > Indianapolis > Salt Lake City > Denver > Saint Paul > Madison > Harrisburg > Springfield > Atlanta

> Montpelier > Nashville > Columbia > Dover > Augusta > Boise > Santa Fe > Oklahoma City > Albany > Frankfort ]

This fittest route has a total distance of 65267.94926399844 for the entire journey.

As the population size determines the number of candidate solutions present in each generation, it plays an important role in the genetic algorithm. A larger population size generally leads to a better exploration of the search space and a higher likelihood of finding a good solution [18].

This is the fittest route found from an initial population of 500:

[ Columbia > Atlanta > Nashville > Jefferson City > Springfield > Indianapolis > Frankfort > Columbus > Lansing > Harrisburg > Dover > Hartford > Montpelier > Concord > Augusta > Boston > Providence > Albany > Trenton > Annapolis > Richmond > Raleigh > Tallahassee > Montgomery > Baton Rouge > Jackson > Little Rock > Denver > Cheyenne > Pierre > Bismarck > Helena > Boise > Salt Lake City > Phoenix > Carson City > Sacramento > Salem > Olympia > Juneau > Honolulu > Santa Fe > Austin > Oklahoma City > Topeka > Lincoln > Des Moines > Saint Paul > Madison > Charleston ]

This fittest route has a total distance of 58230.23844776648 for the entire journey which is 7037.710816231956 shorter than the shortest route found from an initial population of 100.

Although the larger population size gives a better solution, it also increased the computational cost of the algorithm, as more solutions need to be evaluated at each generation. Therefore, the population size should be chosen based on the specific problem being solved, the available computational resources, and the desired level of optimization.

### 7.3 Selection

After evaluating the fitness of each chromosome in the population, the next step is to select the best individuals to participate in the next generation as parents. The selection process uses a roulette wheel selection method, also known as the fitness proportionate selection, which assigns a probability of selection to each individual based on their fitness. The higher the fitness, the greater the chance of an individual being selected as a parent for the next generation [12].

### 7.4 Crossover

After the selection operator is applied, the fittest individuals are selected to form a mating pool. Pairs of individuals from this pool are then mated using a crossover operator to produce offspring. However, because the Travelling Salesman Problem has a unique constraint that each city must be visited exactly once, a specific type of crossover called ordered crossover is implemented. In this crossover method we randomly select a subset from the first parent route, and then add that subset in the same order to the offspring. The remaining genes in the offspring are filled in by copying genes from the second parent in the order they appear while ensuring that no genes are duplicated in the subset [16] [6]. For example, let's take two parent routes:

[ Dover > Augusta > Boise > Santa Fe > Oklahoma City > Albany > Frankfort ],

[ Frankfort > Albany > Oklahoma City > Santa Fe > Boise > Augusta > Dover]

For convenience, each route has consisted of seven unique cities, and 0 is used to denote an unfilled spot in the offspring route. The randomly selected subset from the first parent is copied to the offspring, like the following:

[0 > 0 > Boise > Santa Fe > Oklahoma City > 0 > 0]

Then, the remaining unfilled spots in the offspring are filled with the cities from the second parent, as following:

[ Frankfort > Albany > Boise > Santa Fe > Oklahoma City > Augusta > Dover]

This crossover technique ensures that the offspring route includes a subset of cities from the first parent, while also preserving the order of the cities in the second parent that are not already included. The resulting offspring route is a valid solution to the Travelling Salesman Problem, and represents a combination of the two parent routes.

### 7.5 Mutation

The newly created offspring individuals through the crossover operator undergo mutation process to explore other space and to maintain the diversity [13]. In order to satisfy the constraint of the Travelling Salesman Problem that each city must be visited exactly once, the algorithm employs the swap mutation operator [6]. The swap mutation operator randomly selects two cities in an individual's route and swaps their positions. Here is an example route:

[ Jefferson City > Austin > Charleston > Lansing > Baton Rouge > Hartford > Providence > Lincoln ]

It can be mutated by swapping the positions of the cities named Jefferson City and Lansing, as shown below:

[ Lansing > Austin > Charleston > Jefferson City> Baton Rouge > Hartford > Providence > Lincoln ]

By introducing such changes, the mutation operator can help to avoid local optima and find better routes in the search space.

### 7.6 Repeat and Result

The genetic algorithm generates new offspring solutions iteratively by performing a sequence of operations including evaluation, selection, crossover, and mutation. The number of generations determines the number of iterations the algorithm will run, so it has a significant impact on the quality of the final solution. Increasing the number of generations enables the algorithm to explore a larger region of the search space and potentially find more optimized solution routes [5].

To evaluate the impact of the number of generations on route optimization, the experiments vary the number of generations while maintaining a fixed population size of 100. For the comparison, the fittest route, the shortest route, in the initial population has a total distance of 65267.94926399844 for the entire journey.

The shortest route after 10 generations is following:

[ Pierre > Oklahoma City > Topeka > Olympia > Salem > Cheyenne > Sacramento > Boise > Santa Fe > Lincoln > Denver > Phoenix > Carson City > Honolulu > Saint Paul > Richmond > Madison > Albany > Trenton > Atlanta > Nashville > Columbia > Baton Rouge > Raleigh > Annapolis > Charleston > Dover > Harrisburg > Frankfort > Tallahassee > Des Moines > Bismarck > Columbus > Indianapolis > Providence > Boston > Augusta > Hartford > Lansing > Concord > Montpelier > Jefferson City > Montgomery > Springfield > Little Rock > Jackson > Austin > Salt Lake City > Juneau > Helena ]

The total distance for the entire journey is 42058.00721194801 which is optimized as 23209.942052050428 compared to the initial distance.

The shortest route after 100 generations is following:

[ Saint Paul > Bismarck > Helena > Boise > Honolulu > Juneau > Salem > Olympia > Sacramento > Carson City > Phoenix > Santa Fe > Salt Lake City > Denver > Lincoln > Oklahoma City > Little Rock > Springfield > Madison > Des Moines > Jefferson City > Tallahassee > Atlanta > Frankfort > Charleston > Annapolis > Richmond > Raleigh > Dover > Trenton > Harrisburg > Columbus > Indianapolis > Lansing > Providence > Boston > Augusta > Montpelier > Albany > Concord > Hartford > Columbia > Montgomery > Nashville > Jackson > Baton Rouge > Austin > Topeka > Cheyenne > Pierre ]

The total distance for the entire journey is 24592.625152701326 which is optimized as 40675.324111297115 compared to the initial distance.

The shortest route after 500 generations is following:

[ Pierre > Bismarck > Helena > Boise > Honolulu > Juneau > Olympia > Salem > Sacramento > Carson City > Salt Lake City > Phoenix > Santa Fe > Denver > Cheyenne > Lincoln > Topeka > Jefferson City > Des Moines > Saint Paul > Madison > Springfield > Little Rock > Nashville > Indianapolis > Frankfort > Charleston > Richmond > Annapolis > Dover > Harrisburg > Columbus > Lansing > Hartford > Providence > Boston > Augusta > Concord > Montpelier > Albany > Trenton > Raleigh > Columbia > Atlanta > Tallahassee > Montgomery > Jackson > Baton Rouge > Austin > Oklahoma City ]

The total distance for the entire journey is 20886.556094193045 which is optimized as 44381.39316980539 compared to the initial distance.

The shortest route after 1500 generations is following:

[ Pierre > Bismarck > Helena > Boise > Honolulu > Juneau > Olympia > Salem > Sacramento > Carson City > Salt Lake City > Phoenix > Santa Fe > Denver > Cheyenne > Lincoln > Topeka > Des Moines > Saint Paul > Madison > Springfield > Jefferson City > Little Rock > Nashville > Frankfort > Indianapolis > Lansing > Columbus > Charleston > Annapolis > Dover > Trenton > Hartford > Providence > Boston > Augusta > Concord > Montpelier > Albany > Harrisburg > Richmond > Raleigh > Columbia > Atlanta > Tallahassee > Montgomery > Jackson > Baton Rouge > Austin > Oklahoma City ]

The total distance for the entire journey is 19663.51299274154 which is optimized as 45604.4362712569 compared to the initial distance.

The shortest route after 5000 generations is following:

[ Topeka > Pierre > Bismarck > Helena > Boise > Carson City > Sacramento > Honolulu > Juneau > Olympia > Salem > Salt Lake City > Phoenix > Santa Fe > Denver > Cheyenne > Lincoln > Des Moines > Saint Paul > Madison > Springfield > Jefferson City > Little Rock > Nashville > Frankfort > Indianapolis > Lansing > Columbus > Charleston > Annapolis > Dover > Trenton > Hartford > Providence > Boston > Augusta > Concord > Montpelier > Albany > Harrisburg > Richmond > Raleigh > Columbia > Atlanta > Tallahassee > Montgomery > Jackson > Baton Rouge > Austin > Oklahoma City ]

The total distance for the entire journey is 19361.82369640121 which is optimized as 45906.12556759723 compared to the initial distance, but not very different with the solution distance from 1500 generations.

Increasing the number of generations can lead to finding a more optimized route with a shorter distance, but it also comes with the cost of increased computational time. Therefore, selecting an appropriate number of generations is essential for the genetic algorithm, as it significantly affects the algorithm's performance and the quality of the final solutions obtained.

## 8 Analysis

The genetic algorithm is used to solve the Travelling Salesman Problem for a set of 50 state capital cities in the United States. The goal of the problem in the experiment is to find the shortest possible route that visits all 50 state capitals in the United States exactly once and returns to the starting city. Since the Travelling Salesman Problem is an NP-hard problem, the genetic algorithm is used with genetic operators to find optimized solutions in a reasonable time.

The location of each city is represented by x and y coordinates. The distance formula derived from the Pythagorean Theorem is used to calculate the distance between two cities. To test the impact of population size on the performance of the algorithm, it was tested with both an initial population size of 100 and 500. As expected, the algorithm found better solutions with larger population size, highlighting the importance of selecting an appropriate population size in solving this problem.

To assess the effectiveness of the genetic algorithm in optimizing solutions, all three genetic operators, selection, crossover, and mutation, are employed. These operators are widely recognized for their ability to enhance the performance of the algorithm by promoting elitism, exploring new search spaces, and maintaining diversity in the population. Through the existing literature and relevant research, it is expected that the application of these operators will result in improved solutions for the problem at hand. For easier comparison of the experimental results, the population size is kept constant at 100 while the number of generations varies. This ensures that any observed differences in performance between experiments can be more confidently attributed to differences in the number of generations. The experiment results show significant optimization. After 10 generations, the distance of the fittest route, which is the shortest route found, is 42058.00721194801. This is 23209.942052050428 shorter than the fittest route of the initial population, which is at generation 0. With a larger number of generations, the genetic algorithm found the better route. After 1500 generations, the distance of the fittest route is 19663.51299274154 which is 45604.4362712569 shorter compared to the fittest route of the initial population.

Through the experiment, the genetic algorithm has proven to be a useful approach for solving the Travelling Salesman Problem. However, it is important to note that increasing the population size and the number of generations can lead to optimized solutions, but also result in computationally expensive and time-consuming implementation.

## 9 Conclusion

In this paper, the genetic algorithm is used to solve the Travelling Salesman Problem for a set of 50 state capital cities in the United States. The algorithm starts with an initial population of possible routes and evolves these routes over generations through selection, crossover, and mutation operators until an optimal solution is found. During the experiments, the importance of setting formulation and selecting the fitness function was highlighted, and the use of all three genetic operators, selection, crossover, and mutation, was found to enhance the performance of the algorithm in promoting elitism, exploring new search spaces, and maintaining diversity in the population. The results of the experiment showed that the genetic algorithm is an effective method for solving the Travelling Salesman Problem.

To make further progress, one area of future work that could be explored is the impact of different selection methods on the performance of the algorithm. The roulette wheel selection method is used for selection operators, but there is another famous selection method called the tournament selection. The tournament selection involves randomly selecting a group of individuals and selecting the highest fitness individual from that group. This process is repeated to choose the second parent [16]. Additionally, the effect of changing the crossover and mutation rates on the algorithm's performance could also be examined by varying these rates separately to observe their impact on the algorithm's ability to find optimized solutions.

## References

[1] Emile H. L. Aarts, Jan H. M. Korst, and Peter J. M. van Laarhoven. 1988. A quantitative analysis of the simulated annealing algorithm: A case study for the traveling salesman problem. *Journal of Statistical Physics* 50, 1-2 (Jan 1988), 187–206. https://doi.org/10.1007/bf01022991

[2] Robert E. Bixby Vasek. Chvatal Applegate, David L. and William J. Cook. 2011. *The Traveling Salesman Problem : a Computational Study.* Princeton : Princeton University Press.

[3] Computer Science Education Research Group at the University of Canterbury. 2023. The Travelling Salesman Problem - Complexity and Tractability - Computer Science Field Guide. https://www.csfieldguide.org.nz/en/chapters/complexity-and-tractability/the-travelling-salesman-problem

[4] Marco Dorigo and Thomas Stützle. 2004. *Ant Colony Optimization Algorithms for the Traveling Salesman Problem.* 65–119.

[5] Savio D Immanuel and Udit Kr. Chakraborty. 2019. Genetic Algorithm: An Approach on Optimization. In *2019 International Conference on Communication and Electronics Systems (ICCES)*. 701–708. https://doi.org/10.1109/ICCES45898.2019.9002372

[6] LEE JACOBSON. 2012. Applying a genetic algorithm to the traveling salesman problem. https://www.theprojectspot.com/tutorial-post/applying-a-genetic-algorithm-to-the-travelling-salesman-problem/5

[7] Sahib Singh Juneja, Pavi Saraswat, Kshitij Singh, Jatin Sharma, Rana Majumdar, and Sunil Kumar Chowdhary. 2019. Travelling Salesman Problem Optimization Using Genetic Algorithm. *2019 Amity International Conference on Artificial Intelligence (AICAI)* (2019), 264–268.

[8] Fozia Khan, Nasiruddin Khan, Dr. Syed Inayatullah, and Shaikh Nizami. 2010. SOLVING TSP PROBLEM BY USING GENETIC ALGORITHM. *International Journal of Basic Applied Sciences* 9 (08 2010).

[9] Gözde Kizilateş and Fidan Nuriyeva. 2013. On the Nearest Neighbor Algorithms for the Traveling Salesman Problem. In *Advances in Computational Science, Engineering and Information Technology*, Dhinaharan Nagamalai, Ashok Kumar, and Annamalai Annamalai (Eds.). Springer International Publishing, Heidelberg, 111–118.

[10] Marc Kuo. 2020. Solving The Travelling Salesman Problem For Deliveries | Routific. https://blog.routific.com/blog/travelling-salesman-problem

[11] P. Larrañaga, C.M.H. Kuijpers, R.H. Murga, I. Inza, and S. Dizdarevic. 1999. Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. *Artificial Intelligence Review* 13, 2 (1999), 129–170. https://doi.org/10.1023/a:1006529012972

[12] Zbigniew Michalewicz and Marc Schoenauer. 2003. Evolutionary Algorithms. In *Encyclopedia of Information Systems*, Hossein Bidgoli (Ed.). Elsevier, New York, 259–267. https://doi.org/10.1016/B0-12-227240-4/00065-4

[13] Alka Singh and Rajnesh kumar Singh. 2014. Exploring Travelling Salesman Problem using Genetic Algorithm. *International journal of engineering research and technology* 3 (2014).

[14] Shalini S Singh and Ejaz Aslam Lodhi. 2013. Study of Variation in TSP using Genetic Algorithm and Its Operator Comparison.

[15] Ian Stewart. 2021. Pigeons, Curves, and the Traveling Salesperson Problem. https://www.wired.com/story/pigeons-curves-and-the-traveling-salesperson-problem/

[16] Eric Stoltz. 2018. Evolution of a salesman: A complete genetic algorithm tutorial for Python. https://towardsdatascience.com/evolution-of-a-salesman-a-complete-genetic-algorithm-tutorial-for-python-6fe5d2b3ca35

[17] Florida State University. 2011. CITIES - City Distance Datasets. https://people.sc.fsu.edu/~jburkardt/datasets/cities/cities.html

[18] Xin-She Yang. 2021. Chapter 6 - Genetic Algorithms. In *Nature-Inspired Optimization Algorithms (Second Edition)* (second edition ed.), Xin-She Yang (Ed.). Academic Press, 91–100. https://doi.org/10.1016/B978-0-12-821986-7.00013-5