Travelling Salesman Problem using Genetic Algorithm

Kadiyala Monish Mithra
dept. of Computer Science(AI)
Amrita Vishwa Vidyapeetham
Kerala, India
monimithra@gmail.com

Kumpatla Mahit Venkat Gautam
dept. of Computer Science(AI)
Amrita Vishwa Vidyapeetham
Kerala, India
venkatgowtham42@gmail.com

Maddala H S M Krishna Karthik
dept. of Computer Science(AI)
Amrita Vishwa Vidyapeetham
Kerala, India
karthikmaddala24@gmail.com

Paladugula Pruthvi
dept. of Computer Science(AI)
Amrita Vishwa Vidyapeetham
Kerala, India
pruthvichowdary12@gmail.com

***Abstract*—The traveling salesman problem is addressed in this paper using a customizable genetic algorithm-based approach. In this challenge, TSP is used as a domain. TSP has historically been known as an NP-complete problem and is a well-known special type of challenge. Traditional approaches like integer programming as well as graph theory algorithms have been used to solve this problem, with varied degrees of success. This research provides a strategy that employs a genetic algorithm to have the most accurate approximation of the problem while lowering costs. In genetic algorithms, TSP utilizes crossover as a primary operator. There were numerous attempts to locate an adequate crossover operator. This paper explains how to use a new crossover mechanism for just a genetic algorithm to find a near - optimum solution to these problems, leading to a high TSP solution. The efficiency of the crossover operator is examined to those in other crossover operators. The purpose of this research is to use a genetic algorithm to investigate the traveling salesman problem.**

***Index Terms*—Travelling Salesman Problem, Genetic Algorithm, Crossover**

I. Introduction

The Traveling Salesman Problem (TSP) was among the most well-known NP-hard problems, meaning that no explicit algorithm exists to address it in polynomial time. The predicted time to find the best solution is exponential. The shortest path problem is characterized as a rearrangement problem with the goal of finding the shortest path. Cities are the graph's vertices, pathways are the graph's edges, and a path's distance is the edge's length, hence TSP can be represented as just an undirected weighted graph. It's a minimization issue that starts and ends at a certain vertex after only visiting each other vertex once. The model is frequently a complete graph. If there is no path between two cities, an infinitely large edge can be added to finish the graph without changing the best route.

The problem generates a sequence of n cities (c1, c2...cn) and permutations (p1..., pn), the goal is to find pi such that the total among all Distance measure between every vertex and its successor is as small as possible. The first vertex in the combination is the parent of the final node. As demonstrated below, the Euclidean distance d between these two cities with coordinates (x1, y1) and (x2, y2) could be determined as below.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Fig. 1. Eucledian Distance

The most common practical applications of TSP include routine transport of products or resources, determining the shortest customer service route, planning bus lines, and so on, but it can also be used in fields unrelated to travel routes. A computational intelligence technique known as a genetic algorithm (GA) is a search strategy being used in computer science to discover approximate answers for optimization problems.

Genetic algorithms are much more accurately described as a natural evolution-based optimization technique. The livelihood of a fittest idea algorithm is one of them. The goal is to "predict" the solutions first, then combine the best of them to raise a generation of solutions that should outperform the prior one.

The following are the steps in the genetic algorithm process:

1) Encoding: For our problem, a reasonable encoding is determined so that each feasible solution has a unique encoding, which is in the form of a string.
2) Selection: The initial set is then chosen, usually at random, though other methods based on heuristics have been proposed. The fitness of every person in the population would then be calculated, i.e., how well the automatic assembly the problem whether it is close

to the optimum in comparison to another people in a population.
3) Crossover: This fitness is being used to determine a person's likelihood of crossover. Crossover occurs when two persons are recombined to form new individuals that are then copied into the next generation.
4) Mutation: The next mutation takes place. Some people are picked at random to be mutated, and perhaps a mutation location is chosen at random. The character in the string's corresponding position is altered.
5) Decoding: When this is completed, a new generation is created, as well as the cycle is continued until a stopping point is reached. The individuals that are closest to the optimal are decoded at this point, and the procedure is complete.

II. Literature Review

The genetic algorithm was devised by John Holland in the 1970s, although it was not well known until the late 1980s. Genetic algorithms are search and optimization algorithms available upon Darwin's Theory of Natural Selection. Darwin's Natural Selection hypothesis states, "Select the best and remove the rest". Consider a forest community of a certain animal type. Some of the population's organisms are much more capable than the others, and these characteristics let species exist in that situation better than others. Assume that natural resources in the jungle, like food and supplies, are few. As a consequence, these animals are forced to compete for resources. Only the fittest and strongest people will make it out alive, while everyone else will perished. GAs are used to solve a variety of problems that are hard to solve using traditional techniques.

The Traveling Salesman Problem (TSP) is one of the most researched combinatorial optimization problems, according to Aybars Uur's work throughout the "Genetic algorithm based solution for tsp on a sphere". The shortest trip across a certain series of parameters in d-dimensional Metric space is determined by the Euclidean TSP, which is an NP-hard problem. A computational intelligence strategy known as a genetic algorithm (GA) is a search strategy used during computer science to generate approximate answers to optimization problems. Crossover and mutation are biologically inspired themes that are used in this game.

The major goal of FOZIA HANIF KHAN's work throughout "Solving tsp problem by using genetic algorithm" is to offer a new chromosomal representation approach based on binary matrices and new fittest criteria to be utilized as a strategy for finding the best TSP solution. The proposed method's concept is based on a genetic algorithm with artificial inelegance, which has been employed as a search algorithm to locate near optimal solutions. He adds more improvements by supplying an algorithm for both symmetric and asymmetric TSP. Here, we're trying to implement the fresh fittest requirements as well as a new representation of the asymmetric matrix, and we're getting better our workaround by applying crossover and mutation repeatedly to get the best result.

III. Methodology

The below are the essential terms needed to use a genetic algorithm to address the travelling salesman scenario. A city is thought of as a gene with (x, y) coordinates, and then a single route is thought of as an individual.

The term "population" refers to a group of people (potential routes). Routes that were linked to establish a new route are called parents. The mating pool seems to be a group of parents who are used to create the following generation. The fitness method is used to select the effectiveness of each route. Mutation is a method of introducing variation into our population by switching two cities along a path at random. Elitism is a method of carrying on the smartest people from one generation to the next.

The genetic algorithm will take the following phases-

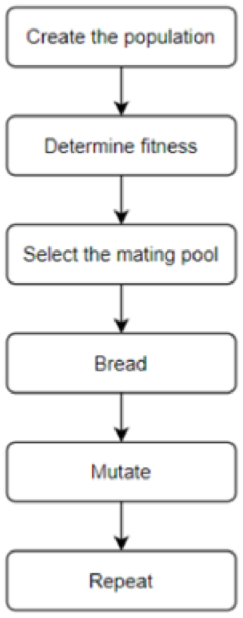


Fig. 2. Steps in Genetic Algorithm

The most important modules, like as numpy as well as pandas, are imported first. After that, we'll have two classes: City and Fitness. Cities can be created and managed using the City class. (x, y) coordinates are used to represent cities. We use repr to add a distance computation and a simpler approach to display the towns as coordinates within the City class. The Fitness class involves calculating actual fitness. In our scenario, we'll use the inversion of a route distance to calculate fitness. We want to reduce route mileage as much as possible, so a higher fitness score is preferable. Because we must begin and end at the same location, this additional computation is factored into the distance calculation.

A. Create the Population

We may now start building our first population, often referred as the very first generation. To do so, we'll need to figure out how to write a function that generates routes that meet our criteria. We choose the sequence in which we attend each city at random to create an individual.

We want a complete population, so we'll cycle through the constructed route function unless we see as many routes because we need for our population. Only the initial population will be created using these techniques. Following generations will be created by breeding and mutation.

B. Determine Fitness

After that, the evolutionary games begin. To recreate our ”survival,” we can use Fitness to estimate each member of the population. Our result will become an ordered list of route ids and fitness scores for each one.

C. Select the Mating Pool

There have been a few various methods to choose the parents who will generate the future generation. The quickest and easiest way are fitness proportionate choice (roulette wheel selection) as well as tournament choice.

The fitness of every individual compared to the general population is employed to assign a chance of selection in fitness proportionate selection. Consider it a fitness-based probability of being chosen. A predetermined number of persons are arbitrarily chosen from the population during Tournament selection, with the very first parent becoming the one with the highest fitness value in the group. To choose another second parent, repeat the process.

Through use of elitism is another design component to consider. With elitism, the population's top performers will automatically pass down from generation to generation, guaranteeing that one of the most successful people survive.

In two steps, the mating pool would be developed. To begin, we'll utilise the rankRoutes function's result to identify which routes to include in our search algorithm. After that, the roulette wheel was set up by calculating each person's relative fitness weight. We next choose our mating pool by relating these weights to a randomly generated number. We'll probably should save our best routes; therefore, elitism will be implemented.

Finally, the selection method contains a list of path ids, which we could use in the matingPool operation to establish the mating pool. We can build the mating pool knowing that we already have the credentials of the paths that will form our mating pool out from search algorithm. We're merely removing the persons we want from our population.

D. Breed

We can develop the following generation in a procedure known as crossover now that we've established our mating pool (breeding). We can actually select a crossover point then fuse the two strings with each other to make an offspring if our people were sequence of zeros and ones and our two requirements didn't apply. The TSP, on the other hand, is unusual in that we must include all places at the same time. We can utilise an unique mating method called ordered crossover to follow this requirement.

During ordered crossover, we arbitrarily take a random sample of its first parent string but then fill the rest of the route with traits again from second parent in the sequence they arrive, without replicating any traits in the first parent's subset.

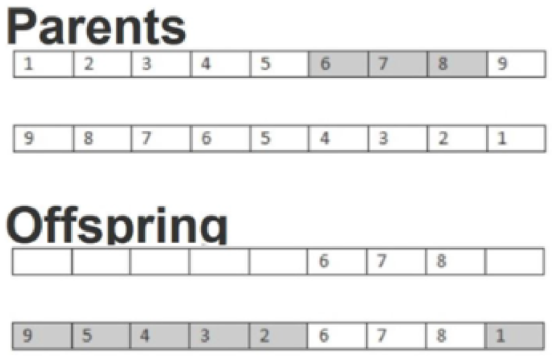


Fig. 3. Ordered Crossover Illustration

We'll next generalise this to generate our population of offspring. We use elitism to maintain the best paths from the current population. After that, the breed mechanism is employed to generate the rest of the next generation.

E. Mutate

In a genetic algorithm, mutation is significant because it helps to prevent local completion by introducing new routes that enable us to explore different areas of the optimal solutions. Whenever it comes to mutation, the TSP has a unique consideration, similar to crossover. Similarly, if we could have a chromosome made up entirely of zeros and ones, mutation would simply suggest a low likelihood of a gene switching from 0 to 1 or vice versa. We can't drop cities, though, because we have to follow our rules. We'll use swap mutation instead. This signifies those two cities will switch places in our itinerary with a predetermined low probability. In our mutate function, we'll perform this for one person. The mutate function can then be extended to execute through all the new population.

F. Repeat

Now we'll combine these functions to make a function that generates a new generation. First, we use rankRoutes to evaluate the paths in the current generation. The selection function is then used to discover our possible parents, allowing us to use the matingPool function to build the mating pool.

Finally, we use the breedPopulation function to produce our new generation before utilising the MutatePopulation function to apply mutation.

Lastly, most of the modules essential for the genetic algorithm's construction are created. We'll start by creating the original population, then repeat across as many iterations as we like. We'd also like to see the optimal route and how far we've progressed, so we may record the starting distance, the starting point, the final destination, and the safest route.

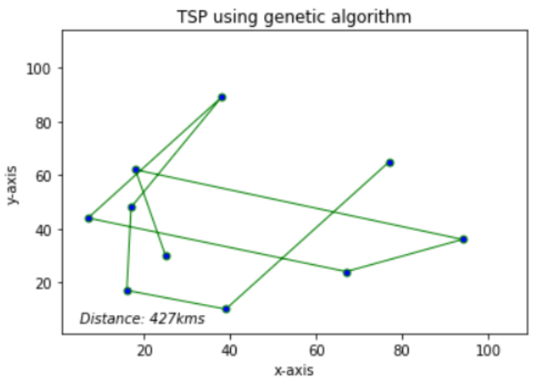
We'll need to have a list of places to travel between in order to execute the genetic algorithm. We'll make a list of various cities at random for this presentation. Then we should create certain assumptions that will help our genetic algorithm work optimally. For example, in each generation, we can pick 100 individuals, maintain 20 elite individuals, apply a 1% mutation rate in terms gene, and repeat the process for 500 generations.

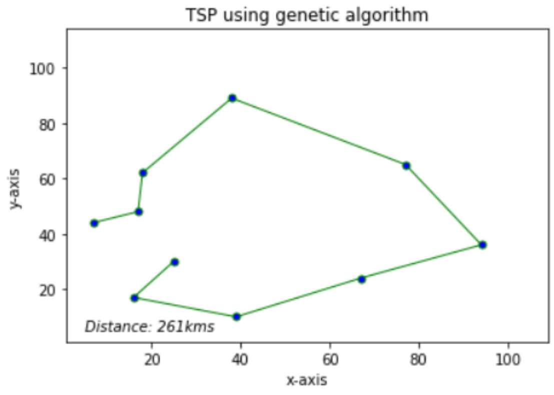
IV. Algorithm

1) Create a population at random.
2) Determine the chromosome's fitness.
3) Repeat until all of the steps have been completed:
   - Choose your parents.
   - Crossover and mutation should be done.
   - Determine the new population's fitness.
   - Add that to the gene pool as an addition.

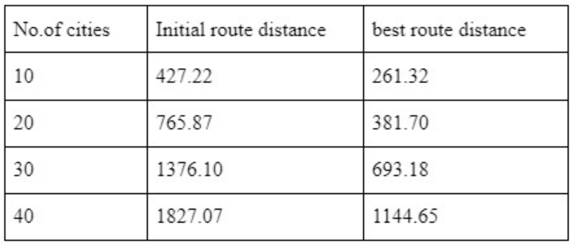
V. Result

The optimum route will be achieved and the procedure will be ended once a variety of processors, such as selection, crossover, and mutation, have been applied to the problem. We should create the best assumptions in order to get the optimal route. We employed 100 children in each generation, maintained 20 elite persons, used a 1 % mutation rate for each trait, then repeated many times through 500 generations in this work.The below Two Figures Depict The Normal and the Best Path Possible for a problem of 10 cities.





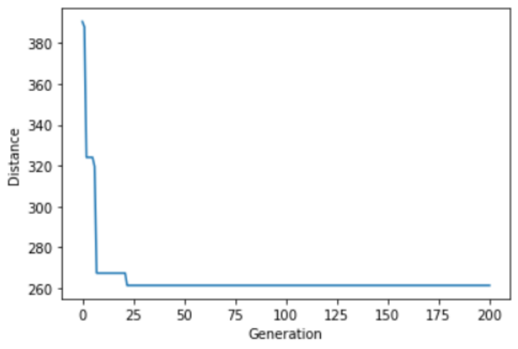
The performance of the proposed optimization technique for a variety of cities are displayed below.



No.of cities	Initial route distance	best route distance
10	427.22	261.32
20	765.87	381.70
30	1376.10	693.18
40	1827.07	1144.65

The table shows that using a genetic algorithm is a speedy way to get the best outcomes.

Finally, an improvement graph is drawn to show how distance has improved over time for every generation.



VI. Conclusion & Future Scope

Combining information from heuristic approaches and evolutionary algorithms to solve the TSP is an effective option. Genetic algorithms appear to discover effective results for such traveling salesman task; however, this is highly dependent on how the challenge is encoded and the crossover and mutation mechanisms used. For the purpose of solving TSP, a variety of genetic algorithm strategies have been examined and surveyed. Different hybrid selection, crossover, and mutation operators can be added to the research. The suggested method can be used to solve a variety of network models, such as task scheduling models and vehicle navigation routing models. The same method can be used to assign frequencies to cells in a cellular network.

Appendix

For further references :
https://github.com/karthik1124/TSP_using_Genetic_Algorithm

References

[1] Varshika Dwivedi, Taruna Chauhan, Sanu Saxena and Princie Agrawal. Article: Travelling Salesman Problem using Genetic Algorithm. IJCA Proceedings on Development of Reliable Information Systems, Techniques and Related Issues (DRISTI 2012) DRISTI(1):25 - 30, April 2012.

[2] Gupta, Saloni Panwar, Poonam. (2013). Solving Travelling Salesman Problem Using Genetic Algorithm. International Journal of Advanced Research in Computer Science and Software Engineering. 3. 376-380.

[3] S. S. Juneja, P. Saraswat, K. Singh, J. Sharma, R. Majumdar and S. Chowdhary, ”Travelling Salesman Problem Optimization Using Genetic Algorithm,” 2019 Amity International Conference on Artificial Intelligence (AICAI), 2019, pp. 264-268, doi: 10.1109/AICAI.2019.8701246.

[4] D. Chen, ”Solving a new type of TSP using genetic algorithm,” IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society, 2017, pp. 3333-3339, doi: 10.1109/IECON.2017.8216564.

[5] R. Liu and Y. Wang, ”Research on TSP Solution Based on Genetic Algorithm,” 2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS), 2019, pp. 230-235, doi: 10.1109/ICIS46139.2019.8940186.

[6] W. Ellili, M. Samet and A. Kachouri, ”Traveling salesman problem of optimization based on genetic algorithms,” 2017 International Conference on Smart, Monitored and Controlled Cities (SM2C), 2017, pp. 123-127, doi: 10.1109/SM2C.2017.8071832.

[7] L. -Y. Wang, J. Zhang and H. Li, ”An Improved Genetic Algorithm for TSP,” 2007 International Conference on Machine Learning and Cybernetics, 2007, pp. 925-928, doi: 10.1109/ICMLC.2007.4370274.

[8] D. Kaur and M. M. Murugappan, ”Performance enhancement in solving Traveling Salesman Problem using hybrid genetic algorithm,” NAFIPS 2008 - 2008 Annual Meeting of the North American Fuzzy Information Processing Society, 2008, pp. 1-6, doi: 10.1109/NAFIPS.2008.4531202.

[9] M. Agrawal and V. Jain, ”Applying Improved Genetic Algorithm to Solve Travelling Salesman Problem,” 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), 2020, pp. 1194-1197, doi: 10.1109/ICIRCA48905.2020.9182884.