Solving Traveling Salesman Problem with Genetic Algorithm Approach

Sangryu Park

May 3rd, 2021

Abstract

Traveling Salesperson Problem(TSP) is NP-hard problem which requires extensive calculation and large number of resources. TSP targets to find the minimum cost of routes that visits all routes in the given nodes and return to the start node. With Genetic Algorithm(GA) approach, a random search method inspired by the law of biological evolution, finds the minimum cost of the routes of the inputted graph. In the same dataset, with different numbers of generations and number of population effects the minimum cost that GA finds. In addition, the number of nodes impacts the accuracy of minimum cost.

1 Introduction

Traveling Salesperson Problem (TSP) is known as optimization problem where the goal of the problem is to find the shortest path that visits each city that is given the list once and return to the starting city [4]. Finding an optimal path in the route is crucial nowadays to use least cost and least time to spend. However in current systems where systems are complicated, calculating least cost brings extensive computation in a complicated system. Therefore, developing the idea of solving TSP has been widely studied in the computer science area with different approaches.

TSP is considered to be NP-hard problem where NP-problem is non-deterministic polynomial time hardness problem. While nodes which represent cities in the real world and routes which connect nodes are simple, the problem of TSP would not be considered to be a complicated problem. However, the real world problem would not be that simple but rather to be complicated such as different routes would be available to travel with different weights. In addition, the number of nodes and routes would be big. Therefore, it is almost impossible to compute all possibilities of route, since the possible routes would be $O(\text{n}!)$, where n is the number of nodes that inputted.

Artificial Intelligence demonstrates extensive calculation based from the algorithms that are created by humans. With coded algorithms, Artificial Intelligence uses extensive amounts of resources from computers able to consider a big number of possibilities or large number computation compared to humans. In addition, with appropriate pruning and selecting, the Artificial Intelligence would provide the result of targeting which is close to the best.

Therefore, TSP using Artificial Intelligence would be an appropriate approach. As mentioned earlier, since Artificial Intelligence is suitable for extensive computation and large resources, TSP which is NP-hard problem will be suitable to use Artificial Intelligence. In this aspect, TSP is an interesting problem in Artificial Intelligence since the TSP problem itself is considered to be NP-hard problem, meaning extensive complexity and compute the result. By establishing steps of how Artificial Intelligence would approach TSP, Artificial Intelligence would decide behavior of its own and also consider how Artificial Intelligence would behave efficiently to approach the goal of TSP which is reducing the cost of route and compute a better route from an already computed route.

2 Literature Review

Traveling salesman problem(TSP) is finding the shortest cost path from origin to the target cities while traveling all cities that are given. TSP was introduced in 1985 from "The Traveling Salesman Problem" and from then TSP has been researched in the computer science area with different methods that developed.

The basic method to find optimal solution for TSP is to compare all possible tour routes and compare the cost of each possible route. However, this method is time consuming and costs a lot.

Thora Tenbrink and Jan Wiener suggest to solve TSP in novel perspective in two respects. In two respects, the methods used are The Nearest Neighbor(NN) strategies, and The Cluster strategy. With those two strategies, the article has conducted two different experiments with different datasets. The first experiment was conducted with simple cognitive strategies. Differ from the first experiment, the second experiment was testing region-based planning strategies. With these different strategies with different experiments, analyzed the performance of strategies, used percent above the optimal solution(PAO) which can derive to behavioral analysis and linguistic analysis [7].

One of the approaches for TSP is using heuristics. Heuristics approach constructs feasible solution and upper bound for optimal solution for route. *Karla L. Hoffman, Manfred Padberg, and Giovanni Rinaldi*, find upper bound for optimal solution and estimate the optimal solution for TSP. In order to find the upper bound for optimal solution, it is important to find a lower bound that reflects optimal solution and with lower bound that found, can prove the upper bound for the optimal solution. Therefore to find lower bound and improve it, using relaxation and if better solution for lower bound found and with found lower bound compute the upper bound to compute optimal solution for TSP[5].

The other approach for TSP that *Sharad N. Kumbharana and Gopal M. Pandey* introduces Firefly Algorithm (FA). FA is a meta-heuristic algorithm based originally designed to solve continuous optimization problems. However, the article implemented functions that are suitable to TSP such as InitialSolution() and Distance() functions. With the implemented FA for TSP, using MATHLAB2010 to produce optimal solution for TSP. The solution from the implemented FA was compared with Any Colony Optimization(ACO), Genetic Algorithm(GA), and Simulated Annealing(SA). Overall the result of FA, ACO, GA, and SA comparing from optimal solution and the performance of strategies, FA suggests better results[6].

One of the TSP typical problems is the Traveling Salesperson Problem with Hotel Selection(TSPHS). TSPHS target to find the hotel that would make the least cost to travel all cities. Usually TSPSH had been solved with MIP formulation. The article by *Luiz Henrique Barbosa and Eduardo Uchoa* introduce new solution instead of MIP formulation, which is Branch-Cut-and-Price algorithm(BCP) with several features of state-of- the-art algorithm that usually use in vehicle routing. With BCP, the article conducted an experiment and showed proposed BCP shows optimal solution for medium-sized instances and some large instances[1] .

In addition, to TSPHS has been explored and developed and the article by *Marques Moreira de Sousa, Pedro Henrique González, Luiz Satoru Ochi, and Simone de Lima Martins* introduced new approach for TSPHS which using hybrid Iterated Local Search heuristic with random variable neighborhood descent procedure. During the experiment the article compares with other Data Mining techniques and other techniques that proposed in other studies and shows the proposed algorithm does show significantly improved to optimal solution [3].

Since TSP has been developed in many different strategies, the article "TSP- Infrastructure for the Traveling Salesperson Problem" by *Michael Hahsler and Kurt Hornik* used the "TSP" package in R software. This "TSP", representing a symmetric, TSP package in R software uses distance matrix or symmetric matrix. While "ATSP", representing asymmetric TSP, uses and is represented by a square matrix. The "TSP" package uses different strategies such as Nearest neighbor algorithm, Nearest insertion, Farthest insertion, Cheapest insertion, Arbitrary insertion. With different strategies in "TSP" package, shows TSP with the given package, can be easily implement the TSP problem [4].

As TSP expands the area of using, Multiple Traveling Salesperson Problem (MTSP) develops where MTSP is targeting scheduling more than 1 sales person to visit implemented cities with optimal solution. Many of the studies regarding MTSP used Genetic algorithm (GA) with standard TSP operators with single chromosome techniques. While in "A new approach to solving the multiple traveling sales person problem using genetic algorithms' ' by *Aurther E. Carter and Cliff T. Ragsddale* proposed developed idea from previous studies, using new two-part chromosome in MTSP approach. With developed technique, the two-part chromosome algorithm shows the results with better performances compared to previous studies for MTSP [2].

3 Method

3.1 Overview of Genetic Algorithm

Genetic Algorithm (GA) is a random search method introduced by Professor Holland inspired from the laws of biological evolution [2]. In overall, GA would generate their next generation by combining two parent genes and these generated genes would run GA again and generate the next generation. As parents generate through the next generation, the gene would perform better compared to its parents becauses the next generation genes will adapt from parents which perform better compared to others. Therefore after many generations passed, the result would return the result that adapted the best performance from its parents.

3.2 Approach

Algorithm 1 Pseudocode for Genetic Algorithm

```
Create k-number of random gene into population
while n-number of generation do
    Rank population and add select superior gene with randomly selected gene
    Select superior gene and add with new gene from breed procedure
    By muted possibility, mutate gene
end while
return minimum cost and path from generated population
```

This pseudocode is a general approach for GA. From the specified k- number, number of genes in the population, GA first would randomly create genes. With the created population, GA would run generation procedure for n-time. The generation procedure would first rank or sort from top to bottom. After ranking these genes, a specified number of top rank genes would survive into the next generation and add a number of genes randomly selected from the population. After selection, GA would breed this population, while the top number of genes would survive and another number of genes would breed with superior genes and add to the next generation population. Lastly, these genes would randomly mutate according to the possibility of mutation. With this procedure, GA would return the minimum cost route and path.

The main methods of GA are breed and mutate. With these procedures, GA would select superior genes from the population by generation to generation.

Algorithm 2 Breed Function

```
procedure BREED($parent1, parent2, start$)
    child, p1, p2 = []
    ch1 = parent1[1 : len(parent1) - 1]
    ch2 = parent2[1 : len(parent2) - 1]
    apoint = random.randint(0, len(ch1))
    bpoint = random.randint(0, len(ch1))
    while apoint == bpoint do
        bpoint = random.randint(0, len(ch1))
    end while
    start = min(apoint, bpoint)
    end = max(apoint, bpoint)
    for i in range (start, end) do
        p1.append(ch1[i])
    end for
    p2 = [item for item in ch2 if item not in p1]
    child = p1 + p2
    child.insert(0, start)
    child.append(1)
return child
end procedure
```

Algorithm 3 Breed Population

```
procedure BREEDPOPULATION(population, breedrate, start)
    breedresult = []
    pool = random.ssample(population, len(population))
    save = int(len(population) * breedrate)
    for i in range(save) do
        breedreuslt.append(population[i])
    end for
    for j in range(len(population - save)) do
        child = breed(pool[j], pool[len(population) - j - 1], start)
        breedresult.append(child)
    end forreturn breedresult
end procedure
```

The breed method, with two genes named parent1 and parent2, will breed to create a child generation of gene. First, from parent1 and parent2 the breed function would copy the genes except the first and last node, which is the start node. The breed method would then select start and end index to copy from parent1 randomly using random method. With start and end index, the breed method will copy the node in parent1 within the index between start and end point to the child gene. Then other node not in the child gene would randomly copy to the child. Lastly, since TSP should start from the start node and return to the same node, add start node at index of 0 and to last.

Algorithm 4 Mutate Function

```
procedure MUTATION(gene)
    r = random.randint(1, len(gene) - 2)
    r1 = random.randint(1, len(gene) - 2)
    while r == r1 do
        r1 = random.randint(1, len(gene) - 2)
    end while
    temp = gene[r]
    gene[r] = gene[r1]
    gene[r1] = temp
return gene
end procedure
```

The mutation method will change one node in the gene. From a given gene, except the first and last node in the gene, due to the characteristic of TSP, two random values bound in index of 1 to length of path - 2 would be chosen. With this value, the node value of two indexes will interchange with each other.

Algorithm 5 Creating Next Generation

```
1: procedure NEXTGENERATION($graph, population, start, breedrate, mutationrate$)
2:     rankpop = sort(population)
3:     children = BREEDPOPULATION(population, breedrate, start)
4:     nextpopulation = []
5:     for i in range(len(childre)) do
6:         possible = random.randint(0, 100)
7:         if possible ¡ mutationrate then
8:             mutated = mutation(children[i])
9:             nextpopulation.append(mutated)
10:        else
11:            nextpopulation.append(children[i])
12:        end if
13:    end for
14: return nextpopulation
15: end procedure
```

With these given methods, GA would generate gene that would be close to the optimal route. With inputted mutation rate and breed rate, GA would breed genes that are not high ranked in the population with high ranked genes and genes will mutate according to mutation rate per generation.

4 Experiment

Before the experiment there are several assumptions to conduct. First assumption is all routes created by TSP would start from a specific node and travel all nodes once and return to the start node that is specified. The second assumption is all nodes are connected to other nodes, meaning every node is able to travel to all other nodes. Lastly, all nodes do not have a specific direction to travel, meaning if nodes x and y are connected, it is possible to travel x to y and y to x.

First experiment is solving GA with TSP and check if GA would return close to actual minimum cost. There are five different datasets to be used in the test which contains 5 nodes, 17 nodess, 26 nodes, 42 nodes and 48 nodes. Each datasets do have their own minimum cost that travel and using GA, compare how different from actual minimum cost to GA minimum cost. In this experiment, the mutate rate would be 1, equivalent to 1 percent of mutation rate and breed rate of 0.2 with 500 generations to run. In addition to get an average, run 10 tests for each dataset and average those costs. In addition, by comparing the average of costs to minimum cost, able to find how GA returns accurate minimum cost.

Second experiment is checking how the number of generations would impact the minimum cost that GA would find. To conduct this experiment, for each step to 500 when generation passes to the next generation during the GA process, find the minimum cost of each generation and record into a list and plot the point. During this experiment, the mutate rate is set to be 1 percent and breed rate of 0.2. Since GA runs in random mode, to see the average trend, 3 different GAs will run and plot into the same graph. In addition, the 26 nodes dataset used to be tested.

5 Result

5.1 Test1

The first experiment is to test if GA would return close to actual minimum cost with 5 different datasets which are containing 5 nodes, 17 nodes, 26 nodes, 42 nodes and 48 nodes

Table 1 shows the GA minimum cost of each test. The datasets used for this test contain 5 nodes in the dataset and have minimum cost of 19. Each GA test returns the minimum of 19, average of 19 took 33.16 seconds for running 10 tests.

Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9	Test10
19	19	19	19	19	19	19	19	19	19

Table 1: Minimum cost of GA with dataset containing 5 nodes

Table 2 shows the GA minimum cost with a dataset containing 17 nodes having a minimum cost of 2085. The minimum cost of GA for each test has a minimum of 2085 to maximum of 2272. Running 10 tests with 17 nodes in the dataset took 86.09 seconds.

Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9	Test10
2100	2085	2085	2155	2085	2272	2172	2132	2148	2143

Table 2: Minimum cost of GA with dataset containing 17 nodes

Table 3 shows the GA minimum cost per each test tested with a dataset contains 26 nodes having a minimum cost of 937. The minimum cost of GA for each test has a minimum of 1000 to maximum of 1169. Running 10 tests with the dataset took 121.90 seconds.

Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9	Test10
1000	1099	1169	1109	1085	1142	1025	1137	1021	1067

Table 3: Minimum cost of GA with dataset containing 26 nodes

Table 4 shows the GA minimum cost in each test tested with a dataset contains 42 nodes having minimum cost of 699. The minimum cost of GA for each test is 914 and maximum is 1302. Running 10 tests with the dataset took 185.11 seconds.

Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9	Test10
1095	1163	1175	914	952	1302	1137	1136	1076	1072

Table 4: Minimum cost of GA with dataset containing 42 nodes

Table 5 shows the GA minimum cost in each test tested with a dataset contains 48 nodes having minimum cost of 33523. GA tests return the minimum cost from 47746 to 54614 and running 10 tests with the dataset took 210.71 seconds.

Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9	Test10
51353	54614	55962	47746	48390	53720	48643	49476	53001	52082

Table 5: Minimum cost of GA with dataset containing 48 nodes

5.2 Test2

The second test is to check how the number of generations would impact the minimum cost of GA. The dataset used for this test contains 26 nodes with minimum cost of 937 and 48 nodes having minimum cost of 33523. The plot graph of cost for generation until 500 generations is recorded into Figure 1 and 2.

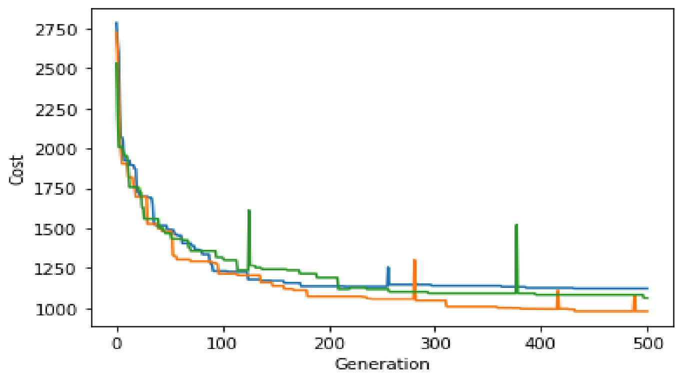

Figure 1: Minimum cost of per generation of 3 GA tests with 26 nodes.

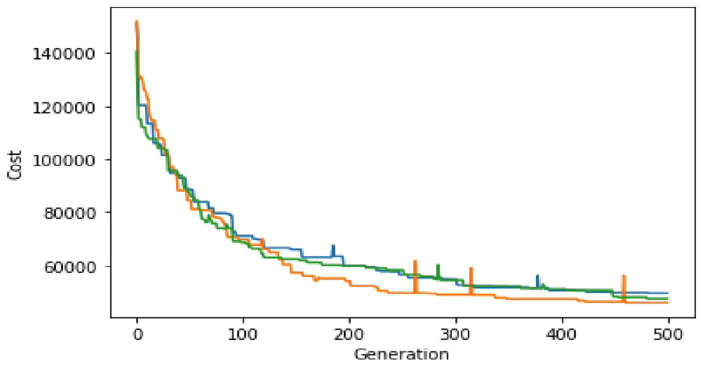

Figure 2: Minimum cost of per generation of 3 GA tests with 48 nodes.

6 Discussion

6.1 Interpreting Result

In test1, the accuracy may be affected by the number of nodes in the dataset. In test1, while testing GA with different numbers of nodes, as the number of nodes increase, the accuracy of minimum cost from GA to actual minimum cost decreases. However, since GA is based on random search, not all results were following the trend. While the GA with 42 nodes dataset shows dramatic decrease compared to other datasets that has lower node, GA with 46 nodes gives the higher accuracy compared to 42 nodes dataset and even similar accuracy compared to other lower number nodes datasets.

Nodes	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	Avg.
5	100	100	100	100	100	100	100	100	100	100	100
17	99.28	100	100	96.64	100	91.03	95.83	97.75	96.98	97.22	97.47
26	93.27	82.71	75.24	81.64	84.20	78.12	90.61	78.66	91.04	86.13	84.16
42	43.35	33.62	31.90	69.24	63.80	13.73	37.33	37.48	46.06	46.64	42.32
46	92.45	85.62	82.79	100	98.65	87.49	98.12	96.38	88.99	90.92	92.14

Table 6: Accuracy of GA with each dataset for each test and average of accuracy

In test 2, as seen in Figure 1 and 2 in page 8, as generation passes, the cost would decrease since GA would breed low rank genes with top rank genes resulting in better rank compared to previous genes. Both Figure 1 and 2 which is equivalent to a test with dataset which has 26 nodes and dataset which has 48 nodes, from 0 generation to around 200 generation, the cost would drastically decrease. However, after 200 generations, the rate of decrease decreases, and shows almost a rate of 0 or slightly higher than 0 which shows almost straight in trends. In addition, during GA process to 500 generations, shows some spikes, due to random matching and GA process may result worse compared to other generations.

6.2 Limitation of GA process

The GA randomly selects nodes from the node list into the population. From the selected population GA will rank each gene in the population and each gene randomly mutates. Since GA does most of the process randomly, the performance of GA would not be stable. Even though GA would run multiple times and by breeding and mutating, if the initial population chose that is not close to minimum path, GA would not be able to return minimum path that is actually close to optimal path.

If GA would select the initial population using another algorithm instead of random selecting, GA would generate next generations with a population that is much closer to minimum cost. The approach of GA is specialized in processing population into higher rank or to the target, but if initial population is too far from target or all of the genes in initial population is not close to target, even though all genes will generate through out number of generations, the result would not be improve since all genes in the population is not close to target and new genes would not be introduced into population.

In addition, when the number of nodes increases, the possibilities of node combinations are significantly increased, even if the number of generations is big, GA would have to take lots of generations to find a better combination of nodes to genes. Until GA generates genes that would be in higher rank, GA would have to run generations that would not be helpful to find higher rank which is unnecessary steps.

Moreover, GA in this paper has an assumption that all nodes are connected to each other and undirected. Therefore, the GA can randomly change nodes in the gene and do not have to consider whether the gene, the path, is able to visit all nodes only once and return to the start node. However if nodes are not connected to all other nodes or have direction, GA have to change or adapt other search algorithms that would be able to fit in the dataset.

7 Conclusion

TSP is the problem where from the specific node, travel all other nodes once and return to the specific node that starts. To solve TSP, GA has been adapted to solve. GA is a random search method inspired by the law of biological evolution, where from the population GA breeds with other gene in population and randomly GA would mutate in gene, changing one node to another node, to create the next generation of population into a specific number of generations to reach. The GA used to test 5 different datasets which contain 5, 17, 26, 42, and 46 nodes with mutation rate of 1 percent and breed rate of 0.2. The test shows as the number of nodes increase, the cost of path is increased or returned far from the actual minimum cost of the dataset.

In addition GA, as the number of generations increase, the minimum cost that GA returns is decreasing, specifically until 200 generation, the costs decrease dramatically and from 200 to specific number of generations, the cost is almost similar but slightly decreases. However, GA shows limitations where since initial population select randomly and GA would run inside the population and small number of new genes introduced, therefore if initial population is far from optimal path, GA would not be able to generate gene that is close to optimal path. In addition, GA is suitable for datasets that all nodes are connected to each other and undirected. Therefore, if all nodes are not connected to each other or the dataset is directed, GA would not be suitable and may consider implementing other search methods with GA.

References

[1] Luiz Henrique Barbosa and Eduardo Uchoa. A branch-cut-and-price algorithm for the traveling salesperson problem with hotel selection. *Computers & Operations Research*, 123:104986, 2020.

[2] Arthur E Carter and Cliff T Ragsdale. A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European journal of operational research*, 175(1):246–257, 2006.

[3] Marques Moreira de Sousa, Pedro Henrique González, Luiz Satoru Ochi, and Simone de Lima Martins. A hybrid iterated local search heuristic for the traveling salesperson problem with hotel selection. *Computers & Operations Research*, 129:105229, 2021.

[4] Michael Hahsler and Kurt Hornik. Tsp-infrastructure for the traveling salesperson problem. *Journal of Statistical Software*, 23(2):1–21, 2007.

[5] Karla L Hoffman, Manfred Padberg, Giovanni Rinaldi, et al. Traveling salesman problem. *Encyclopedia of operations research and management science*, 1:1573–1578, 2013.

[6] Sharad N Kumbharana and Gopal M Pandey. Solving travelling salesman problem using firefly algorithm. *International Journal for Research in science & advanced Technologies*, 2(2):53–57, 2013.

[7] Thora Tenbrink and Jan Wiener. The verbalization of multiple strategies in a variant of the traveling salesperson problem. *Cognitive processing*, 10(2):143–161, 2009.