MEF UNIVERSITY

Your Freedom in Learning

TRAVELING SALESMAN PROBLEM WITH GENETIC ALGORITHM

TERM PROJECT REPORT

IE 307

FALL 2019-2020

Barkın Börk
Beytullah Oğuz Akgün
Beyza Alhan
Derya Çırakoğlu
Doruk Can Güngördü
Elif Selin Acet
Emre Özelsağıroğlu
Kağan Eroğlu

EXECUTIVE SUMMARY

This report is related to our term project on genetic algorithm which is one of the heuristic methods that we use to solve the (NP-hard) traveling salesman problem which is one of the combinational optimization problems.

The report consists of five sections; a brief description of the problem, a literature search, an explanation of the algorithms and codes, the evaluation of the result, and the distribution of tasks of the group members.

In our report, we aimed to explain how we create the first population, how we do elimination, how parents are identified, how we do the crossover operator and mutation with the genetic algorithm we use.

Key Words: Traveling salesman problem, genetic algorithms, heuristic methods.

TABLE OF CONTENTS

.

1. INTRODUCTION

The Traveling Salesman Problem is one of the combinational optimization problems. The solution to this problem is very difficult and is among the NP-hard problems in the literature. It can be applied in areas such as planning and logistics in the sector.

Genetic algorithms are one of the methods used to solve combinational optimization problems and it is an heuristic algorithm. Genetic algorithms are an optimization method similar to the evolution process in biological systems. This algorithm is preferred because it leads us to faster and near-optimal results.

We have a list of 3 cities with distances between them. We go back to the city where we started with the condition of stopping each city only once with the traveling seller problem. And it aims to find the shortest path during all operations. In this project, we aimed to find a solution to this TSP problem with the help of genetic algorithms, an intuitive method. We realized this solution with the code we developed using Phyton program. With this result, we tried to show that more efficient routes can be determined by using this method to save time and cost.

2. LITERATURE REVIEW

2.1. Traveling Salesman Problem

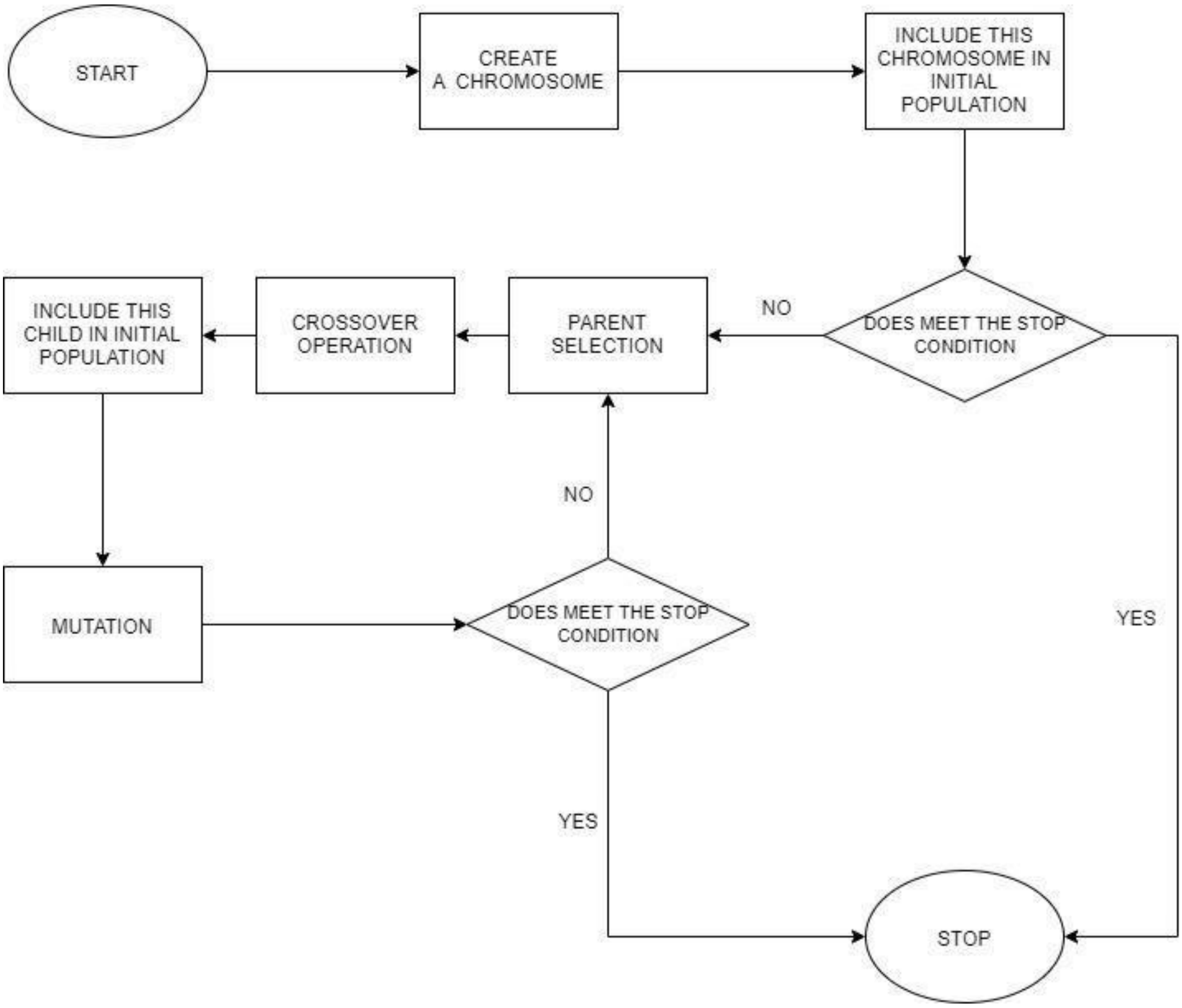
Gupta and Panwar (2013) examine the Solving Traveling Salesman Problem Using Genetic Algorithm. The traveling salesman problem (TSP) is a combinatorial optimization problem. NP is difficult and is the most studied problem in the field of TSP optimization. However, with the increase in the number of cities, the complexity of the problem continues to increase. In this paper, he solved the Travel Salesman Problem using the Genetic algorithm approach. The system starts from a matrix of Euclidean distances calculated between the cities to be visited by the traveling vendor and the randomly selected city order as the first population. Then, when the stop criterion is reached, new generations are created repeatedly until the appropriate path is reached.

2.2. Tournament Selection

Razali and Geraghty (2011) examined Genetic Algorithm Performance with Different Selection Strategies in Solving TSP. The genetic algorithm (GA) has several genetic operators that can be modified to improve the performance of certain applications. These operators include parental selection, crossover and mutation. Selection is one of the important processes in the GA process. There are multiple ways to choose from. This article presents a comparison of GA performance in the solution of the traveling vendor problem (TSP) using different selection strategies. Several examples of TSP have been tested and show that the tournament selection strategy outperforms proportional roulette and sequential roulette selections and achieves the best solution quality with low calculation times.

3. SOLUTION METHOD

3.1Coding Scheme



3.2 Creating Initial Population

Initial Population is selected by Random Initialization. Population size N = 5000.

First, we've created a list of all the cities in the files. then we chose a random city from this list. We deleted this city from the list we created and added it to our new chromosome. The reason we do this is to avoid re-selecting the same city. When we placed all cities on these chromosomes we created, we added this chromosome to our population because the process was completed. And we reset everything and go back to the beginning of the loop. We stopped this loop when we created 5000 chromosomes.

3.3 Parent Selection

Parental selection is very important for the convergence rate of GA, as good parents lead individuals to a better and more fit solution. Tournament Selection is also extremely popular in literature. In K-Way tournament selection, we select K=10 individuals from the population at random and select the best out of these to become a parent. The same process is repeated for selecting the next parent. We selected 10 random chromosomes from the population. Among these chromosomes, we identified parents with the least length. We identified the same process as second parents.

3.4 Crossover Operator

Initially we identified a children's cluster, a neighboring cluster, an unused cluster of cities, and an element representing the current city. We equalized the first city in the first parent to the current city element. We then assigned this element to the children's set. We looked at the neighbors in this city element we assigned to parents 1 and 2. These neighbors are also in the neighboring cluster. We checked whether any of the elements in the neighboring cluster were in the children's cluster. If this element exists in both clusters, we have removed this element from the neighboring cluster. Because the same element in the child set can not be more than one. Then, if there is an element in the neighboring set, we have assigned the element closest to the last element in the child set to the child set. And we removed this element from the unused cities set. If there are no elements in the neighbor set, we have assigned the closest element to the child set from the unused cities set and removed it from the unused cities set. We have replaced the element that represents the current city with the last element we added to the children's set. We continued this cycle until the set of unused cities was empty. Because we have to use all cities.

3.5 Mutation

The mutation allows the algorithm to avoid local optimum. The mutation diversifies the search area and protects against loss of genetic material caused by crossing. The mutation process is applied to a certain percentage of children, not all of the children who are formed as a result of the cross. This is called the mutation rate. The mutation rate is expressed as Pm. In the swap mutation, two random genes on the chromosome are selected and their values are exchanged. This is common in permutation-based coding. We determined Pm=0.01.

3.6 Stop Condition

The path length of each child is calculated after the child starts creating. Based on the initial route, newly created children are controlled. If there is no reduction in the last 5000 steps, the process stops. If a new reduction occurs, the process starts all over again and the cycle continues in this way.

4. COMPUTATIONAL ANALYSIS

4.1. Experimental Results

Instance 1

For 29 cities data firstly we take 5000 initial population and our stop condition is 1000. In parent choice we did a tournement selection and we said to n is 10.

Run Time	Path Length
5.58	2108
7.5	2113
7.08	2082
6	2080
5.72	2094
5.6	2093
5.48	2090
6.19	2142
7.66	2094
6.08	2104
Average Run time	6.289
Max Path Length	2080
Min Path Length	2142

We only change initial population to 10000 for looking the changes

Run Time	Path Length
5.85	2223
7.93	2125
7.52	2176
8.3	2092
8.09	2121
Average Run time	7.538
Max Path Length	2223
Min Path Length	2092

We want to observing what gives if we gave 2500 to initial population.

Run Time	Path Length
6	2117
4.03	2123
4.65	2107
4.76	2112
6.1	2042
Average Run time	5.108
Max Path Length	2123
Min Path Length	2042

When we look our comparing scheme time, shortest path and creating child gave best solution when we gave the 5000 to initial population. We tried for find a better condition doing 5000 stop condition and our initial population.

Run Time	Path Length
17.37	2085
14.91	2092
15.25	2085
17.97	2085
12.26	2066
15.2	2085
19.94	2085
13.87	2085
18.59	2085
16.7	2085
Average Run time	16.206
Max Path Length	2066
Min Path Length	2092

Instance 2

For 42 cities data firstly we take 5000 initial population and our stop condition is 1000. In parent choice we did a tournement selection and we said to n is 10.

Run Time	Path Length
5.18	780
4.15	772
5.14	784
4.01	757
8.21	754
4.35	768
4.38	755
6.13	758
4.51	760
4.47	759
Average Run time	5.053
Max Path Length	784
Min Path Length	754

We decrease the initial population to 10000.

Run Time	Path Length
4.75	785
5.53	814
7.6	771
6.79	796
8.68	789
Average Run time	6.67
Max Path Length	814
Min Path Length	771

We gave 5000 stop condition and initial population.

Run Time	Path Length
22.1	731
16.15	738
8.98	739
16.11	732
17.95	743
Average Run time	16.258
Max Path Length	743
Min Path Length	731

When we looked this values if we change the stop condition not change our solution too much.Now we gave 10000 stop condition and 5000 to initial population

Run Time	Path Length
43.9	709
18.74	716
18.78	730
27.09	702
37.52	716
Average Run time	29.206
Max Path Length	730
Min Path Length	702

Instance 3

For 76 cities data firstly we take 5000 initial population and our stop condition is 1000. In parent choice we did a tournement selection and we said to n is 10.

Run Time	Path Length
10.5	594
11.6	585
15.17	578
15.29	574
10.8	581
15.5	570
21.5	559
12.85	579
16.14	561
18.07	568
Average Run Time	14.742
Max Path Length	594
Min Path Length	559

We gave 10000 to initial population.

Run Time	Path Length
23.06	562
20.89	588
13.72	623
19.7	597
16.6	587
Average Run Time	18.794
Max Path Length	623
Min Path Length	562

We gave 10000 to stop condition and 5000 to initial population.

Run Time	Path Length
47.12	556
68.59	557
47.34	562
68.55	562
50.35	562
Average Run Time	56.39
Max Path Length	562
Min Path Length	556

We gave 10000 both of them stop condition and initial population.

Run Time	Path Length
101.144	558
54.39	571
75.2	558
69.68	561
83.65	558
Average Run Time	76.8128
Max Path Length	571
Min Path Length	558

4.2. Interpretation of the Results

Instance 1

When we looked the 29 cities data shortest path is coming when we gave 2500 to initial population and our stop condition is 1000 when we gave these numbers our route is [27,24,8,1,28,6,12,9,5,26,29,3,2,21,20,10,13,16,19,4,15,18,14,17,22,11,25,7,23] Creating Children: 3337

Run time : 6,10

Shortest Path: 2042

When we observe all above we can show the worst shortest path is coming when initial population is 10000 and stop condition is 1000. Our route like this [29,3,26,5,9,12,6,1,21,28,24,27,16,19,4,15,18,14,17,22,11,25,7,23,8,13,10,20,2] Creating Children: 1623

Run time : 5,85

Shortest Path: 2223

So we should make the stop condition 1000 and initial population 2500.

Instance 2

When we looked the 42 cities data shortest path is coming when we gave 5000 to initial population and our stop condition is 10000 when we gave these numbers our route is [14. 15. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 1. 42. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 17. 16.]

Toplam oluşturlan çocuk sayısı : 21662

Run time : 27.09179629999994

Shortest Path : 702

When we observe all above we can show the worst shortest path is coming when initial population is 10000 and stop condition is 1000. Our route like this [38. 37. 35. 34. 36. 32. 33. 31. 30. 28. 29. 21. 22. 23. 17. 16. 18. 19.20. 15. 14. 13. 11. 12. 10. 25. 26. 27. 24. 9. 8. 7. 6. 5. 4. 3. 1. 41. 42. 2. 39. 40.]

Creating Child : 2330

Run time : 5.53067759999999

Shorthest path:814

So we should make the stop condition 10000 and initial population 5000.

Instance 3

When we looked the 76 cities data shortest path is coming when we gave 5000 to initial population and our stop condition is 10000 when we gave these numbers our route is [41. 43. 42. 64. 22. 61. 69. 36. 71. 60. 70. 20. 37. 5. 15. 57. 13. 54. 19. 8. 46. 34. 52. 27. 45. 29. 48. 47. 21. 4. 30. 2. 74. 28. 62. 73. 1. 33. 63. 16. 3. 44. 32. 9. 39. 72. 58. 12. 40. 17. 51. 6. 68. 75.76. 67. 26. 7. 35. 53. 14. 59. 11. 66. 65. 38. 10. 31. 55. 25. 50. 18.24. 49. 23. 56.]

Creating Child : 15754

Run time : 47.124235999999655

Shortest Path:556

So we should make the stop condition 1000 and initial population 10000.

Lower Bound

We find the lower bound with using Phyton. For instance 1, lower bound is 1719. For instance 2, lower bound is 591. For instance 3, lower bound is 474.

REFERENCES

https://www.researchgate.net/publication/280978365_Solving_Travelling_Salesman_Problem_Using_Genetic_Algorithm

https://pdfs.semanticscholar.org/f5d7/e26f21603c13dfe8e412106127039a547edb.pdf?_ga=2.225282202.1480650305.1578580371-1096437014.1576176262