➔ Regular Research Paper – NS

Solving the Traveling Salesman Problem with Greedy Genetic Algorithm

Recep Colak
Department of Computer Technologies, Isparta Applied Sciences University, Isparta, Turkey
recepcolak@isparta.edu.tr

Abstract

The traveling salesman problem (TSP) is an NP-hard problem that is difficult to solve. TSP is a general problem with many sub-branches such as vehicle routing, cargo distribution, circuit element placement for electronic cards, school busses. The problem can be solved in *N!* time by trying all possibilities. When the number of points to be visited is small, precise methods can be used, while when the number of points is high, a solution in an acceptable time is not possible with precise methods. When the number of points is large, the problem is solved with heuristic algorithms that give solutions close to the optimum solution. Genetic Algorithm (GA), which is an heuristic method, is a frequently used method in solving the TSP problem. In this study, the effect of assigning the initial chromosomes of GA with Greedy Heuristic (GH) on the solution was investigated. In the Greedy approach, the first point is randomly selected, the next visit point is selected, and the closest point to the last selected point is selected and the tours are completed. The obtained results showed that the initial routes were assigned with GH instead of random assignment, giving better solutions. In addition, fewer chromosomes were needed in the solutions obtained with GH, thus shortening the runtime.

Keywords: *Tsp, genetic algorithm, greedy heuristic, initial route*

1. INTRODUCTION

The traveling salesman problem (TSP), which is a combinatorial optimization problem, is that a seller can visit n cities in the shortest path. The problem to be solved can be defined as a graph consisting of cities and the edges that connect them. In this case, the shortest Hamilton tour in which each point is visited only once gives the solution [1]. In TSP, which is easy to define and the solution algorithm, the real difficulty arises in time complexity. The time complexity of the TSP is calculated as *N!* for N points. When the number of points is low, the shortest tour can be easily found by trying all possibilities. When the number of points increases, the time required to try all possibilities increases and turns into an NP-hard problem. Since the solution of the problem depends on the data, precise methods [2], [3], heuristics [4]–[6] and hybrid methods [7] have been tried to solve the problem.

Genetic Algorithms (GA) is a probability-based heuristic algorithm that operates on the principle of survival of the best in the theory of evolution. GA has been frequently used for solving NP-hard problems in acceptable time [8]–[10]. In this study, a Greedy approach is proposed to solve the GA and TSP problem. Greedy algorithms are iterative algorithms that seek a solution by choosing the most optimal one among the available options [11]. In this study, the initial routes were assigned with a Greedy Heuristic (GH) algorithm instead of being randomly generated. In this way, better results can be obtained by searching closer points to the solution instead of searching the entire solution space. Searching at points closer to the solution instead of the entire solution space also allowed to obtain a solution with less chromosome number. Using less number of chromosomes provided positive results in solution time. In the second part of the study, the literature summary, the method used in the third part, and the findings obtained in the fourth part are given. In the last section, the obtained results are discussed.

2. RELATED WORKS

TSP was first introduced by the 18th century Irish mathematician Hamilton [12]. The problem, which was expressed mathematically for the first time in the 1930s, is generally classified as symmetrical, asymmetrical and multiple salesman problem [13]. In symmetric problems, the distance between cities A-B and B-A are considered the same. In asymmetrical problems, these two directions can be of different lengths. Especially when the points in the city are taken into consideration, the points to be visited have an asymmetrical structure due to the intersections and opposite directions. In multiple salesman problems, the points to be visited are made by more than one salesman in different tours. TSP has been applied to different areas such as vehicle routing [14], school bus routing [15], cargo distribution [16], pick and distribute problems [17], circuit element layout for electronic printed circuit boards [18] Researchers have proposed different algorithms for the solution of TSP. Since TSP is NP-Hard, heuristic methods are preferred instead of exact methods [19], [20].
Nurdiawan et al. [21] used GA for the shortest way to make tourist visits. In their study, they observed that crossover and mutation operators affect the result obtained. Instead of the standard GA crossover of two ancestors, Roy et al. [22] attempted to cross four ancestors. In comparison to the crossover with two ancestors, they discovered that the new crossover method produced worse results. Gomes et al. used GA to solve the product allocation problem of a food and beverage service organization as a multi-vendor TSP. They created separate routes by dividing the points to be visited in the city into clusters, and the results showed that the total distance traveled was shortened and customers reached the products earlier [23]. When the literature is examined, it becomes clear that GA has been used to solve TSP successfully. In contrast to other studies, this study applies GA to the problem, but the initial routes are chosen based on a predetermined rule. In this study, crossover, mutation, and elitism operators are used to optimize the routes after they are initially generated by a greedy heuristic algorithm.

3. MATERIAL METHOD

3.1. Problem Defination

TSP, which is an NP-hard problem, can be solved by graph theory. The objective of the problem is to find the shortest Hamiltonian path. In this study, asymmetric TSP is considered when the distance between two cities can be of different lengths. In the problem, the cities are considered as points, and the roads connecting the cities are considered as edges. An example graph structure is given in **Figure 1**. In the graph in **Figure 1**, two-way arrows show the distance between two cities. One-way arrows are the distance from one city to another city. In an asymmetric TSP, the distances between two cities may be different due to reasons such as reversed directions or intersections. In the graph in **Figure 1**, the set of cities are the vertices and the connections between cities are the edges.

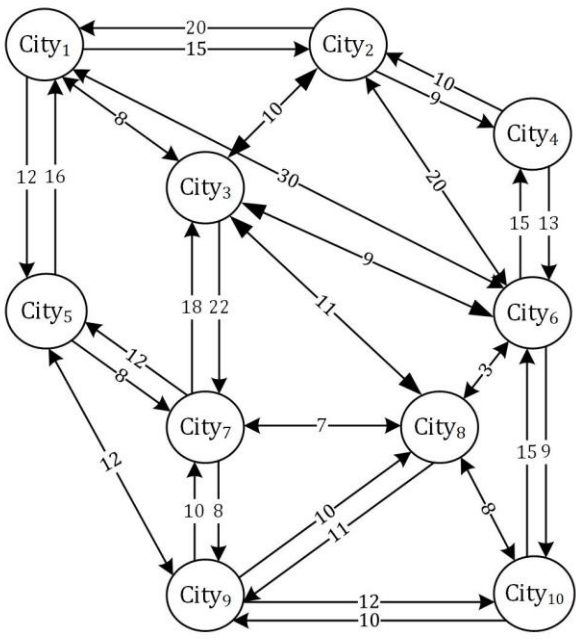


Figure 1. Example graph for asymmetric TSP

The graph in Figure 1 can be expressed as the set $G = (V, A)$, where V denotes the cities and A denotes the connections between the cities. According to the above graph, our mathematical model is expressed below.

$V: \{1..n\}$, is the set of points (Cities) to be visited.

$A: \{(i, j); i, j \in V; i \neq j\}$, A_{ij} V_i, V_j is a set of vertices (path) between points.

c_{ij}: A_{ij} is the distance value for the edge. It can be Ecludian distance or real path distance value.

$R: \{1..n\}$, The array obtained by permutation of V. k is the array index. (if $R_k = 5$, $R_{k+1} = 8$, it means that the visit to the 8th point after the 5th point has taken place)

$$x_{ij} \begin{cases} 1, & if\ a\ visit\ is\ made\ from\ i\ to\ j \\ 0, & else \end{cases}$$

Objective function:

$$Minimize\ Z = \left(\sum_{k=1}^{n-1} \boldsymbol{c_{R_k\, R_{k+1}}}\right) + \boldsymbol{c_{R_n\, R_1}} \tag{1}$$

Consraints:

$$R_k \in \{R\}, \sum_{k=1}^{n-1} x_{R_k\, R_{k+1}} = 1 \tag{2}$$

In the above model, V is the set of points to be visited and A is the edges connecting these points. R is an array consisting of a complete tour using all the elements in the set V. Equation (1) finds the array with the shortest tour, and constraint (2) ensures that each point is visited only once.

3.2. Genetic Algorithms

GA is an evolutionary optimization algorithm first proposed by John Holland in the 1970s [24]. It is an algorithm that mimics natural evolution processes in order to achieve the best possible result. With GA, acceptable solutions to NP-Hard problems can be produced in a short time. GA is an iterative algorithm that allows the initial solutions to evolve to produce better results through crossover, mutation, and selection. The flowchart showing the operation of GA is given in **Figure 2**.

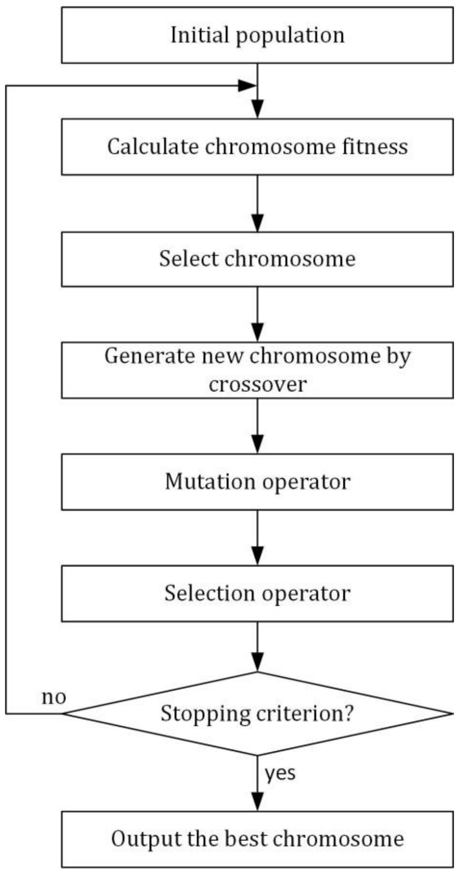


Figure 2. The flow chart of the genetic algorithm

In GA, each representative solution is called a chromosome. Depending on the type of problem, chromosomes can be coded in different ways. Since the solution in TSP consists of elements from a certain set, chromosomes in these problems are coded as permutations of the points to be routed. Different chromosome coding examples are shown in **Figure 3** In this study, the chromosome structure is defined as a one-dimensional array. The size of the array used depends on the number of cities to be routed.

The cities in the array show the points to be visited in order. It is assumed to return from the last point of the array to the first assigned point of the array. For example, in a sequence such as 1234, it means that a visit route is created from city 1 to city 2, from city 2 to city 3, from city 3 to city 4, and from city 4 to city 5. Array size changes dynamically according to the problem size.

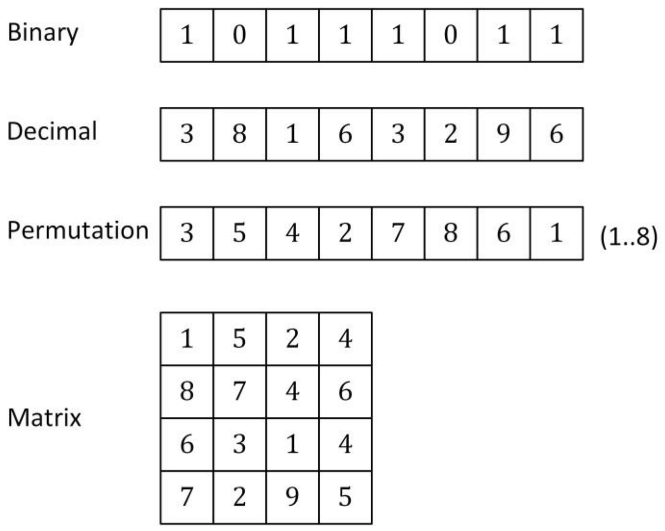


Figure 3. Chromosome coding examples

3.2.1. Generation of initial routes with greedy heuristic

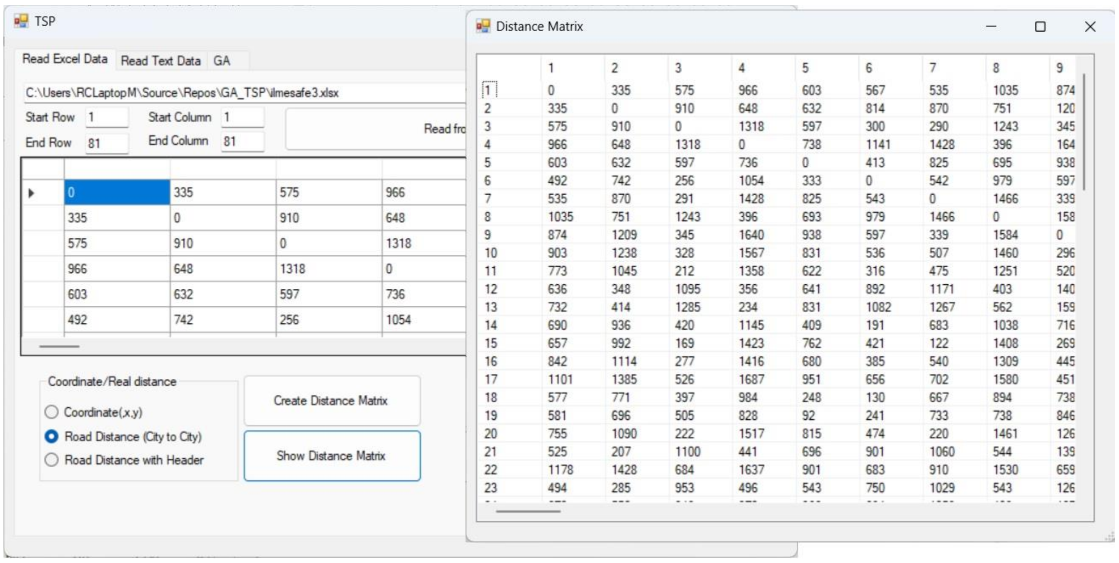
In GA, initial routes are an important factor affecting the result. In this study, unlike other studies in the literature, the initial routes were created using a tour constructor method. The constructor methods start the solution with chromosomes assigned according to certain rules instead of randomly assigned chromosomes. In this way, the GA solution can start working with short routes. In this study, GH is used as a tour constructor method. The tour constructor algorithm used is given below.

Algorithm 1

```
route=null
unvisited_city←cites
current_city← Rand(unvisited_city)
route.Add(current_city)
unvisited_city.Remove(current_city)
while (unvisited_city!=null)
   min_distance=max_number
   for city in unvisited_city do
      if (distance(current_city,city)<min_distance)
         new_city=city;
         min_distance= distance(current_city,city)
      end if
      route.Add(new_city)
      unvisited_city.Remove(new_city)
   end for
end while
```

After the starting point of the route generated by the above algorithm is randomly selected, the nearest city is selected for the next point. Although the tour distance of the routes obtained in this way is shorter than random tours, it is not the shortest tour. These routes are optimized with GA operators to obtain shorter tours. An application was developed in the Visual C# environment to measure the effect of GA, or random assignment, of initial routes on TSP.Tests were carried out with the developed application. An application was developed in Visual C#

environment to measure the effect of GA, or random assignment, of initial routes on TSP. With the developed application, tests were carried out with the data read from the data set **Figure 4** Figure 4 shows the data reading interface of the application.



C:\Users\RCLaptopM\Source\Repos\GA_TSP\ilmesafe3.xlsx

Start Row 1 Start Column 1
End Row 81 End Column 81

0	335	575	966
335	0	910	648
575	910	0	1318
966	648	1318	0
603	632	597	736
492	742	256	1054

Coordinate/Real distance
Coordinate(x,y) [✓ Road Distance (City to City)] Road Distance with Header

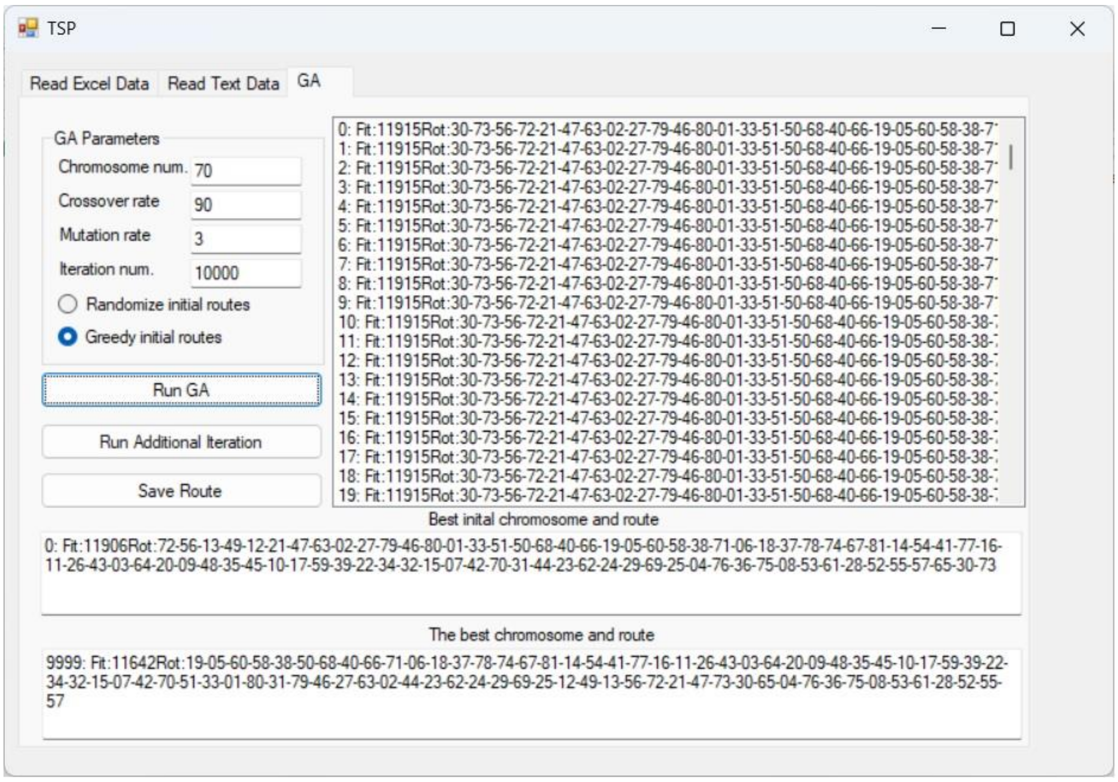
Create Distance Matrix
Show Distance Matrix

	1	2	3	4	5	6	7	8	9
1	0	335	575	966	603	567	535	1035	874
2	335	0	910	648	632	814	870	751	120
3	575	910	0	1318	597	300	290	1243	345
4	966	648	1318	0	738	1141	1428	396	164
5	603	632	597	736	0	413	825	695	938
6	492	742	256	1054	333	0	542	979	597
7	535	870	291	1428	825	543	0	1466	339
8	1035	751	1243	396	693	979	1466	0	158
9	874	1209	345	1640	938	597	339	1584	0
10	903	1238	328	1567	831	536	507	1460	296
11	773	1045	212	1358	622	316	475	1251	520
12	636	348	1095	356	641	892	1171	403	140
13	732	414	1285	234	831	1082	1267	562	159
14	690	936	420	1145	409	191	683	1038	716
15	657	992	169	1423	762	421	122	1408	269
16	842	1114	277	1416	680	385	540	1309	445
17	1101	1385	526	1687	951	656	702	1580	451
18	577	771	397	984	248	130	667	894	738
19	581	696	505	828	92	241	733	738	846
20	755	1090	222	1517	815	474	220	1461	126
21	525	207	1100	441	696	901	1060	544	139
22	1178	1428	684	1637	901	683	910	1530	659
23	494	285	953	496	543	750	1029	543	126

Figure 4. Data reading and distance matrix

A distance matrix was created from the data read from the file in accordance with the asymmetric TSP. Initial routes were assigned in two different types with GH and random assignment and tests were performed. The screen developed for the test can be seen in

Figure 5.



GA Parameters
Chromosome num. 70
Crossover rate 90
Mutation rate 3
Iteration num. 10000
Randomize initial routes [✓ Greedy initial routes]

```
0: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-7
1: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-7
2: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-7
3: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-7
4: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-7
5: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-7
6: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-7
7: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-7
8: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-7
9: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-7
10: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-
11: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-
12: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-
13: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-
14: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-
15: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-
16: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-
17: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-
18: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-
19: Fit:11915Rot:30-73-56-72-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-
```

Best inital chromosome and route

```
0: Fit:11906Rot:72-56-13-49-12-21-47-63-02-27-79-46-80-01-33-51-50-68-40-66-19-05-60-58-38-71-06-18-37-78-74-67-81-14-54-41-77-16-
11-26-43-03-64-20-09-48-35-45-10-17-59-39-22-34-32-15-07-42-70-31-44-23-62-24-29-69-25-04-76-36-75-08-53-61-28-52-55-57-65-30-73
```

The best chromosome and route

```
9999: Fit:11642Rot:19-05-60-58-38-50-68-40-66-71-06-18-37-78-74-67-81-14-54-41-77-16-11-26-43-03-64-20-09-48-35-45-10-17-59-39-22-
34-32-15-07-42-70-51-33-01-80-31-79-46-27-63-02-44-23-62-24-29-69-25-12-49-13-56-72-21-47-73-30-65-04-76-36-75-08-53-61-28-52-55-
57
```

Figure 5. Application software

4. FINDINGS

In order to measure the effect of generating the initial routes with GH on the solution, tests were performed with the same data. **Figure 6** shows the effect of GH and random assignment for a dataset of 81 cities [25].

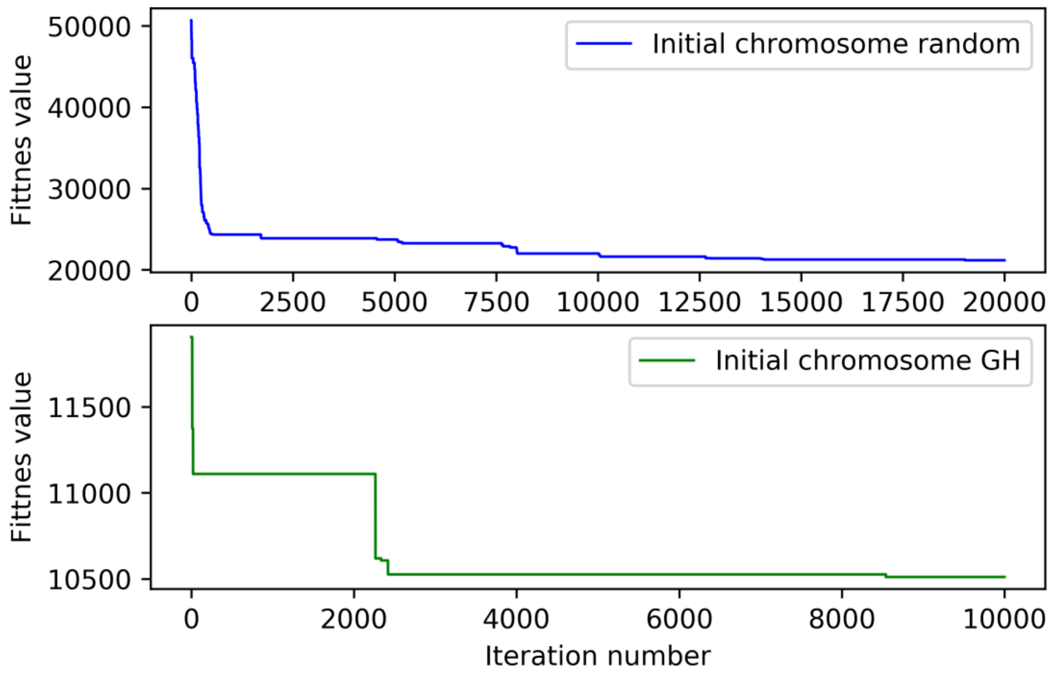


Figure 6. The effect of GH and Random assignment of initial routes on the solutions

In order to understand the effect of assigning initial routes with GH on the solution, TSPLIB [26] data, which is frequently used for comparison in the literature, was used. The same data was run 20 times with both methods to measure the effect of initial chromosomes. The best results obtained can be seen in Table 1.

Table 1. Effect of initial chromosomes as GH or Random

Test Data	Nuber of Cities	Best	GH (Suggested)	Random	GH Route
br17	17	39	39	39	15-06-07-16-04-05-08-09-17-01-12-03-14-13-02-11-10
ftv33	34	1286	1367	1515	34-31-05-07-06-03-04-01-14-13-10-33-08-09-11-12-32-18-19-20-21-22-23-25-24-27-28-29-30-26-17-15-16-02
ftv35	36	1473	1519	1660	34-19-20-18-11-10-35-09-14-01-17-16-15-12-13-06-08-07-05-33-36-03-04-02-27-26-25-21-23-29-30-32-31-28-24-22

Table 1 continues on the next page.

Table 2. Effect of initial chromosomes as GH or Random (continued)

Test Data	Nuber of Cities	Best	GH (Suggested)	Random	GH Route
kro124	100	36230	40815	42057	57-12-07-09-87-51-61-81-25-73-50-44-02-64-54-40-69-68-85-30-96-78-05-52-37-33-13-76-95-82-39-48-14-41-100-71-03-43-46-29-34-55-27-86-35-62-60-23-45-32-15-17-11-59-74-72-10-84-38-24-36-99-21-47-91-98-77-58-28-93-01-08-92-63-06-49-90-53-16-22-94-18-79-88-70-65-66-04-26-97-75-19-56-42-80-31-89-67-20-83
Ftv64	65	1839	2151	2989	31-54-32-63-53-52-25-26-64-30-33-55-29-28-27-22-49-50-23-60-24-36-21-20-47-46-11-61-16-01-62-02-38-04-15-48-40-37-09-41-42-10-43-12-13-44-45-08-06-07-58-59-35-57-65-39-03-05-14-17-18-19-51-56-34

4. CONCLUSION

In this study, the TSP problem, which is a well-known NP-hard problem, is discussed. In solving the problem with GA, the effect of the initial chromosomes on the result was investigated. The obtained results show that the assignment of chromosomes with GH gives better results instead of random assignment. When the initial chromosomes are assigned with GH, GA searches closer to the solution and gives better results. The tests were repeated for different chromosome numbers and different iterations. If the initial chromosomes were random, the increase in the number of chromosomes provided better results. When the first routes were created with GH, it was observed that acceptable results were obtained with less chromosome number. These results show that establishing the initial routes with GH provides both results with fewer chromosomes and better solutions.

REFERENCES

[1] V. Cacchiani, C. Contreras-Bolton, L. M. Escobar-Falcón, and P. Toth. (2023). A matheuristic algorithm for the pollution and energy minimization traveling salesman problems. International Transactions in Operational Research, vol. 30, no. 2, pp. 655–687.

[2] R. M. Lusby, J. Larsen, M. Ehrgott, and D. Ryan. (2010). An exact method for the double TSP with multiple stacks. International Transactions in Operational Research, vol. 17, no. 5, pp. 637–652.

[3] N. M. Yunos, A. Shurbevski, and H. Nagamochi. (2016). A polynomial-space exact algorithm for TSP in degree-6 graphs. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9943 LNCS, pp. 228–240.

[4] L. Shen and B. W. Kernighan. (1973). An effective heuristic algorithm for the traveling-salesman problem. Operations Research, vol. 21, no. 2, pp. 498–516.

[5] N. Rokbani et al..(2021). Bi-heuristic ant colony optimization-based approaches for traveling salesman problem. Soft Computing, vol. 25, no. 5, pp. 3775–3794.

[6] A. Uğur, S. Korukoğlu, A. Çalişkan, M. Cinsdikici, and A. Alp. (2009). Genetic algorithm based solution for TSP on a sphere. Mathematical and Computational Applications, vol. 14, no. 3, pp. 219–228.

[7] C. A. R. Jahuira and E. C. Vargas. (2002). Hybrid genetic algorithm with exact techniques applied to TSP. in In Second international workshop on Intelligent systems design and application, pp. 110–124.

[8] M. Mahjoob, S. S. Fazeli, S. Milanlouei, L. S. Tavassoli, and M. Mirmozaffari. (2022). A modified adaptive genetic algorithm for multi-product multi-period inventory routing problem. Sustainable Operations and Computers, vol. 3, pp. 1–9.

[9] E. Messaoud. (2022). Solving a stochastic programming with recourse model for the stochastic electric capacitated vehicle routing problem using a hybrid genetic algorithm. European Journal of Industrial Engineering, vol. 16, no. 1, pp. 71–90.

[10] A. Maskooki, K. Deb, and M. Kallio. (2022). A customized genetic algorithm for bi-objective routing in a dynamic network. European Journal of Operational Research, vol. 297, no. 2, pp. 615–629.

[11] V. Chvatal. (1979). A greedy heuristic for the set-covering problem. Mathematics of operations research, vol. 4, no. 3, pp. 233–235.

[12] N. L. Biggs, E. K. Lloyd, and R. J. Wilson, Graph Theory 1736-1936. New York: Oxfort University Press, 1986.

[13] Y. Şahin, K. Karagül. (2019). Gezgin satıcı probleminin melez akışkan genetik algoritma (MAGA) kullanarak çözümü. Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi, 25(1), 106-114. vol. 25, no. 1, pp. 106–114.

[14] K. Karagul, Y. Sahin, E. Aydemir, and A. Oral. (2019). A simulated annealing algorithm based solution method for a green vehicle routing problem with fuel consumption. International Series in Operations Research and Management Science, vol. 273, pp. 161–187.

[15] T. Yiğit, Ö. Ünsal, and Ö. Deperlioğlu. (2018). Using the metaheuristic methods for real-time optimisation of dynamic school bus routing problem and an application. International Journal of Bio-Inspired Computation, vol. 11, no. 2, pp. 123–133.

[16] Atmaca, E. (2012). Bir kargo şirketinde araç rotalama problemi. Tübav Bilim Dergisi, 5(2), 12-27.

[17] D. Schubert, H. Kuhn, and A. Holzapfel. (2021). Same-day deliveries in omnichannel retail: Integrated order picking and vehicle routing with vehicle-site dependencies. Naval Research Logistics, vol. 68, no. 6, pp. 721–744.

[18] Y. Wang. (2019). PCB Drill Path Optimization by Improved Genetic Algorithm. 5th International Conference on Control, Automation and Robotics. ICCAR 2019, 2019, pp. 744–748.

[19] D. Messon, D. Verma, M. Rastogi, and A. Singh. (2022). Comparative Study of Time Optimization Algorithms for Traveling Salesman Problem. Lecture Notes in Networks and Systems, vol. 392, pp. 555–566.

[20] J. Wang et al..(2022). A Carnivorous Plant Algorithm With Heuristic Decoding Method for Traveling Salesman Problem. IEEE Access, vol. 10, pp. 97142–97164.

[21] O. Nurdiawan, F. A. Pratama, D. A. Kurnia, Kaslani, and N. Rahaningsih. (2020). Optimization of Traveling Salesman Problem on Scheduling Tour Packages using Genetic Algorithms. Journal of Physics: Conference Series, vol. 1477, no. 5, pp. 141–151.

[22] A. Roy, A. Manna, and S. Maity. (2019). A novel memetic genetic algorithm for solving traveling salesman problem based on multi-parent crossover technique, Decision Making: Applications in Management and Engineering, vol. 2, no. 2, pp. 100–111.
[23] D. E. Gomes, M. I. D. Iglésias, A. P. Proença, T. M. Lima, and P. D. Gaspar. (2021). Applying a genetic algorithm to a m-tsp: Case study of a decision support system for optimizing a beverage logistics vehicles routing problem. Electronics (Switzerland), vol. 10, no. 18.

[24] H. Cui, J. Qiu, J. Cao, M. Guo, X. Chen, and S. Gorbachev. (2023). Route optimization in township logistics distribution considering customer satisfaction based on adaptive genetic algorithm. Mathematics and Computers in Simulation, vol. 204, pp. 28–42.

[25] Kara Yolları Genel Müdürlüğü. (2023). https://www.kgm.gov.tr/SiteCollectionDocuments/KGMdocuments/Root/Uzakliklar/ilmesafe.xlsx
Date of Access: 15/12/2022

[26] Http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/atsp/
Date of Access: 15/12/2022