DEÜ FMD 24(71), 665-677, 2022

Dokuz Eylül Üniversitesi Mühendislik Fakültesi Fen ve Mühendislik Dergisi
Dokuz Eylul University Faculty of Engineering Journal of Science and Engineering

Basılı/Printed ISSN: 1302-9304. Elektronik/Online ISSN: 2547-958X

Gezgin Satici Problemi Için Yeni Bir Sezgisel:maxS

A Novel Heuristic For The Traveling Salesman Problem: maxS

Kenan Karagül [1*], Gürhan Gündüz [2]

[1] Pamukkale University, Department of Logistics, Denizli, Turkey
[2] Muğla Sıtkı Koçman University, Computer Engineering Department, Muğla, Turkey

Sorumlu Yazar / Corresponding Author *: gurhangunduz@mu.edu.tr

Geliş Tarihi / Received: 05.11.2021 *Araştırma Makalesi/Research Article*
Kabul Tarihi / Accepted: 28.01.2022 *DOI:10.21205/deufmd.2022247129*
Atıf şekli/ How to cite: KARAGÜL, K., GÜNDÜZ, G. (2022). Gezgin Satici Problemi Için Yeni Bir Sezgisel:maxS. DEUFMD, 24(71), 665-677.

Abstract

In this study, a new initial solution heuristic was proposed for the traveling salesman problem. The proposed maxS method is based on a new distance matrix obtained by normalizing the distance matrix of the problem being addressed according to the maximum row value. The proposed method was tested on 20 small and 11 large-scale problems, recommended by Hougardy and Zhong, which are difficult to solve optimally. The same problems were also solved by Greedy, Boruvka, Quick-Boruvka, Nearest-Neighborhood and Lin-Kernighan heuristics working on the Concorde software. Based on the comparisons, it is seen that the recommended maxS heuristic performance was better than that of Greedy and Nearest-Neighborhood heuristics and it showed a similar performance with Boruvka in small-scale problems. When the same comparisons were made for large-scale problems, maxS showed better performance than Quick Boruvka and Nearest-Neighborhood heuristics, on average. The maxS heuristic, which is very effective in terms of solution times, can be proposed as a promising initial solution method.

Keywords: *Traveling Salesman Problem, maxS, Boruvka, Nearest-Neighborhood, Lin-Kernighan, Initial Solutions*

Öz

Bu çalışmada, gezgin satıcı problemi için yeni bir başlangıç çözüm sezgiseli önerilmiştir. Önerilen maxS metodu, üzerinde çalışılan problemin mesafe matrisinin maksimum satır değerine göre normalize edilmesiyle elde edilen yeni mesafe matrisi ile çalışır. Önerilen metot, Hougardy ve Zhong tarafından tavsiye edilen ve optimal çözümü zor olan 20 küçük ve 11 büyük ölçekte problem üzerinde test edilmiştir. Aynı problemler, Concorde yazılımı üzerinde çalışan Greedy, Boruvka, Quick-Boruvka, Nearest-Neighborhood and Lin-Kernighan sezgiselleri ile de çözülmüştür. Çözümler karşılaştırıldığında küçük ölçekli problemler için maxS sezgiselinin performansının Greedy ve Nearest-Neighborhood sezgisellerinden daha iyi olduğu ve Boruvka ile benzer performansta olduğu gözlenmiştir. Benzer karşılaştırmalar büyük ölçekli problemler için yapıldığında maxS, Quick Boruvka ve Nearest-Neighborhood sezgisellerinden ortalama olarak daha iyi performans göstermiştir. Çözüm zamanları açısından çok etkili olan maxS sezgiseli, gelecek vaadeden başlangıç çözüm yöntemi olarak önerilebilir.

Anahtar Kelimeler: *Gezgin Satıcı Problemi, maxS, Boruvka, Nearest-Neighbourhood, Lin-Kernighan, başlangıç çözümü*

1. Introduction

Thousands of years ago, the famous irony of Socrates expressed that knowledge is an immense ocean: "The only true wisdom is in knowing you know nothing." The Traveling Salesman Problem (TSP) can be expressed as the shortest possible travel plan starting from a salesman's starting position and providing all customer locations in the sales area only once and returning to the initial position in case that both all the locations to be traveled and the distances between the pairs of customer locations are known. This definition can tell someone who is not involved in the optimization field: What a simple problem! Indeed, explaining and defining TSP is a simple problem. But the solution is hard enough to remind the famous irony of Socrates, and it has a very important place in the scientific literature. In this context, there is a constant challenge in this area, and efforts to develop better solution approaches are ongoing.

It would be appropriate to start by explaining some concepts from graph theory for TSP. A G graph is a sequential pair of G=(V,E) where V is a finite set and E is a set of two-point subsets of V. The elements of the V set are vertices, the elements of the cluster E are called edges of the G [1]. An example diagram is given in Figure 1 [2]. Walk, path, circuit, Hamiltonian path and TSP will be defined on this graph.

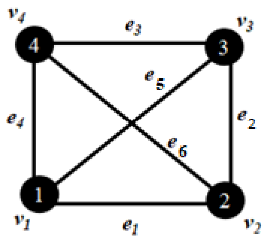

Figure 1: A sample graph [2]

Definition1. In a G graph, $v_1, e_1, v_2, e_2, v_3, \dots, e_k, v_k$ as a list of vertices and edges are defined a walk, and here e_i edge is the one that combines v_i and v_{i+1} vertices. In this case $v_1, e_1, v_2, e_2, v_3, e_3, v_4, e_6, v_{2,}, e_2, v_3,$ list in Figure 1 is a walk. A walk is considered to be closed if the starting vertex is the same as the ending vertex, that is v0=vk. A walk is considered open otherwise.

Definition 2: A Trail is defined as a walk with no repeated edges. In Figure 1, $v_1, e_1, v_2, e_2, v_3, e_3, v_4, e_4, v_{1,}, e_5, v_3$ list is a trail.

Definition 3: A Path is defined as an open trail with no repeated vertices. In Figure 1, $v_1, e_1, v_2, e_2, v_3, e_3, v_4$ list is a path.

Definition 4: A Cycle is defined as a closed trail where no other vertices are repeated apart from the start/end vertex. In Figure 1, $v_1, e_1, v_2, e_2, v_3, e_5, v_1$ list is a cycle.

Definition 5. Hamiltonian Cycle is a cycle that visits each node of the graph exactly once. In Figure 1, $v_1, e_1, v_2, e_2, v_3, e_3, v_{4,}, , e_4, v_1$ list is a cycle.

Calculating a tour in a graph with the minimum total weight values that can be found as a Hamiltonian cycle is called a Traveling Salesman Problem (TSP). For TSP, the weight values of the edges are retained as the distance matrix. TSP can be expressed in two ways as symmetric and asymmetric according to the distance matrix. For a TSP, if n is assumed to be the number of towns in the salesman's region, it is expressed by an n-node graph. The distance between these n nodes is expressed as $D = [c_{ij}], nxn$ distance matrix. In the distance matrix, if $c_{ij} = c_{ji}$ and $c_{ij} = 0, \forall i = j$, it is defined as symmetric TSP. If $c_{ij} \neq c_{ji}, \exists i = j$ and $c_{ij} = 0, \forall i = j$ then it is defined as an asymmetric TSP.

Flood, a well-known researcher in the field of TSP, worked on school bus routing in 1937 to find optimal solutions. In the mid-1950s the TSP became one of the most up-to-date and challenging issues. One of the first references to the term TSP was given in 1949 by Robinson in his report entitled "Hamilton Game (Traveling Salesman Problem)". This report written by Robinson is a TSP solution report prepared due to a challenge for the RAND Corporation. The Hamiltonian cycle term was used for the memory of him. Hamilton is known for his work on the dodecahedron which shows that anyone can return to the starting point by moving over the distances, regardless of the point where he/she has started. In 1972, Karp showed that the Hamiltonian problem was NP-complete. TSP is a problem in the NP-difficult class [3,4,5]. Along with the improvements in computer software and hardware, 24978 vertices TSP solution was reached in 2004, 50 years after the 49-node GSP solution of Dantzig, Fulkerson and Johnson. Table 1 shows solution milestones for TSP instances [6].

Table 1. *TSP Milestones* [6]

Year	Researchers	Problem Size	Problem Name
1954	G. Dantzig, R. Fulkerson, and S. Johnson	49	dantzig42
1971	M. Held and R.M. Karp	64	64 random points
1975	P.M. Camerini, L. Fratta, and F. Maffioli	67	67 random points
1977	M. Grötschel	120	gr120
1980	H. Crowder and M.W. Padberg	318	lin318
1987	M. Padberg and G. Rinaldi	532	att532
1987	M. Grötschel and O. Holland	666	gr666
1987	M. Padberg and G. Rinaldi	2392	pr2392
1994	D. Applegate, R. Bixby, V. Chvátal, and W. Cook	7397	pla7397
1998	D. Applegate, R. Bixby, V. Chvátal, and W. Cook	13509	usa13509
2001	D. Applegate, R. Bixby, V. Chvátal, and W. Cook	15112	d15112
2004	D. Applegate, R. Bixby, V. Chvátal, W. Cook,	24978	sw24798
2004	K. Helsgaun	24978	sw24798

Different solution algorithms are available for TSP. A classification of solution approaches can be made in the form of constructive algorithms, tour improvement algorithms and hybrid algorithms.

Constructive algorithms usually continue to visit the nodes by completing one of the nodes to be visited in each iteration until the tour is completed and finds a suitable solution. The nearest-neighborhood algorithm can be given as an example. The tour improvement algorithms consider a given initial solution and investigate whether there is a more least costly tour with changes to nodes and/or edges. If a possible low-cost tour is available, the tour will be improved. An example of tour improvement algorithms is the 2-Opt algorithm. Hybrid algorithms are used to obtain the initial solution using any tour constructive algorithms and improve this initial solution with a metaheuristic algorithm [7,8]. In this study, the literature search will be concentrated at this point on constructive algorithms, as the algorithm is proposed to produce a constructive initial solution for TSP.

Srour et al. [9] proposed an approach for TSP solution called the Water Flow-Like Algorithm. In this study, the initial solutions were constructed with the nearest-neighborhood algorithm and water flow algorithm and ant colony system (ACS) solutions were compared. In another study, Brute Force, Greedy, Nearest-Neighborhood, 2-Opt, Branch-Bound, Genetic Algorithm, Simulated Annealing and Artificial Neural Networks were used for TSP solutions and in terms of solution quality and solution times on the test bed. [10]. In another study, the initial solutions for a Water Flow-Like and Tabu Search hybrid method are constructed randomly [11]. Kamarudin et al. [12] proposed two different initial solutions for TSP: The Simulated Annealing and Nearest-Neighborhood algorithms and analyzed the performance of the Water Flow-Like algorithm and suggested that they achieved better performance with initial solutions constructed by Simulated Annealing. Wu et al. [13] used the Enhanced Water Flow-Like Algorithm for scheduling and sequencing of identical machines and constructed the initial solutions in a random format. Demiriz [14] proposed a solution based on the rank technique for TSP and solutions comparisons have been made using Concorde software.

Some of the researchers in the field of combinatorial optimization think that initial solutions are not useful, while others suggest that initial solutions are useful. Lin and Kernighan [15] proposed a very effective TSP solution algorithm and this algorithm is referred to as the Lin-Kernighan algorithm. The Lin and Kernighan algorithm randomly produces an initial solution as the first step and then tries to improve it. Later on, the Lin-Kernighan algorithm was improved by Helsgaun [8] and is now known as Lin-Kernighan-Helsgaun (LKH). It is one of the most effective TSP solution algorithms. It has been proposed by Helsgaun about the Lin-Kernighan algorithm and its initial solutions: The Lin-Kernighan algorithm repeatedly applies edge changes to different initial solutions for the same problem. The original Lin-Kernighan algorithm selects the initial tours randomly. Lin-Kernighan argues that the time spent on initial solutions is vaste of energy. They produce only constructive solutions that’s why there is only one initial solution. Furthermore, Helsgaun claims that the problem of dealing with initial solutions is not an easy-to-answer question. On the other hand, LKH code uses different initial solution algorithms. These algorithms are Boruvka, Greedy, Nearest-Neighborhood, Quick-Boruvka, Sierpinski, Random Walk. The same algorithms are also included in the Concorde software, the world's fastest exact solver [6]. Karagül has proposed new solution approaches for TSP, based on Transportation Problem solution [16], based on Hungarian solution [17], Prüfer based solution [18], 2-opt local search algorithm based solution[22] and hybrid fluid genetic algortihm based solution[23]. Sahin et. al proposed metaheuristics approaches for TSP on a spherical surface[24]. Aydemir at al proposed an algorithm for generating initial solutions for capacitated vehicle routing problem[25].

In this study, an algorithm that produces initial solutions using a constructive solution approach for TSP is proposed. In the second section, the proposed algorithm is given and explained on a small sample graph. In the third section, the performance of the proposed algorithm is compared with various initial solution algorithms from the literature and the results are analyzed. In the last section, conclusions and discussions for further studies are given.

2. Material and Method

2.1. Explanation of maxS algorithm on Small TSP

Explaining newly developed techniques through small sample problems facilitates both understanding and analysis. Therefore, the maxS method will be explained through the TSP example used in Demiriz [14]. The small instance problem data and the solution steps have been demonstrated step by step in Table 2.

Step 1: Table 2(a) shows the distance matrix for the problem. The problem corresponds to the symmetric TSP problem and it has seven vertices.

Step 2: Before moving to Table 2(b), the maxS column appears. This column represents the maximum value in each row. The maxS matrix is obtained by dividing each line of the distance matrix by the elements in the maxS column.

Step 3: Table 2(b) is the solution matrix of the proposed method. Using this solution matrix, the steps of the algorithm are completed and the TSP initial solution is obtained.

Step 4: As in Table 2(c), the first row is used and the element with the smallest on this row is found. The smallest element in this row is 0, which corresponds to the first column. Therefore, the first node of the TSP solution becomes 1. This column is then closed with 1 values. Then as the selected node is 1, the algorithm goes to the related row 1.

Step 5: In Table 2(d), the element with the smallest value in row 1 is 0.32 which corresponds to column 7. In this case, node 7 is added as the second node of the TSP solution and column 7 is closed with 1 value.

Step 6: In Table 2(e), the algorithm goes to row 7 where the element with the smallest value is 0.44 which corresponds to column 4. Thus, the next node of the TSP solution is added as 4 and column 4 is closed with 1 value.

Step 7: In Table 2(f), the algorithm is positioned on row 4 in the maxS matrix where the smallest element is 0.32. This cell points to column 3 and thus node 3 is added to the TSP solution. And then column 3 is closed by 1 value.

Step 8: In Table 2(g), the algorithm goes to row 3 in the maxS matrix where the smallest element is 0.71, which indicates column 6. Thus, node 6 is added to the TSP solution and column 6 is closed with 1 value.

Step 9: In Table 2(h), the algorithm moves to row 6 in the maxS matrix and the smallest element indicates column 2 with 0.29. Therefore, node 2 is added to the TSP solution and column 2 is closed with 1 value.

Step 10: In Table 2(i), the algorithm goes to row 2 in the maxS matrix where the smallest element is 0.61. This cell points to column 5. Therefore, node 5 is added to the TSP solution and column 5 is closed with 1 value.

Step 11: In Table 2(j), as all of the maxS matrices are covered with 1 value, there is no node left to be added to another TSP solution. This terminates the algorithm.

Table 2. *Proposed Algortihm: maxS Solution Steps*

(a)

Distance Matrix

	1	2	3	4	5	6	7	maxS
1	0	786	549	657	331	559	250	**786**
2	786	0	668	979	593	224	905	**979**
3	549	668	0	316	607	472	467	**668**
4	657	979	316	0	890	769	400	**979**
5	331	593	607	890	0	386	559	**890**
6	559	224	472	769	386	0	681	**769**
7	250	905	467	400	559	681	0	**905**

(b)

maxS Matrix

	1	2	3	4	5	6	7
1	0.00	1.00	0.70	0.84	0.42	0.71	0.32
2	0.80	0.00	0.68	1.00	0.61	0.23	0.92
3	0.82	1.00	0.00	0.47	0.91	0.71	0.70
4	0.67	1.00	0.32	0.00	0.91	0.79	0.41
5	0.37	0.67	0.68	1.00	0.00	0.43	0.63
6	0.73	0.29	0.61	1.00	0.50	0.00	0.89
7	0.28	1.00	0.52	0.44	0.62	0.75	0.00

(c)

	1	2	3	4	5	6	7
1	0.00	1.00	0.70	0.84	0.42	0.71	0.32
2	0.80	0.00	0.68	1.00	0.61	0.23	0.92
3	0.82	1.00	0.00	0.47	0.91	0.71	0.70
4	0.67	1.00	0.32	0.00	0.91	0.79	0.41
5	0.37	0.67	0.68	1.00	0.00	0.43	0.63
6	0.73	0.29	0.61	1.00	0.50	0.00	0.89
7	0.28	1.00	0.52	0.44	0.62	0.75	0.00
TSP	1						

(d)

	1	2	3	4	5	6	7
1	1	1.00	0.70	0.84	0.42	0.71	0.32
2	1	0.00	0.68	1.00	0.61	0.23	0.92
3	1	1.00	0.00	0.47	0.91	0.71	0.70
4	1	1.00	0.32	0.00	0.91	0.79	0.41
5	1	0.67	0.68	1.00	0.00	0.43	0.63
6	1	0.29	0.61	1.00	0.50	0.00	0.89
7	1	1.00	0.52	0.44	0.62	0.75	0.00
TSP	1	7					

(e)

	1	2	3	4	5	6	7

(f)

	1	2	3	4	5	6	7

1	1	1.00	0.70	0.84	0.42	0.71	1
2	1	0.00	0.68	1.00	0.61	0.23	1
3	1	1.00	0.00	0.47	0.91	0.71	1
4	1	1.00	0.32	0.00	0.91	0.79	1
5	1	0.67	0.68	1.00	0.00	0.43	1
6	1	0.29	0.61	1.00	0.50	0.00	1
7	1	1.00	0.52	0.44	0.62	0.75	1
TSP	1	7	4				

(g)

1	1	1.00	0.70	1	0.42	0.71	1
2	1	0.00	0.68	1	0.61	0.23	1
3	1	1.00	0.00	1	0.91	0.71	1
4	1	1.00	0.32	1	0.91	0.79	1
5	1	0.67	0.68	1	0.00	0.43	1
6	1	0.29	0.61	1	0.50	0.00	1
7	1	1.00	0.52	1	0.62	0.75	1
TSP	1	7	4	3			

(h)

	1	2	3	4	5	6	7
1	1	1.00	1	1	0.42	0.71	1
2	1	0.00	1	1	0.61	0.23	1
3	1	1.00	1	1	0.91	0.71	1
4	1	1.00	1	1	0.91	0.79	1
5	1	0.67	1	1	0.00	0.43	1
6	1	0.29	1	1	0.50	0.00	1
7	1	1.00	1	1	0.62	0.75	1
TSP	1	7	4	3	6		

(i)

	1	2	3	4	5	6	7
1	1	1.00	1	1	0.42	1	1
2	1	0.00	1	1	0.61	1	1
3	1	1.00	1	1	0.91	1	1
4	1	1.00	1	1	0.91	1	1
5	1	0.67	1	1	0.00	1	1
6	1	0.29	1	1	0.50	1	1
7	1	1.00	1	1	0.62	1	1
TSP	1	7	4	3	6	2	

(j)

	1	2	3	4	5	6	7
1	1	1	1	1	0.42	1	1
2	1	1	1	1	0.61	1	1
3	1	1	1	1	0.91	1	1
4	1	1	1	1	0.91	1	1
5	1	1	1	1	0.00	1	1
6	1	1	1	1	0.50	1	1
7	1	1	1	1	0.62	1	1
TSP	1	7	4	3	6	2	5

	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1
TSP	1	7	4	3	6	2	5

Cost = 2585 km / Optimal=2575 km

The maxS solution for this problem was found to be 2585 km and the related route is [1-7-4-3-6-2-5]. The optimal solution value for the problem is given as 2575 km. In this case, the maxS solution approach was able to approach the optimal solution with a 0.388% gap value.

2.2. Proposed Algorithm maxS Matlab/ Octave Code

In this subsection, Matlab / Octave code for the proposed solution approach is given as in Figure 2, to guide the reader. Octave is an alternative open source application to the Matlab scientific computing language. The code in Figure 2 is designed to be easy to read and easy to use in scientific studies. Using the code of the proposed algorithm maxS, performance analysis will be explained in the next section.

```
% maxS.m algorithm Matlab/Octave Code for TSP
xy=Read(berlin52.tsp);   % Read the TSP data file and get the xy coordinates.
D=Distance(xy);          % Calculate the distance matrix and assign to matrix D.
[m,n]=size(D);           % Get the size information.
xD=D;                    % Prepare the temporary distance matrix xD.
maxS=zeros(1,m);         % Create an empty maxS vector.
for i=1:m
   maxS(i)=max(xD(i,:));   % Get each row's max value and assign to the maxS vector.
end
M=zeros(m,m);            %Create an empty maxS matrix.
for i=1:m
   M(i,:)=xD(i,:)./maxS(i); % Calculate the elements of maxS matrix.
end
A=M; % Assign the maxS matrix to matrix A and use matrix A for routing.
%--------Creating TSP tour------------------------
rotaMx=zeros(1,m);  % Create an empty route vector.
t=1; ss=1;
while t<=m
   [~,bx]=min(A(ss,:));
   rotaMx(t)=bx;
   A(:,bx)=1;
   ss=bx;
   t=t+1;
end
rotaCost=CostTSP(rotaMx,D);   % Calculate TSP cost and assign cost to rotaCost.
```

Figure 2. maxS Algorithm Code for Matlab/Octave

3. Computational Analysis for maxS Algorithm

For the analysis and comparison of the proposed method, a TSP test bed was chosen and the algorithms in version 1.1 of the Concorde software was used for comparisons. Concorde software can produce solutions for Greedy, Boruvka, Quick-Boruvka (QBoruvka), Nearest-Neighborhood (NN), Lin-Kernighan (L-K) algorithms. The proposed algorithm maxS was coded in Matlab environment. The solutions of the maxS heuristic were obtained using Matlab version 2016b, 2.40 GHz Intel Dual Core, 8 MB memory and single kernel on Linux operating system. For the analysis of the heuristics on the Concorde software, the Windows operating system, Intel Core (TM) i7-4800MQ CPU 2.70GHz, 16 MB RAM was used with only a single core.

In their study conducted on the test bed instances by Hougardy and Zhong [19], detailed explanations about the problems were given. They have explained how difficult it is to solve these new problem types optimally, and at the same time they analyzed solutions from 52 to

199 nodes with Concorde software, the world's fastest exact solver. As seen in Table 3, Concorde's solution times were far beyond the acceptable limits. However, solutions were produced and reported by Helsgaun using LKH for all of these problems [20]. The test problems produced by Hougardy and Zhong can be found on the University website of Hougardy [21]. In our study, the first 20 test bed problems produced by Hougardy and Zhong were selected for the analysis.

Table 3. Exact and heuristic solutions of Concorde and maxS solutions

P.No	P. Name	A (s)	B (s)	Optimal	maxS	Greedy	Boruvka	Qboruvka	NN	L-K
1	Tnm52	12	0,004006	551609	**616205**	621663	635322	610775	663064	552619
2	Tnm55	17	0,001212	605778	**676292**	679763	696292	679406	731891	606838
3	Tnm58	21	0,000935	660687	**734525**	743899	755968	732429	735026	661279
4	Tnm61	30	0,001033	716131	**795235**	811773	871037	792790	812696	717865
5	Tnm64	33	0,000831	770162	**831125**	862167	851075	855268	882447	772302
6	Tnm67	47	0,000908	825328	**918639**	945407	929682	921815	915631	825328
7	Tnm70	68	0,000771	881036	**989672**	1001787	988407	980248	984974	881440
8	Tnm73	84	0,000781	893843	**1043838**	1066162	1075388	1041748	1063838	938396
9	Tnm76	103	0,000955	949961	**1069196**	1141042	1116756	1117109	1093550	992771
10	Tnm79	152	0,000790	1006535	**1170144**	1195862	1149375	1138971	1137707	1048105
11	Tnm82	190	0,000862	1062686	**1252852**	1214567	1250845	1203570	1203439	1107116
12	Tnm85	164	0,000900	1117381	**1314011**	1315646	1216265	1275674	1339318	1156776
13	Tnm88	196	0,000944	1172734	**1296026**	1316025	1277425	1291828	1277426	1174331
14	Tnm91	275	0,001672	1228726	**1318062**	1396595	1338027	1353120	1326122	1229432
15	Tnm94	397	0,001604	1285416	**1425383**	1396066	1399991	1396675	1396066	1285626
16	Tnm97	566	0,001022	1342086	**1503342**	1481644	1466578	1443332	1474709	1342567
17	Tnm100	664	0,001247	1398070	**1574837**	1565822	1507563	1513639	1544845	1399036
18	Tnm103	478	0,001465	1412229	**1584255**	1589900	1557687	1560003	1602903	1455346
19	Tnm106	761	0,001446	1469617	**1717744**	1659819	1628262	1654474	1688920	1513698
20	Tnm109	1068	0,001692	1527709	**1780021**	1699171	1709318	1667927	1760381	1569687
Averages				**1043886**	**1180570**	**1185239**	**1171063**	**1161540**	**1181748**	**1061528**
A : Concorde run time (s) / B: maxS run time (s) / Optimal: Concorde optimal solutions / s:seconds										

In Table 3, the numbers next to each problem name refer to the number of nodes in the related problem. The times shown in column A are Concorde's solution times and for instance 1069 seconds were spent for a 109 node TSP. The solution times for the maxS approach are given, but there are no solution times for heuristic methods on the Concorde software interface. Therefore, no comparisons will be made for the time durations. Only the simulated times for maxS are added as a reference for further studies. Since the L-K approach in these heuristic solutions uses these solutions by using an initial solution, the L-K algorithm is not only used for comparisons but is intended as a reference for future studies. In order to make a better comparision between the maxS approach and the approaches that produce different starting solution, the gap% values that indicate the deviations from the optimal are given in Table 4.

Table 4. Gaps % of the Concorde heuristics and maxS solutions from optimal

		Gap %					
P.No	**P. Adı**	**maxS**	**Greedy**	**Boruvka**	**Qboruvka**	**NN**	**L-K**
1	Tnm52	11.71	12.70	15.18	10.73	20.21	0.18
2	Tnm55	11.64	12.21	14.94	12.15	20.82	0.17
3	Tnm58	11.18	12.59	14.42	10.86	11.25	0.09
4	Tnm61	11.05	13.36	21.63	10.70	13.48	0.24
5	Tnm64	7.92	11.95	10.51	11.05	14.58	0.28
6	Tnm67	11.31	14.55	12.64	11.69	10.94	0.00
7	Tnm70	12.33	13.71	12.19	11.26	11.80	0.05
8	Tnm73	16.78	19.28	20.31	16.55	19.02	4.98
9	Tnm76	12.55	20.11	17.56	17.60	15.12	4.51
10	Tnm79	16.25	18.81	14.19	13.16	13.03	4.13
11	Tnm82	17.89	14.29	17.71	13.26	13.25	4.18
12	Tnm85	17.60	17.74	8.85	14.17	19.86	3.53
13	Tnm88	10.51	12.22	8.93	10.16	8.93	0.14
14	Tnm91	7.27	13.66	8.90	10.12	7.93	0.06
15	Tnm94	10.89	8.61	8.91	8.66	8.61	0.02
16	Tnm97	12.02	10.40	9.28	7.54	9.88	0.04
17	Tnm100	12.64	12.00	7.83	8.27	10.50	0.07
18	Tnm103	12.18	12.58	10.30	10.46	13.50	3.05
19	Tnm106	16.88	12.94	10.79	12.58	14.92	3.00
20	Tnm109	16.52	11.22	11.89	9.18	15.23	2.75
Averages		**12.86**	**13.75**	**12.85**	**11.51**	**13.64**	**1.57**

When Table 4 and the heuristic approach compared to the first 20 problems selected from the Euclidean GSP test bed of Hougardy and Zhong that are difficult to solve, are evaluated, it is possible to sort the algorithms QBoruvka, Boruvka and maxS at the first row as scoreless and then NN as the second one and Greedy as the third one according to the average solution gaps. These comparisons are also clearly visible on the graph given in Figure 3. The algorithms shown by the signs A, B, C, D, E in Figure 3 are maxS, Greedy, Boruvka, Qboruvka, NN, respectively. As can be seen from this comparison chart, it can be said that maxS shows a competitive deviation from the optimal on average.

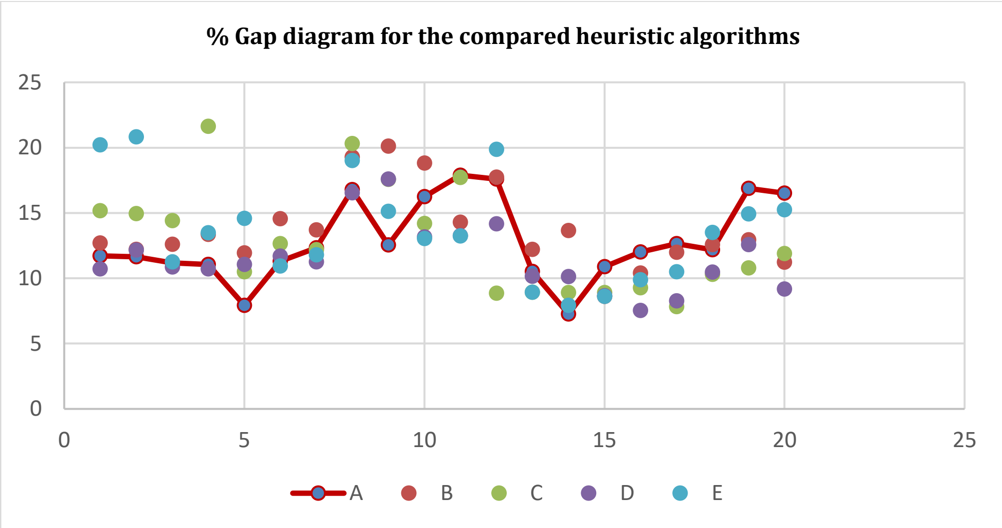

Figure 3: maxS Algorithm and Other Heuristics Deviations from the Optimal

Table 5. Heuristics and maxS solutions for Large-Scale Instances

P.No	P. Name	H	maxS (sec)	BKS	maxS	Greedy	Boruvka	Qboruvka	NN	L-K
1	Tnm502	*	0.01	8749995	9106673	9030246	9006411	8978114	9362888	8755518
2	Tnm1000	*	0.08	18137296	18553989	18454589	18438723	18426701	19618679	18145598
3	Tnm2002	*	0.22	37029600	37475105	37370253	37288700	38108787	37698300	37046387
4	Tnm3001	*	0.50	55939349	56399706	56373914	56197326	56623001	59962938	55948513
5	Tnm4000	*	0.86	74858233	75252693	75236866	75226869	75254384	76282814	74863285
6	Tnm5002	*	1.32	93784081	94254080	94154563	94084686	94487854	97922507	93790079
7	Tnm6001	*	1.96	112708118	113181440	113071223	112989346	113787111	113797025	112712247
8	Tnm7000	*	2.58	131633371	132120880	132008338	131992314	132189996	135825853	131642878

9	**Tnm8002**	*	3.42	150561446	151037518	151018483	150902514	150878943	152023700	150561486
10	**Tnm9001**	*	4.86	169487546	169889923	169864054	169804045	170203775	175317636	169492338
11	**Tnm10000**	*	7.22	188414262	188893121	188781763	188716742	188683999	196678861	188415184
Averages			**2.09**	**94663936**	**95105920**	**95033117**	**94967970**	**95238424**	**97681018**	**94670319**
* : Keld Helsgaun Solutions / BKS: Best Known Solutions calculated by Keld Helsgaun [20].										

The most large-scale problem that can be solved with Concorde is the 199-node Tnm example. Therefore, for large-scale test problems, 11 large-scale problems produced by Hougardy and Zhong are selected. These problems do not seem to be solvable by the Concorde software in today's conditions. In Table 5, with the BKS column, the solutions obtained by Helsgaun with LKH code are given. At the same time, the solution values of the maxS method in seconds are given for reference in future studies. In Table 6, the percentage gap values of solved heuristics from BKS are given both for maxS and for the algorithms of Concorde software. The Greedy and Boruvka algorithms were found to have a better mean deviation than maxS in the case of large-scale problems. On the other hand, the Qboruvka and NN methods are behind the maxS performance.

Table 6. Heuristics and maxS gap (%) values for large-scale TSPs

		Gap %					
P.No	**P. Name**	**maxS**	**Greedy**	**Boruvka**	**QBoruvka**	**NN**	**L-K**
1	Tnm502	4.08	3.20	2.93	2.61	7.00	0.06312
2	Tnm1000	2.30	1.75	1.66	1.60	8.17	0.04577
3	Tnm2002	1.20	0.92	0.70	2.91	1.81	0.04533
4	Tnm3001	0.82	0.78	0.46	1.22	7.19	0.01638
5	Tnm4000	0.53	0.51	0.49	0.53	1.90	0.00675
6	Tnm5002	0.50	0.40	0.32	0.75	4.41	0.00640
7	Tnm6001	0.42	0.32	0.25	0.96	0.97	0.00366
8	Tnm7000	0.37	0.28	0.27	0.42	3.18	0.00722
9	Tnm8002	0.32	0.30	0.23	0.21	0.97	0.00003
10	Tnm9001	0.24	0.22	0.19	0.42	3.44	0.00283
11	Tnm10000	0.25	0.20	0.16	0.14	4.39	0.00049
Averages		**1.00**	**0.81**	**0.70**	**1.07**	**3.95**	**0.01800**

4. Conclusions and Discussion

In this study, maxS was proposed as a new initial solution method for TSP. Solutions and the solution times were obtained on 20 small size and 11 large size problems that were chosen from the problem group defined by Hougardy and Zhong as difficult to find the optimal solution problems. The size of the small problems ranges from 52 nodes to 109 nodes. The size of the large problems ranges from 502 nodes to 10000 nodes. The same problems were also solved with the Greedy, Boruvka, Qboruvka, NN and L-K heuristics provided by the Concorde software and the results were recorded.

The average deviations of maxS, Greedy, Boruvka, QBoruvka, NN and L-K heuristic algorithms for small problems were calculated as 12.86, 13.75, 12.85, 11.51, 13.64, 1.57. The average deviations of maxS, Greedy, Boruvka, QBoruvka, NN and L-K heuristic algorithms for large-scale problems were found as 1.00, 0.81, 0.70, 1.07, 3.95, 0.018. The maxS algorithm shows an equal performance with the Boruvka algorithm while showing a better performance than the Greedy, and NN algorithms in small problems. For the large-scale problems, the maxS algorithm performed better than the Qboruvka and NN algorithms, but remained behind the Greedy and Boruvka algorithms. In the light of these analysis, maxS heuristics which is proposed as a new initial solution algorithm, shows a very competitive performance. Another case is the performance of the proposed maxS solution times. The average solution time for 20 small problems is 0.0012 seconds. The average solution time for 11 large-scale problems was recorded as 2.09 seconds.

The proposed new approach is important from two points of view. The first one is that it is competitive with the methods in the literature in terms of the test results. Therefore, some tour improvement methods and/or initial solutions for metaheuristics may be proposed as constructive heuristics. From another point of view, it can be proposed as a constructive solution approach to the application and solution of different problems that can be modeled as TSP because it produces fast and effective results.

References

[1] Matausek, J., Nesetril, J., 2008. An Invitation to Discrete Mathematics, (2nd edition). Oxford University Press, New York, USA.

[2] Papadimitriou, C. H., Steiglitz, K., 1998. Combinatorial Optimization: Algorithms and Complexity, Dover Publications, New York, USA.

[3] Cook, W.J., 2012. In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation, Princeton University Press, New Jersey, USA.

[4] Gass, S.I., Assad, A.A., 2005. An Annotated Timeline of Operations Research: An Informal History, Kluwer Academic Publishers, New York, USA.

[5] Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B., 1985. The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, John Wiley & Sons, New York, USA.

[6] Cook, W., "The Traveling Salesman Problem", http://www.math.uwaterloo.ca/tsp/ history/milestone.html, Accessed:20/03/2019.

[7] Kim, I.B., Shim, I.J., Zhang, M., 1998. Comparison of TSP algorithms, Project for Models in Facilities Planning and Materials Handling.

[8] Helsgaun, K., 2000. "An effective implementation of the Lin–Kernighan traveling salesman heuristic", European Journal of Operational Research, 126(1): 106–130.

[9] Srour, A., Othman, Z.A., Hamdan, A.R., 2014. "A water flow-like algorithm for the travelling salesman problem", Advances in Computer Engineering, Hindawi Publishing Corporation, 1-14.

[10] Abid, M.M., Muhammad, I., 2015. "Heuristic approaches to solve traveling salesman problem", TELKOMNIKA Indonesian Journal of Electrical Engineering, 15(2): 390-396.

[11] Bostamam, J.M., Othman, Z., 2016. "Hybrid water flow-like algorithm with Tabu search for traveling salesman problem", AIP Conference Proceedings 1761, 020058.

[12] Kamarudin, A.A., Othman, Z.A., Sarim, H.M., 2016. "Improvement initial solution water flow like algorithm using simulated annealing for travelling salesman problem", International Review of Management and Marketing, 6(8): 63-66.

[13] Wu, G-H., Cheng, C-Y., Yang, H-I., Chena, C-T., 2018. "An improved water flow-like algorithm for order acceptance and scheduling with identical parallel machines", Applied Soft Computing, 71, 1072-1084.

[14] Demiriz, A., "Solving Traveling Salesman Problem by Ranking", http://www.ayhandemiriz.com/SakaryaWebSite/papers/benelearn09.pdf, Accessed: 19/03/2019.

[15] Lin, S., Kernighan, B.W., 1973. "An Effective Heuristic Algorithm for the Traveling-Salesman Problem", Operations Research, 21(2): 498–516.

[16] Karagül, K. (2019). A Novel Solution Approach for Travelling Salesman Problem: TPORT. DEUFMD, 21(63), 819-832

[17] Karagül, K. (2019). A Novel Solution Approach for Travelling Salesman Problem Based on Hungarian Algorithm. Mühendislik Bilimleri ve Tasarım Dergisi , 7 (3) , 561-571 . DOI: 10.21923/jesd.523623

[18] Karagül, K . (2019). Prüfer-Karagül Algorithm: A Novel Solution Approach for Travelling Salesman

Problem. Mehmet Akif Ersoy Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi , 6 (2) , 452-470 . DOI: 10.30798/makuiibf.508842

[19] Hougardy, S., Zhong, X., 2018. "Hard to Solve Instances of the Euclidean Traveling Salesman Problem", arXiv:1808.02859 [cs.DM].

[20] Helsgaun, K., "Lin-Kernighan-Helsgaun", http://akira.ruc.dk/~keld/research/LKH/, Accessed:01/03/2019.

[21] Hougardy, S., Zhong, X., "Hard to Solve Instances of the Euclidean Traveling Salesman Problem", http://www.or.uni-bonn.de /~hougardy/HardTSPInstances.html, Accessed: 01/03/2019.

[22] Karagul, K., Aydemir, E. and Tokat, S., 2016. Using 2-Opt based evolution strategy for travelling salesman problem. An International Journal of Optimization and Control: Theories & Applications (IJOCTA), 6(2), pp.103-113.

[23] Şahin, Y. and Karagül, K.,2019. Solving Travelling Salesman Problem Using Hybrid Fluid Genetic Algorithm (HFGA). Pamukkale University Journal of Engineering Sciences, 25(1), pp.106-114.

[24] Sahin, Y., Aydemir, E., Karagul, K., Tokat, S. and Oran, B., 2021. Metaheuristics Approaches for the Travelling Salesman Problem on a Spherical Surface. In Interdisciplinary Perspectives on Operations Management and Service Evaluation (pp. 94-113). IGI Global.

[25] Aydemir, E., Karagül, K. And Tokat, S., Kapasite Kisitli Araç Rotalama Problemlerinde Başlangiç Rotalarinin Kurulmasi İçin Yeni Bir Algoritma. Mühendislik Bilimleri Ve Tasarım Dergisi, 4(3), Pp.215-226.