TWMS J. App. and Eng. Math. V.15, N.2, 2025, pp. 412-420

A CONVEX HULL BASED ALGORITHM FOR SOLVING THE TRAVELING SALESMAN PROBLEM

F. NURIYEVA[1,2*], H. KUTUCU [3], §

Abstract. This paper introduces a new algorithm for solving the Traveling Salesman Problem, based on the convex hull. The proposed algorithm has two versions, in the first version, starting from the vertices farthest from the center, unselected vertices are successively added to the current tour and in the second version, starting from the vertices closest to the center, unselected vertices are successively added to the current tour. The proposed algorithm has been implemented in the Python programming language, and computational experiments have been conducted on library problems of varying dimensions. The obtained results are compared with well-known algorithms, demonstrating the efficiency of the proposed algorithm. The results of these experiments demonstrate the effectiveness and efficiency of the proposed algorithm.

Keywords: traveling salesman problem, insertion heuristic, convex hull, nearest neighbor, greedy algorithm.

AMS Subject Classification: 05C85, 90C27

1. Introduction

The Traveling Salesman Problem (TSP) is a well known optimization problem. TSP is a combinatorial optimization problem that seeks to find the most efficient route for a salesman who needs to visit a set of cities exactly once and return to the starting city while minimizing the total distance or cost traveled [2, 3, 4, 9]. Despite its seemingly simple description, solving the TSP is a challenging task as it belongs to the class of NP-hard problems [4, 8, 9, 11]. Given n is the number of cities to be visited, the total number of possible routes covering all cities can be given as a set of feasible solutions of the TSP and is given as $\frac{(n-1)!}{2}$.

The TSP can be classified into three main categories: the Symmetric Traveling Salesman Problem (sTSP), the Asymmetric Traveling Salesman Problem (aTSP), and the Multi-Traveling Salesman Problem (mTSP) [15].

sTSP: In the Symmetric Traveling Salesman Problem (sTSP), consider a set of cities represented by $V = \{v_1, \ldots, v_n\}$, with the edge set $E = \{(i, j) : i, j \in V\}$, and let $c_{ij} = c_{ji}$

[1] Dokuz Eylul University, Faculty of Science, Department of Computer Science, Izmir, Türkiye.
e-mail: fidan.nuriyeva@deu.edu.tr; ORCID: https://orcid.org/0000-0001-5431-8506.

[2] Institute of Control Systems, The Ministry of Science and Education of the Republic of Azerbaijan, Baku, Azerbaijan.

[*] Corresponding author.

[3] Karabuk University, Faculty of Engineering, Department of Software Engineering, Karabuk, Türkiye
e-mail: hakankutucu@karabuk.edu.tr; ORCID: https://orcid.org/0000-0001-7144-7246.

§ Manuscript received: April 09, 2024; accepted: August 02, 2024.
TWMS Journal of Applied and Engineering Mathematics, Vol.15, No.2; © Işık University, Department of Mathematics, 2025; all rights reserved.

denote a cost of traveling associated with the edge $(i, j) \in E$. The objective of sTSP is to identify the shortest closed tour that visits each city exactly once. In this scenario, the cities, represented by $v_i \in V$, are defined by their coordinates (x_i, y_i), and if c_{ij} corresponds to the Euclidean distance between cities i and j, we refer to it as an Euclidean TSP [8].

aTSP: In the Asymmetric Traveling Salesman Problem (aTSP), the distinction lies in the fact that if the cost of traveling c_{ij} is not equal to c_{ji} for at least one pair (i, j), then the TSP transforms into an aTSP [2].

mTSP: The Multi-Traveling Salesman Problem (mTSP) is defined as follows: within a given set of vertices, there exist m salesmen stationed at a central depot vertex. The remaining vertices, designated as cities, serve as intermediate vertices to be visited. The objective of the mTSP is to determine tours for all m salesmen, each starting and ending at the depot. Every intermediate vertex must be visited exactly once, aiming to minimize the total cost associated with visiting all vertices. The cost metric can include factors such as distance, time, etc. Several variations of the mTSP exist, introducing complexities to the problem [3].

This problem is not only of theoretical interest but also has numerous practical applications in logistics, transportation, and network design, making it a fundamental topic in the field of optimization and computational mathematics [10].

In this paper, we proposed a new algorithm based on convex hull for solving symmetric TSP.

The rest of the paper is organized as follows: Section 2 outlines the definition and mathematical model of the Traveling Salesman Problem (TSP). In Section 3, a comprehensive introduction to well-known solution approaches for TSP is provided. Information about convex hull is presented in Section 4. The steps of the proposed algorithm are given in Section 5. A demonstration of the steps of the proposed algorithm are given in Section 6. Computational experiments conducted are detailed in Section 7, and finally, in Section 8, we conclude the paper with a summary.

2. Mathematical Formulation of TSP

The TSP can be effectively defined as a graph problem, a formulation that simplifies its representation and highlights its combinatorial nature. In the context of the TSP, a weighted graph is used to represent the cities and the distances between them. The graph is typically denoted as $G = (V, E)$, where:

V is the set of vertices representing the cities to be visited. Each vertex v_i in V corresponds to a city.

E is the set of edges connecting the cities, and each edge (v_i, v_j) in E is associated with a weight or cost c_{ij}, which represents the distance, time or cost of traveling from city i to city j.

The objective of the TSP as a graph problem is to find a Hamiltonian cycle, which is a closed path that visits each city exactly once and returns to the starting city. The goal is to determine the Hamiltonian cycle with the minimum total edge weight, which represents the shortest tour that allows the salesman to visit all cities and return to the starting city.

According to the definition of the TSP, its mathematical description is as follows [10, 15]:

Let x_{ij} be a binary variable, where $x_{ij} = 1$ if the salesman travels directly from city i to city j, and $x_{ij} = 0$ otherwise, N is a set of cities to be visited. Typically, $N = \{1, 2, 3, \ldots, n\}$, with n being the number of cities, c_{ij} is the cost or distance between city i and city j, where $i, j \in N$, c_{ij} is a non-negative value representing the cost, distance, or time to travel between these cities.

$$\min \sum_{i,j} c_{ij} x_{ij} \quad (1) \qquad (1)$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, 2, \ldots, n \quad (2) \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, 2, \ldots, n \quad (3) \tag{3}$$

$$u_i - u_j + n x_{ij} \leq n - 1; \quad \forall i, j = 1, 2, \ldots n; \quad i \neq j; \quad u_k \geq 0; \quad \forall k = 1, 2, \ldots n; \tag{4}$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 1, 2, \ldots, n, \quad i \neq j \tag{5}$$

Constraint (1) aims to minimize the total cost. Constraint (2) says that we arrive at city j from exactly one other city. Constraint (3) says that we leave city i to go to exactly one other city. Constraint (4) is subtour breaking constraint. Finally, constraint (5) imposes binary integrality on the variables.

3. Solving Approaches for TSP

Over the years, various solution approaches have been developed for solving this problem, each with its own strengths and weaknesses [10, 15]. Some of them are as follows:

3.1. **Exact Algorithms.**

Exact algorithms aim to find the optimal solution by exhaustively examining all possible permutations of city visits. While they guarantee the best solution, they can be computationally expensive for large instances [10, 16].

Some notable exact algorithms include:

Branch and Bound: This algorithm systematically prunes branches of the search tree to avoid exploring unpromising solutions [10].

Branch and Cut: An extension of branch and bound, this approach incorporates cutting planes to tighten the relaxation of the problem, reducing the search space [10].

3.2. **Heuristic Algorithms.**

Heuristic algorithms provide heuristic solutions quickly and are often used when optimality is not the primary concern. They are valuable for large-scale instances [1, 5, 12, 13].

Some popular heuristic methods include:

Nearest Neighbor: This simple algorithm starts at an arbitrary city and repeatedly selects the nearest unvisited city as the next destination until all cities are visited. It may not always yield the optimal solution, but it's quick and can be used as an initial solution for other algorithms [6, 7].

Greedy Algorithms: Greedy algorithms are a method of finding a feasible solution to the TSP. The algorithm creates a list of all edges in the graph and then orders them from smallest distance to largest distance. It then chooses the edges with the smallest distance first, providing they do not create a subtour [1, 8].

Insertion Heuristics: Insertion Algorithms start with a TSP subtour and insert one new vertex at each step until we have a valid Hamiltonian path. Then, like Nearest Neighbor algorithm, it connects the last vertex to the first to complete the TSP tour [5, 8].

3.3. **Genetic Algorithms.**

Taking inspiration from the mechanisms of natural selection, genetic algorithms iterate and evolve a population of potential solutions, aiming to discover improved routes for the Traveling Salesman Problem (TSP).

3.4. **Approximation Algorithms.** Approximation algorithms guarantee solutions within a certain factor of the optimal solution. They strike a balance between solution quality and efficiency [8].

Notable approximation algorithms include:

Minimum Spanning Tree (MST) Heuristic: This approach constructs an MST of the cities and then converts it into a tour. The resulting tour is at most twice as long as the optimal tour [8].

Christofides Algorithm: This algorithm, while computationally intensive, guarantees a solution within $\frac{3}{2}$ times the length of the optimal tour for TSP instances with Euclidean distances [8, 10].

3.5. **Metaheuristic Algorithms.** Metaheuristic algorithms are high-level strategies that guide the search for good solutions. They are often used to find high-quality solutions for large and complex TSP instances [10].

Common metaheuristics include:

Simulated Annealing: This method mimics the annealing process in metallurgy, gradually cooling the system to escape local optima and converges to a near-optimal solution.

Ant Colony Optimization (ACO): Inspired by the foraging behavior of ants, ACO algorithms simulate the path-finding process of ants to discover good solutions to the TSP.

Tabu Search: This local search algorithm uses memory to avoid revisiting previously explored solutions and explores the neighborhood of the current solution to find improvements.

Particle Swarm Optimization (PSO): PSO algorithms are inspired by the collective behavior of birds or fish. Particles explore the solution space and adjust their positions based on their own experience and that of their peers.

3.6. **Hybrid Approaches.** Hybrid approaches combine two or more solution methods to harness the strengths of each. For example, combining a genetic algorithm with local search can improve the quality of solutions [7, 10].

In practice, the choice of solution approach depends on factors like the size of the TSP instance, the required solution quality, and the computational resources available.

4. Convex Hull

The convex hull of a set of points is defined as the smallest polytope in such a way that the straight line between each pair of points lies inside the polytope. The convex hull problem appears as an important problem in many applications of computational geometry. It is used in many areas such as computer graphics, CAD/CAM applications, collision analysis, pattern recognition, image processing and machine learning.

Sharafutdinov et al. proposed a method to improve the generalization ability of machine learning models based on the convex hull overlap between multivariate datasets [17]. Zhou and Shi used a convex hull of training samples as the similarity measure for the face recognition problem [19]. Shesu et al. developed a method using alpha convex hulls to detect abnormal oceanic in situ temperature and salinity profiles [18]. Ostrouchov et al. showed that FastMap which is a dimension reduction technique picks all of its pivots from convex hull faces that provides a bridge to results in robust statistics [14].

There are several algorithms for computing 2D convex hulls of a given set of points such as Gift Wrapping, QuickHull and Graham's Scan with the time complexity of $O(nh)$, $O(n^2)$ and $O(nlogn)$, respectively, where n is the number of points and h is the number of points on the convex hull. In this paper, we implemented the QuickHull algorithm for computing the convex hull.

5. Proposed Algorithm

This section introduces the proposed algorithm, which relies on both convex hull and insertion heuristics. The proposed algorithm has two versions:

a) In the first version, starting from the farthest vertices from the center, the unselected vertices are successively added to the current tour.
b) In the second version, starting from the vertices closest to the center, the unselected vertices are successively added to the current tour.

Generally, we can describe the algorithm as follows:

Step 1. The Convex Hull (CH) of the set of vertex points (V) of the TSP graph is determined. $CH(V)$ creates the Initial Tour for TSP. Let's denote the set of vertex points that make up V_1 and $CH(V)$, and let $V_2 = V - V_1$. Let, $|V_1| = s$ and $p = n - s$.
Step 2. The center point of the set V_2 is determined. For this, $x = \frac{\sum_{i=1}^{p} x_i}{p}$, and $y = \frac{\sum_{i=1}^{p} y_i}{p}$ are calculated, and the point closest to the center (O) becomes the center point ($O(x, y)$).
Step 3. Starting from the farthest (nearest) vertices from the center, the vertices are successively added to the initial tour ($k = 1, 2, \ldots, p$).
Step 4. Continue with Step 3 until there are no more cities to be added (i.e., until $k > p$).
Step 5. Stop.

The INSERTION Procedure is performed as follows:

One of the vertices not yet included in the tour is selected as the farthest (nearest) vertex from the center, and it is denoted as "A_k". The determination of where this vertex will be added to the tour is based on the selection of an edge from the tour (let this edge be (A_l, A_m), meaning that the endpoints of this edge are (A_l) and (A_m)) according to the following criterion:

$$A_l A_k + A_k A_m - A_l A_m = min.$$

In other words, an edge (A_l, A_m) is selected from the tour such that the sum of the distances between its endpoints (A_l) and (A_m) to point A_k is subtracted from the length of edge (A_l, A_m), resulting in the smallest possible difference.

This edge (A_l, A_m) is removed from the tour, and in its place, edges $A_l A_k$ and $A_k A_m$ are added to the tour. Vertex A_k is removed from the set $V(k = k + 1)$.

6. Demonstration of the steps of the proposed algorithm

The steps of the algorithm are shown in the Figure 1 given below.

In Figure 1 (A), the set of vertices $V = \{v_1, v_2, \ldots, v_{14}\}$ is given. In Figure 1 (B), the Convex Hull of this set is determined, and the point O is selected as the center. This convex hull forms the initial tour.

In Figure 1 (C), among the vertices not included in the created tour, the furthest vertex from the center (v_8) is selected, and it is added to the previously formed tour. For this, the edge $v_1 - v_{11}$ is removed from the initial tour, and in its place, the edges $v_1 - v_8$ and $v_8 - v_{11}$ are added.

In Figure 1 (D), among the vertices not included in the created tour, the second furthest vertex from the center (v_{12}) is selected, and it is added to the previously formed tour. For this, the edge $v_{13} - v_{14}$ is removed from the initial tour, and in its place, the edges $v_{13} - v_{12}$ and $v_{12} - v_{14}$ are added.

This process continues, and the 4th, 5th, ..., 8th furthest points are successively added to the tour (Figures 1 (E) – (K)), and finally, the center vertex is added to complete the tour (Figure 1 (L)).

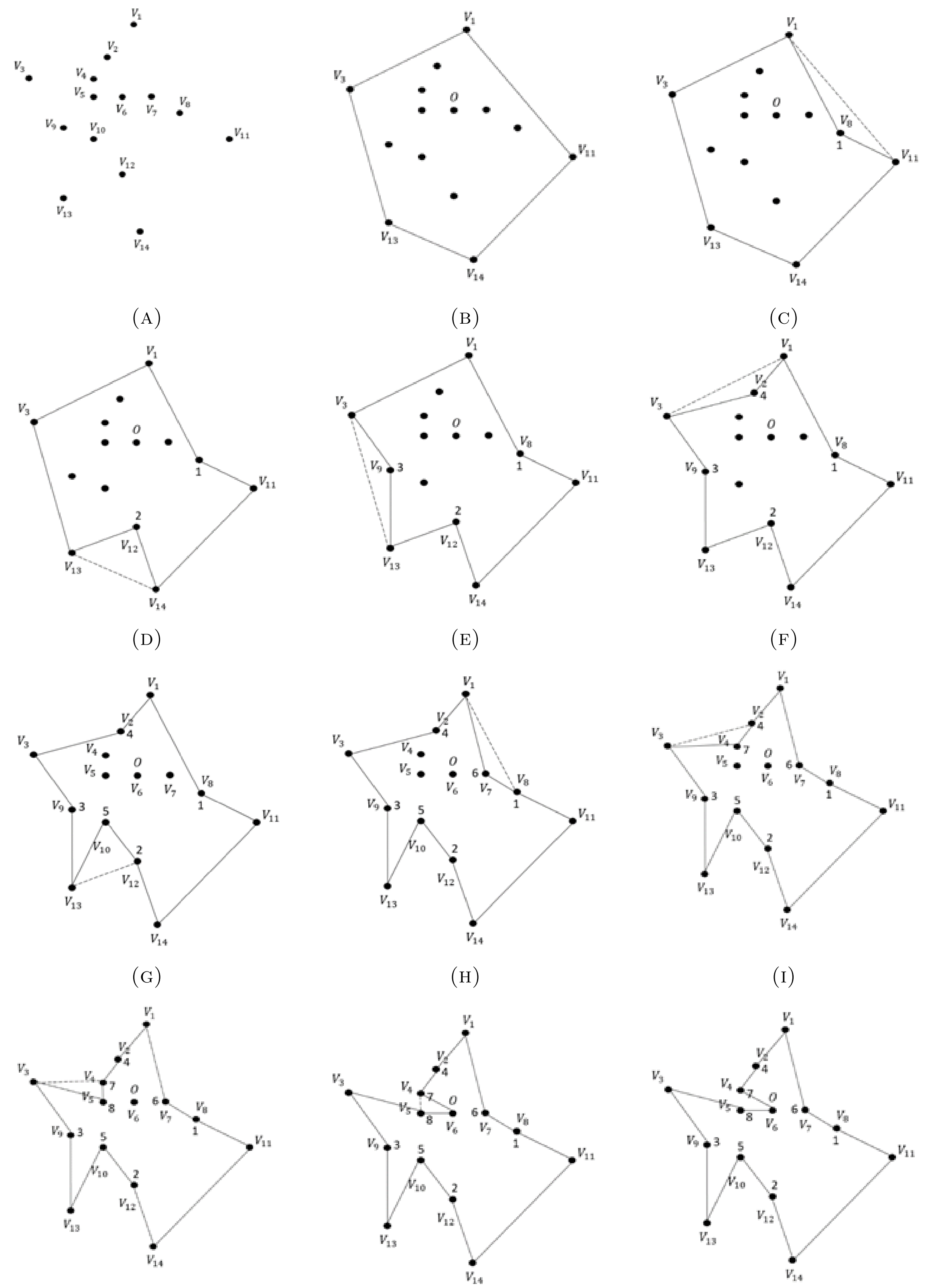

Figure 1. Demonstration of the steps of the proposed algorithm

Table 1. Computational Experiments

G	Optimal	NN Time(s)	Greedy Time(s)	Proposed Algorithm Version 1	Proposed Algorithm Version 2
ulysses16	74.108	78.127 0.000	88.923 0.010	74.23 0.003	74.59 0.003
ulysses22	75.665	86.906 0.000	89.436 0.010	76.15 0.004	76.36 0.004
eil51	426	505.774 0.016	481.518 0.125	487.923 0.02	459.065 0.02
berlin52	7542	8182.192 0.000	9954.062 0.281	8344.549 0.016	8037.864 0.016
eil76	538	612.656 0.016	617.131 0.672	608.649 0.041	607.909 0.041
rat99	1211	1369.535 0.016	1528.308 1.875	1395.813 0.67	1353.430 0.66
kroA100	21282	24698.497 0.016	24197.285 1.937	24589.593 0.070	21955.472 0.068
kroB100	22141	25882.973 0.016	25815.214 2.469	24170.217 0.068	24271.923 0.068
kroC100	20749	23566.403 0.015	25313.671 2.610	22512.937 0.068	22306.579 0.068
kroD100	21294	24855.799 0.016	24631.533 2.359	23105.397 0.068	23460.942 0.068
kroE100	22068	24907.022 0.016	24420.355 2.609	24562.072 0.68	23365.636 0.068
rd100	7910	9427.333 0.015	9900.458 2.648	9207.434 0.068	9167.441 0.068
pr124	6979	8447.973 0.015	8481.267 0.213	7648.189 0.067	7610.118 0.069
lin105	14379	16939.441 0.015	16479.785 3.187	15341.231 0.077	15888.284 0.077
pr107	44303	46678.154 0.016	48261.816 2.109	46226.276 0.074	45801.158 0.073
ch130	6110	7198.741 0.016	7142.045 7.688	7007.479 0.09	6754.876 0.09
kroA150	26524	31482.020 0.047	31442.994 11.094	29742.620 0.149	28939.435 0.148
kroB150	26130	31320.340 0.047	31519.083 11.156	29194.605 0.149	28607.636 0.149
kroA200	29368	34547.691 0.125	37650.812 0.450	33181.416 0.258	32817.646 0.251

7. Computational Experiments

In this section, we present the outcomes of the computational experiments carried out to assess the performance of our proposed algorithm. The algorithm was implemented in the Python programming language and tested on a system equipped with an Intel(R)

Xeon(TM) Silver 4114 CPU operating at 2.2GHz, complemented by 32 GB of RAM, running a 64-bit Windows operating system.

The algorithm was tested on problems obtained from the TSPLIB dataset [20]. The size of each problem instance is indicated by numerical labels appended to their titles. Table 1 illustrates the tour lengths obtained by the Nearest Neighbor (NN), Greedy heuristics, and our proposed algorithm. It is worth emphasizing that the best solutions attained by the algorithms are distinguished in shaded gray cells.

8. Conclusions

In this study, we present a new heuristic algorithm proposed for solving the symmetric Traveling Salesman Problem. This algorithm is based on the notion of convex hull. Furthermore, we conducted a series of experiments employing benchmark instances extracted from the TSPLIB dataset. We performed an exhaustive evaluation, comparing our results with those obtained using various established techniques. The results derived from our experiments unequivocally underscore the exceptional performance of our proposed heuristic. It is noteworthy that our heuristic surpasses the performance of several established methods in this specific context.

References

[1] Alizade, E. and Nuriyeva, F., (2021), An Iterative Heuristic Algorithm for Travelling Salesman Problem, Journal of Modern Technology and Engineering, 6(1), pp. 34-40.

[2] Appligate, D. L., Bixby, R. E., Chavatal, V. and Cook, W. J., (2006), The Travelling Salesman Problem: A Computational Study, Princeton University Press, Princeton and Oxford.

[3] Davendra, D., (2010), Travelling Salesman Problem, Theory and Applications, InTech.org.

[4] Johnson, D. S. and McGeoch, L. A., (1997), The Traveling Salesman Problem: A Case Study. Local Search in Combinatorial Optimization, John Wiley & Sons, pp. 215-310.

[5] Johnson, D. and Papadimitriou, C., (1985), Performance Guarantees for Heuristics. In Lawler et al., The Traveling Salesman Problem: A Guide for Tour of Combinatorial Optimization, Chapter 5, pp. 145-180.

[6] Kizilates, G. and Nuriyeva, F., (2013), On the Nearest Neighbor Algorithms for the Traveling Salesman Problem. Advances in Computational Science, Engineering and Information Technology, Springer, pp. 111-118.

[7] Kizilates, G. and Nuriyeva, F., (2013), A New Hybrid Heuristic Algorithm for Solving TSP, Anadolu University Journal of Science and Technology – B Theoretical Science, 2(2), pp. 143-148.

[8] Korte, B. and Vygen, J., (2007), Combinatorial Optimization: Theory and Algorithms, New York: Springer.

[9] Lawler, E. L., Lenstra, J. K., Rinnoy Kan, A. H. G. and Shmoys, D. B., (1985), The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization, John Wiley & Sons, 465 p.

[10] Matai, R., Singh, S. and Mittal, M. L., (2010), Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches, Traveling Salesman Problem, Theory and Applications, InTech.

[11] Nuriyev, U. and Nuriyeva, F., (2018), Practical Aspects of Solving Combinatorial Optimization Problems, Advanced Mathematical Models and Applications, 3(3), pp. 179-191.

[12] Nuriyev, U., Ugurlu, O. and Nuriyeva, F., (2018), Self-Organizing Iterative Algorithm for Travelling Salesman Problem, IFAC-Papers OnLine, 51(30), pp. 268-270.

[13] Nuriyeva, F., Kizilates, G. and Berberler, M. E., (2013), Improvements on Heuristic Algorithms for Solving Traveling Salesman Problem, International Journal of Advanced Research in Computer Science, 4(11), pp. 1-8.

[14] Ostrouchov, G. and Samatova, N. F., (2005), On FastMap and the Convex Hull of Multivariate Data: Toward Fast and Robust Dimension Reduction, IEEE Trans. Pattern Anal. Mach. Intell., 27, pp. 1340–1343. doi: 10.1109/TPAMI.2005.164

[15] Punnen, A., (2002), The Traveling Salesman Problem: Applications, Formulations and Variations, In Gutin and Punnen, Chapter 1, Kluwer Academic Publishers, Netherlands, pp. 1-28.

[16] Reinelt, G., (1994), The Traveling Salesman: Computational Solutions for TSP Applications, Springer-Verlag, Germany.

[17] Sharafutdinov, K., Bhat, J. S., Fritsch, S. J., Nikulina, K., Samadi, M. E., Polzin, R., Mayer, H., Marx, G., Bickenbach, J. and Schuppert, A., (2022), Application of Convex Hull Analysis for the Evaluation of Data Heterogeneity between Patient Populations of Different Origin and Implications of Hospital Bias in Downstream Machine-Learning-Based Data Processing: A Comparison of 4 Critical-Care Patient Datasets. Front. Big Data, 5:603429. doi: 10.3389/fdata.2022.603429

[18] Shesu, R. V., Bhaskar, T. V. S. U., Rao, E. P. R., Ravichandran, M. and Rao, B. V., (2021), An Improved Method for Quality Control of *In Situ* Data from Argo Floats Using α Convex Hulls. MethodsX, 8, 101337. doi: 10.1016/j.mex.2021.101337

[19] Zhou, X. and Shi, Y., (2009), Nearest Neighbor Convex Hull Classification Method for Face Recognition. In: Allen, G., Nabrzyski, J., Seidel, E., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds) Computational Science – ICCS 2009, ICCS 2009, Lecture Notes in Computer Science, 5545. Springer, Berlin, Heidelberg.

[20] TSPLIB. `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/` (accessed November 5, 2023).

Fidan Nuriyeva received the B.S., M.S. and Ph. D. degrees in Mathematics from Ege University, Izmir, Türkiye. Since 2014, she works in Dokuz Eylul University, Department of Computer Science, Izmir, Türkiye. At the present time, she has focused on combinatorial optimization, operations research and computer mathematics.

Hakan Kutucu received the B.S. degree from Dokuz Eylul University, Department of Mathematics, M.S. and Ph. D. degrees in mathematics from Ege University, Izmir, Türkiye. He works in Karabuk University, Department of Software Engineering, Karabuk, Türkiye. At the present time, he has focused on network design problems, combinatorial optimization and mathematical modeling.