Süleyman Demirel Üniversitesi
Fen Bilimleri Enstitüsü Dergisi
Cilt 30, Sayı 2, 280-299, 2026

Süleyman Demirel University
Journal of Natural and Applied Sciences
Volume 30, Issue 2, 280-299, 2026

DOI: 10.19113/sdufenbed.1763233

TSSIA: A Novel Tabu Search Segment Improvement Algorithm for the Traveling Salesman Problem in Türkiye

Rıfat AŞLIYAN[*1]

[1]Aydın Adnan Menderes University, Faculty of Science, Department of Mathematic, Aydın, Türkiye

(Alınış / Received: 12.08.2025, Kabul / Accepted: 12.05.2026, Online Yayınlanma / Published Online: 25.08.2026)

Keywords
Traveling Salesman Problem,
Tabu Search Algorithm,
Ant Colony Optimization,
Artificial Bee Colony Algorithm,
Turkish provinces

Abstract: In the traveling salesman problem, the aim is for a salesperson to start from one province, pass through each province only once, and come back to the same province. However, this problem can be reached by discovering the shortest possible route in total. When the number of provinces is large, it is a very difficult and time-consuming problem to solve. Therefore, it is generally attempted to be solved with metaheuristic methods. In this work, the traveling salesman problem was attempted to be solved using the distances between the provinces of Türkiye with Ant Colony Optimization, Artificial Bee Colony, and Tabu Search algorithms. The shortest routes were found for each method, and these methods have been compared with each other. As a result, it was seen that the most successful method was the Tabu Search Algorithm. In addition, the best-known route has been determined by adding segment improvement to the Tabu Search Algorithm. While the best-known route for the Traveling Salesman Problem involving Turkish cities was previously 9966 km, it has been reduced to 9906 km using the Tabu Search Segment Improvement Algorithm proposed in this study, achieving a reduction of 60 km.

TSSIA: Türkiye'deki Gezgin Satıcı Problemi için Yeni Bir Tabu Arama Segment İyileştirme Algoritması

Anahtar Kelimeler
Gezgin Satıcı Problemi,
Tabu Arama Algoritması,
Karınca Kolonisi Optimizasyonu,
Yapay Arı Kolonisi Algoritması,
Türkiye şehirleri

Öz: Gezgin satıcı probleminde, bir satıcının bir şehirden başlayarak her şehirden sadece bir kez geçilmesi ve aynı şehre geri dönmesi amaçlanır. Fakat, bu problem, toplamda en kısa rotanın bulunmasıyla çözülür. Şehir sayısı fazla olduğunda çözülmesi çok zor ve zaman alan problemlerdendir. Bu yüzden, genelde meta sezgisel metotlarla çözülmeye çalışılır. Bu çalışmada, Türkiye'nin şehirlerinin mesafeleri kullanılarak Karınca Kolonisi, Yapay Arı Kolonisi ve Tabu Arama algoritmaları ile gezgin satıcı problemi çözülmeye çalışılmıştır. Her metot için en kısa rotalar bulunmuş ve bu metotlar karşılaştırılmıştır. Bu çalışma sonucunda en başarılı metot Tabu Arama algoritması olduğu görülmüştür. Bunula birlikte, Tabu Arama algoritmasına segment iyileştirme eklenmesi suretiyle bilinen en iyi rota tespit edilmiştir. Türkiye şehirleriyle oluşturulan gezgin satıcı probleminin şimdiye kadarki en iyi rotası 9966 km iken bu çalışmada önerilen Tabu Arama Segment İyileştirme algoritmasıyla 60 km azaltılarak 9906 km'ye kadar düşürülmüştür.

1. Introduction

Traveling Salesman Problem (TSP) can be defined as the followings: Given a list of provinces, one is chosen as the starting point, and the goal is to go back to the initial province after visiting each of the other provinces once. However, the total distance traveled must be minimized. Essentially, the solution to this problem requires finding the sequence of provinces to visit that results in the shortest possible route. Although the TSP is very simple to understand, its solution is not so easy. When the number of provinces is small, the exact solution can be found quickly. But, when the number of provinces becomes larger, resolving this TSP is getting harder. In terms of computational complexity, considering there are n provinces, the number of tours rises factorially with $(n-1)!/2$. For this reason, the TSP is among the NP-hard problems [1-4].

* Corresponding author": rasliyan@adu.edu.tr

When we review the historical development of the TSP, we can see that this problem has been studied for 200 years. However, it was first mathematically formulated by the Austrian mathematician Karl Menger. Later, Merrill Flood from Princeton University gave this problem the name "Traveling Salesman Problem", and it became a extensively researched problem. In the 1950s and 1960s, very successful studies were conducted for the exact solution of the TSP. Scientists such as Ray Fulkerson, Selmer Johnson, and George Fulkerson succeeded in obtaining exact solutions when the number of provinces was small by using the cutting-plane method and linear programming. Today, numerous studies on the TSP continue to be carried out [1, 4-7].

1.1. Mathematical formulation

The TSP can be modeled as a graph structure. That's why each province can be defined as a node or vertex, while the roads connecting the provinces can be defined as edges. The edge weights can then represent the distances between provinces.

Different forms of mathematical formulations exist. However, the Integer Linear Programming model is most often preferred [6-11].

Equation 1 represents the standard objective function of the TSP with constraints as shown in Equation 2.

$$Min \sum_{i=1}^{n} \sum_{i \neq j,\ j=1}^{n} c_{ij} x_{ij} \quad (1)$$

c_{ij}: The road length between province *i* and province *j*. n: The number of provinces. x_{ij} takes a value of 1 or 0. If the route continues from province *i* to province *j*, x_{ij}=1, otherwise x_{ij}=0.

Constraints:

$$\sum_{i=1,\ i \neq j} x_{ij} = 1\,, \forall j \in C \quad (2)$$

The constraint in Equation 2 states that every province is visited once. C represents the set of provinces.

$$\sum_{j=1,\ j \neq i} x_{ij} = 1\,, \forall i \in C \quad (3)$$

The constraint in Equation 3 indicates that each province is departed from exactly once.

$$u_i - u_j + n.x_{ij} \leq n - 1, \forall i \neq j \quad (4)$$

Meanwhile, the constraint in Equation 4 ensures that the solution consists of a single, continuous tour. This constraint is known as the Miller-Tucker-Zemlin formulation [7, 9, 10].

Numerous studies have been conducted to solve the TSP, employing a variety of methods. These methods are broadly categorized into three main classes: Approximation, exact, and heuristic or metaheuristic algorithms.

Exact algorithms find the complete solution to the problem, meaning the optimal solution is found, but their computation time is very long. One such exact algorithm, the Brute Force Algorithm, finds the most suitable solution by trying all existing possibilities. The number of all trial solutions is calculated by the formula $(n-1)!/2$. Branch and Bound Algorithm searches for the best solution by creating a tree structure and pruning the branches that yield poor results. The Held-Karp Algorithm, which is expressed as a dynamic programming method, is an algorithm that can be computed as the number of provinces is under twenty and has $O(n^2 2^n)$ time complexity. In the Cutting Plane Algorithm, a solution is reached by starting with a linear programming relaxation and continuing with subtour elimination.

Examples of approximation algorithms include the Nearest Neighbor and the Christofides-Serdyukov Algorithms. These algorithms cannot guarantee that the best solution will be found. They offer a solution that is theoretically presented to be approximately close to the optimal solution by a certain ratio. The Nearest Neighbor method, which is a greedy algorithm, starts from one province and proceeds by visiting the nearest unvisited province. This process continues until all provinces are used, and a solution is sought. The Christofides-Serdyukov Algorithm is among the approximation algorithms that yield very good solutions. It uses the Euclidean metric for distance and can find a solution that is, at most, 1.5 times the optimal solution length.

For solving complex and real-world problems, the most commonly chosen methods are generally heuristic and metaheuristic algorithms. These algorithms do not guarantee the best solution. However, they can find near-optimal solutions in short time. Examples of these algorithms include Local Search and k-opt, Simulated Annealing (SA), Ant Colony Optimization (ACO), Genetic Algorithm, Tabu Search Algorithm (TSA), and Artificial Bee Colony Algorithm (ABCA). K-opt and local search algorithms attempt to create new solution tours by swapping edges in an existing tour. However, it can get stuck in local optima. The SA algorithm, benefitted from annealing of metals in metallurgy, can avoid local optima by also considering worse solutions, thus allowing it to approach the best solution. Genetic Algorithm attempts to obtain new random solutions through crossover and mutation, which occur in evolution. Similarly, better solutions are detected by further crossover and mutation on the best-selected solutions. The TSA, in order to escape local optima,

places solutions in a tabu list for a certain time, which can cause to find better solutions. ACO is a metaheuristic method which uses pheromones by ants to find food sources. Ants initially begin searching for food sources arbitrarily. When they find the sources, they leave a pheromone substance on the nest path, and the pheromone increases on this path. Ants prefer the path with the most pheromone. In this way, the shortest path is identified.

TSP is classified as symmetric or asymmetric depending on whether the distances between two provinces are equal or not. To solve a TSP, the distances from every province to all other provinces must be known. The main purpose is to find the shortest route based on these distances. If for any two provinces the distance between them is the same in both ways, meaning $c_{ij} = c_{ji}$, it is a symmetric problem. If the distances between at least two provinces are different, meaning $c_{ij} \neq c_{ji}$, it will be an asymmetric TSP [9-10].

Several studies have been conducted on the TSP for Turkish provinces. In a study by Köse and Erdoğmuş (2019), systems were developed and compared using Particle Swarm Optimization (PSO), Imperialist Competitive Algorithm (ICA), SA and ACO. The shortest route was found by ICA to be 10394 km [12].

In a study by Aşlıyan (2021), GA and TSA were used, and the best route of 10121 km was found with the TSA[4].

Similarly, in another study by Aşlıyan (2022), ACO, Standard SA, and Population-Based SA Algorithms were used and their performances were compared. Among these methods, the best tour result of 9998 km was achieved with the Population-Based SA algorithm [2].

In the study conducted by Dikmen et al. (2014), Genetic Algorithm and ACO were used and compared. In this study, the most successful result was a route distance of 9966 km, achieved with the ACO [5].

In the systems implemented by Şenaras and İnanç (2017), the ACO was utilized to search the best solution. In solving the TSP for 20 provinces in Türkiye, the shortest route found had a distance of 2529 km [13].

In their study, Ertuğrul and Özçil (2016) addressed TSP with Genetic Algorithm. To optimally schedule the tours of political parties across 21 provinces in Türkiye, a tour planning analysis was conducted using the Genetic Algorithm, and a shortest route of 4034 km was achieved [14].

Hasan and Aladdin (2025) reviews the real-world applications of metaheuristic algorithms. And it evaluates 48 algorithms from 2020 to 2024. It mentions their effectiveness across six prevalent applications. It emphasizes the necessity of selecting appropriate algorithms based on task complexity. The study categorizes and compares these algorithms, and it also reveals that no single algorithm surpasses in all problems. This comprehensive review contributes valuable insights into the many applications [15].

Metaheuristic algorithms can be applied to many areas as logistics, healthcare, finance, and engineering for the solution of hard optimization tasks. The effectiveness of these algorithms in navigating large, nonlinear, and uncertain search spaces makes them more preferable over traditional techniques. Almufti (2025) mentions recent advancements, real-world applications, and hybridizations. This review book aims to guide students, researchers, and practitioners in both the implementation and innovation of metaheuristic optimization techniques [16].

Metaheuristic algorithms are widely applied in various real-world domains such as engineering, logistics, and operations research. They effectively address complex combinatorial optimization problems by balancing computing efficiency and solution quality. Rashid and Zebari (2025) highlights practical implementations, including the Fire Hawk Optimizer, showcasing how these algorithms can be utilized in real-world problems. The work emphasizes the importance of developing reliable and efficient metaheuristic algorithms to tackle complicated optimization challenges [17].

Pop (2020) presents a comprehensive survey on the generalized traveling salesman problem (GTSP). It covers the mathematical formulations, solution approaches, and real-life applications. This paper also mentions the results, datasets, and performance analysis of existing algorithms [18].

Soni et. al. (2021) focuses on many nature-inspired optimization algorithms. This paper also presents their effectiveness, and the gaps between theoretical and practical applications [19].

By integrating the inherent strengths of various algorithms, hybrid meta-heuristics improve solution quality in TSP while providing effective strategies to minimize computational costs.

The advantages of hybridizing algorithms were exemplified by Liu et al. (2023), who developed SAACO and TSACO. By combining ACO with SA and TS, these hybrid approaches achieved superior performance in the quality of solution [20].

Jiang et al. (2020) introduces a hybrid algorithm, Ant Colony Optimization-Partheno Genetic Algorithms (AC-PGA), designed to solve the Multiple Traveling Salesmen Problem (MTSP) involving multiple depots and specific constraints for the cities per salesperson.

The methodology employs a dual-stage approach where PGA explores the initial variable space while ACO is utilized for the precise optimization of subsequent variables. AC-PGA offers superior performance and scalability for large-scale datasets [21].

Yahia et al. (2020) structures a hybrid method which includes ACO combined with Neighbor Joining (NJ) technique and call it a Neighbor Joining ACO met (NACO) to get accurate results [22].

Qamar et al. (2021) proposes an enhanced optimization framework for TSP by integrating a Best-Worst Ant System (BWAS) with PSO to address the efficiency limitations of standard ACO methods. Experimental results indicate that using PSO to initialize the best particle trails significantly accelerates convergence and improves solution quality compared to traditional ACO and hybrid PSO-ACO methods [23].

Kanna et al. (2021) introduces a hybrid meta-heuristic algorithm, Earthworm-based Deer Hunting Optimization Algorithm (DHOA) (EW-DHOA), which integrates DHOA and the EW Optimization (EWO) to address the combinatorial complexity of the TSP. By minimizing total travel distance and optimizing path selection, the proposed model demonstrates superior convergence rates and reduced computational complexity compared with traditional heuristic techniques [24].

The hybridization of an improved discrete Cuckoo Search with the 3-Opt local search, as proposed by Sarucan and Berkaya (2024), has shown to be a robust strategy. This method successfully achieved optimal results with fewer than 150 nodes [25].

Bai et al. (2024) developed the ACS-GA hybrid method, where ACO and GA as independent processes that periodically exchange their best-found solutions. This collaborative approach significantly enhances both the stability of the search and the total quality [26].

Hybrid meta-heuristics, like MACO-LKH Algorithm proposed by Yang et al. (2024), efficiently solve TSP by combining multi-colony ant optimization with Lin-Kernighan's dynamic visibility and pheromone update strategies, leading to quicker convergence on shorter paths and improved solution quality compared to traditional methods [27].

Hybrid meta-heuristics, like Spotted Hyena-based Rider Optimization Algorithm (S-ROA) proposed by Murali Krishna et al. (2021), combine strengths of existing algorithms to minimize the travel distance in TSP, enhancing solution robustness and effectiveness compared to traditional methods and state-of-the-art algorithms [28].

Hybrid meta-heuristics, like Hybrid Genetic Algorithm and ACO (HGAACO) proposed by Thongpiem et al. (2024), enhance TSP solutions by combining strengths of GAs and ACO, improving search performance and optimizing total travel distance effectively compared to traditional methods [29].

Overall, these works show that hybridizing meta-heuristics not only compensates for the weaknesses of single-method approaches but also scales efficiently with TSP complexity, marking it as a significant frontier in optimization research [30, 31].

The importance of this study can be stated as follows. First, TSP, formulated in the context of the provinces of Türkiye, was addressed using three different metaheuristic algorithms that were designed and compared with each other. Among these, the TSA was identified as the most successful. Second, the shortest route to date for this specific TSP has been identified. Finally, the best-known route has been achieved by adding a segment improvement module to the TSA.

It has been observed in previous studies [2, 4, 5, 12-14] that the solutions (best routes) obtained contain incorrect orderings in certain segments, and the applied metaheuristic algorithms struggle to detect these specific misorderings. To address these deficiencies, a Segment Improvement module has been integrated into the TSA, which has yielded the most successful results for the TSP regarding Turkish cities. While the best-known route prior to this study was 9966 km [5], the proposed Tabu Search Segment Improvement Algorithm (TSSIA) achieved a reduction of 60 km, establishing a new best route of 9906 km. Therefore, by focusing on specific segments, these incorrect orderings have been minimized.

The remainder of this paper has been planned as follows. The Material and Method section explains the metaheuristic algorithms utilized in this study. The Results and Discussion part gives the experimental results obtained with these methods. In Conclusion section, the obtained results are interpreted and compared.

2. Material and Method

To solve the TSP for the provinces of Türkiye, this work utilizes TSA, TSSIA, ACO, and ABCA. Accordingly, these methods, including their algorithms and flow diagrams, have been detailed below.

2.1. Tabu Search Algorithm (TSA)

TSA was developed by Fred W. Glover in 1986. Designed as a search algorithm, it is generally effective for solving combinatorial optimization problems [6, 7, 32-37]. As shown in Figure 1, TSA is one of the metaheuristic methods created to counter the problem of methods like Hill Climbing converging to

local optimum value. Some local search algorithms try to find the best solutions by using neighboring solutions derived from existing ones. However, most of them converge to local optima values and cannot find the global optimum solution. The TSA attempts to solve this problem by using a memory mechanism called a tabu list. In other words, it tries to escape the local optimum solution to reach the global optimum. The tabu list is essentially a list of specific moves from recent solutions that are forbidden from being used for a certain period. When the time comes, these moves are removed from the list and can be used in solutions again. Thanks to the forbidden moves in the tabu list, it is possible to escape from solutions stuck in a local optima and reach better solutions [6, 7].

Tabu Search Algorithm:

1. An initial random solution is generated to begin. At the start, the tabu list contains no forbidden moves.
2. The neighboring solutions of the current solution are generated. This neighborhood structure varies depending on the problem. A neighboring solution can be the swapping of the order of two provinces for TSPs.
3. The fitness value of a neighboring solution is calculated with the problem's objective function.
4. The best solutions are identified from among the neighboring solutions which are not on the tabu list or that satisfy the aspiration criterion. The best neighbor found here may be worse than the best solutions.
5. The algorithm selects the best neighbor solutions.
6. The best-known solution is updated whenever a better solution is found.
7. A move is placed on the tabu list, where it remains for a predefined tenure.
8. The loop continues as long as the stopping condition is not met.

The tabu list is considered short-term memory. To ensure escape from local optima, moves are recorded in this list for a certain number of loop iterations. The duration or number of iterations that moves remain on this list is named as tabu tenure. If the tabu tenure is too short, the search may converge to a local optima. However, if a very long tenure is given, the number of available moves may decrease, and better moves could be prevented. Adjusting this tenure is very critical.

Several stopping criteria can be established to terminate the TSA. First, the algorithm can be ended when the iterations reach a maximum value. Second, the iteration is terminated if there is no improvement in the solution within a certain period. Finally, the algorithm's execution can also be stopped if a predetermined cost result has been reached.

Although adjusting the tabu tenure in the TSA is quite difficult, it can be said that this algorithm is advantageous due to its deterministic nature, its adaptability to many problems, and its ability to prevent getting stuck in local optima.

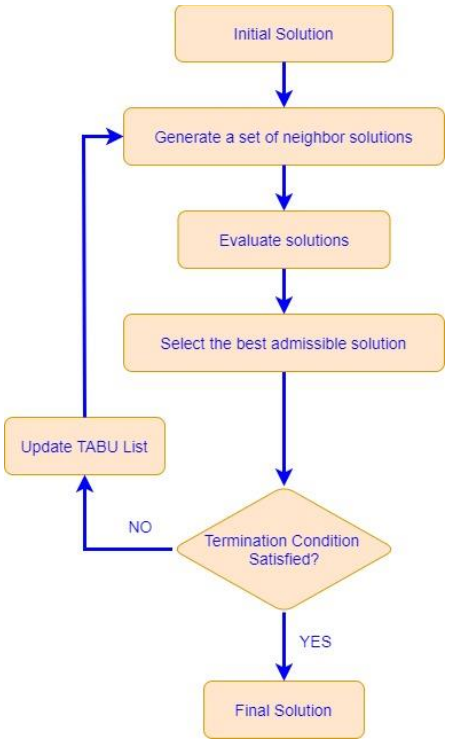

Figure 1. The flowchart of TSA.

2.2. Tabu Search Segment Improvement Algorithm (TSSIA)

The TSSIA is an extension of the standard TSA, with a segment improvement step added after step 5. In this new algorithm, this corresponds to Step 6. Using the standard TSA, an initial random solution is found, and by step 5, the best solution among its neighbors is identified. In Step 6, with Segment Improvement (SI), provinces randomly selected from within segments of a specific length are placed in front of other randomly selected provinces to prevent converging local minimum value. In this way, better solutions are obtained from the regions close to each province within the tour.

TSSI Algorithm:

1. An initial random solution is generated to begin. At the start, the tabu list contains no forbidden moves.
2. The neighboring solutions of the current solution are generated. This neighborhood structure varies depending on the problem. A neighboring solution can be the swapping of the order of two provinces for TSPs.
3. The fitness value of a neighboring solution is calculated with the problem's objective function.

4. The best solutions are identified from among the neighboring solutions which are not on the tabu list or that satisfy the aspiration criterion. The best neighbor found here may be worse than the current best solution.
5. The algorithm selects the best neighbor solution.
6. The best neighboring solution is selected as the new current solution.
7. Segment Improvement for TSA
    a. Determine Segment distance (*sd*).
    b. Determine the number of randomly province selection in the segment (*ssn*).
    c. Repeat
        i. Choose a segment of the solution in order.
        ii. Select a random province (*c1*) in *sd*.
        iii. Select another random province (*c2*) in *sd*.
        iv. Move *c1* in front of *c2* in *sd* to find a new solution.
        v. Repeat *ssn* times between i. and v. steps.
        vi. Choose the best solution
    d. Until all provinces are used in the solution tour.
8. A move is placed on the tabu list, where it remains for a predefined tenure.
9. The loop continues as long as the stopping condition is not met.

2.3. Ant Colony Optimization (ACO)

As displayed in Figure 2, ACO is a metaheuristic method designed by taking into account the manner of ants while they are trying to find food. It is frequently utilized for finding optimal solutions to combinatorial issues such as vehicle routing and TSP [38-42].

As it is known, ants use a chemical substance called pheromone. They leave this substance in the places they go while finding their food, because other ants can follow the pheromone. Many ants scatter randomly to search for food. Since they that search their food leave pheromone, a large amount of pheromone will accumulate on the shortest path to the food, and other ants will use this path. Thus, the shortest path is identified. The event where the pheromone amount increases significantly as other ants use this path is called positive feedback. Over time, this pheromone amount begins to disappear through evaporation. The pheromones on longer and less-used paths start to vanish, meaning that the selection of bad solutions is prevented. In the use of ACO, many ants come together to try to solve problems with collective intelligence.

While the ACO Algorithm is widely utilized in traveling salesman, assignment, and vehicle routing problems, it also has applications in job-shop scheduling, data clustering, image processing, and network routing problems.

ACO Algorithm:

1. Assignment of initial parameter values.
   α: The importance of the pheromone.
   m: The number of ants.
   ρ: The pheromone evaporation rate.
   β: The importance of heuristic information.
2. Assign initial pheromone (τ_0).
3. Construction of solutions: In this step, all ants construct their solutions. First, the ants are placed at the starting point. The ants then move to subsequent points based on the pheromone quantity and heuristic values. If the amount of pheromone for a point is high, the probability of moving to this point is high. Heuristic information, on the other hand, expresses the attractiveness of moving to that point. The lower the distance to that point, the higher its probability of being selected.

   The probability of moving between province *i* and province *j* is calculated as shown in Equation 5 and 6.

   η_{ij}: Represents the heuristic information, i.e., the attractiveness, between *i* and *j*.
   τ_{ij}: Indicates the pheromone quantity between *i* and *j*.
   d_{ij}: The distances between *i* and *j*.
   N_i: The points that ant *i* can travel to.
   P_{ij}: The probability of moving between *i* and *j*.

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{5}$$

$$P_{ij}^{k} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{k \in N_i^k}(\tau_{ik})^{\alpha}(\eta_{ik})^{\beta}}, \quad j \in N_i^k \tag{6}$$

4. Pheromone update: In this step, the pheromone values of the paths on the points that the ants have followed are changed. First, the pheromones are reduced. The purpose is to enable the discovery of new, different solutions. This situation is called pheromone evaporation. The second is the process of increasing the pheromones, that is, accumulation. Generally, ants leave more pheromones on the ways where better solutions exist. Specifically, the amount of pheromone is high in good solutions.
5. Stopping criteria: The loop is continued or terminated according to the stopping criteria. It can be the maximum number of repetitions, a period without finding a better solution, or when an acceptable solution is reached.

The ACO Algorithm is suitable for parallel computation because the ants can find solutions independently. Additionally, this algorithm can find effective solutions for very complex and large problems. Alongside its advantages, it can be slower to reach a solution when compared to some other metaheuristic methods. There is also a possibility of converging to local optimum values if the pheromone parameters are not set correctly.

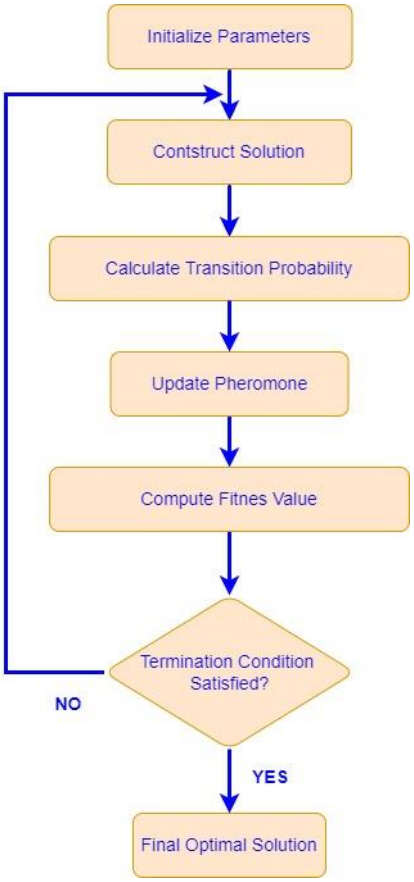

Figure 2. The flowchart of ACO.

2.4. Artificial Bee Colony Algorithm (ABCA)

As shown in Figure 3, the ABC Algorithm is a heuristic method developed by Derviş KARABOĞA and inspired by the food-finding behavior of a honey bee. It attempts to optimize and improve solution paths through swarm intelligence. This algorithm, designed in 2005, can solve many complex problems [43-47].

In a bee colony, scout bees try to find new food sources by flying randomly in all directions. When bees find food sources, they store the amount of nectar at that source and its distance from the beehive in their memory. Upon returning to their hive, they transmit information about the food source to the other bees by performing a "waggle dance". If a food source is very high quality, the bees perform more vigorous and longer-lasting dances. At the same time, the direction and form of the dance determine the source's distance and direction. Onlooker bees within the hive observe the dances performed by the employed bees and head towards the source of the most appealing one. When the nectars on the food source are consumed, they abandon it, and these bees turn back to scout bees to find new sources.

In ABC, food sources are thought as a solution of the problem. The nectar quantity is the solution's quality or fitness value. The bees' types in ABC Algorithm are onlooker, employed and scout bees. Onlooker bees observe the dances of the employed bees to decide which sources to visit. Employed bees are assigned for food sources and who try to find better sources in its vicinity. Scout bees are those that abandon their source when the nectar is exhausted and begin to search randomly for new sources.

ABC Algorithm:

1. Initialization of starting values: N solutions, i.e., food sources, are determined randomly, and employed bees are assigned to these sources.
2. Start of the main loop: These steps will continue until the termination criterion is met.
   a. First, an employed bee assigned to a random solution searches for new solutions in its neighborhood. If the new solutions' fitness values are better, the old source is forgotten, and the new source is adopted. Otherwise, the previous source will continue to be the best solution for that employed bee.
   b. After returning to the hive, information regarding a food source's location and nectar volume is transmitted from employed bees to onlooker bees within the hive via waggled dances. The onlooker bees make a probability-based selection using a roulette wheel mechanism. Food sources with higher nectar amounts and qualities will have a higher probability of being chosen. The onlooker bees, just like the employed bees, try to find better solutions in the neighborhood of the source and prefer the better one.
   c. If there is no increase or improvement in a food source or solution for a certain cycles, that food source is said to be run out. The employed bee assigned to this source, since the source is depleted, transforms into a scout bee and tries to find a random food source. This situation helps prevent getting stuck in local optima.

The ABCA has significant advantages when compared to some other heuristic algorithms, as it is simpler and more flexible, has fewer parameters, and has a lower potential to converge local optimum values due to the effect of scout bees. Additionally, it can be used for solving both continuous and discrete problems. On the other hand, it has disadvantages such as slow convergence in solving complex problems, sensitivity to parameters, and potential performance degradation when solving large-scale problems.

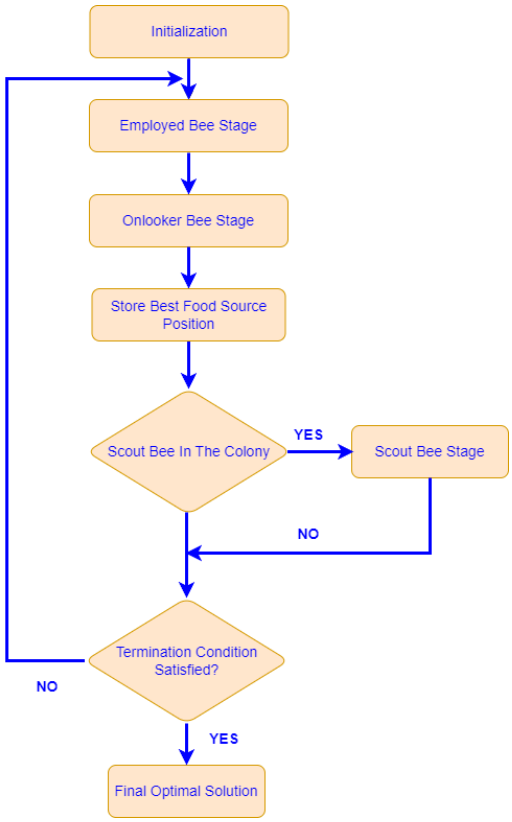

Figure 3. The flowchart of ABCA.

2.5. Time and space complexities of the algorithms

The time complexity of the ACO Algorithm applied to the TSP is given by $O(tmn^2)$ where t is the number of iterations, m is the number of ants, and n is the number of cities. This is derived from the fact that in each iteration, each ant constructs a solution by performing probability calculations for n steps, considering up to n possible connections at each step. The space complexity is $O(n^2)$, primarily mandated by the storage requirements for the distance and pheromone matrices ($n \times n$).

The time complexity of the ABCA for the TSP is estimated as $O(tSNn^2)$, where t denotes the maximum number of iterations (cycles), SN represents the number of food sources (population size), and n is the number of cities. Unlike constructive algorithms that build solutions step-by-step $O(n^2)$, ABC operates by iteratively modifying complete solutions using local search operators (such as swap or inversion), which typically requires $O(n)$ operations per bee to evaluate the new tour cost. The space complexity is $O(n^2 + SNn)$, accounting for the storage of the distance matrix and the population of solutions.

The time complexity of TS for the TSP is generally expressed as $O(tn^2)$, where t is the number of iterations and n is the number of cities. This complexity arises primarily from the size of the neighborhood. In a standard implementation using the Inversion or Swap operator, the number of possible neighbors scales with $O(n^2)$, Assuming an efficient "delta evaluation" technique is used to calculate the cost of a move in constant time $O(1)$, the evaluation of the entire neighborhood per iteration remains $O(n^2)$. The space complexity is $O(n^2)$, which is needed to store the distance matrix and, in some implementations, the recency-based frequency memory structures.

The time complexity of TSSIA for the TSP is expressed as $O(sltn^2)$, where t is the number of iterations, n is the number of cities and sl is the number of segments in the solution. The space complexity is $O(n^2)$, which is required to store the distance matrix and, in some implementations, the recency-based frequency memory structures.

3. Results and Discussion

The proposed metaheuristic algorithms have been implemented and executed within a consistent computational environment to ensure the reproducibility of the performance results. Both the development phase and the experimental simulations have been conducted on a system equipped with an Intel Core i7-12650H CPU (2.40 GHz), 16 GB of RAM, and the Windows 11 Operating System. The algorithms have been scripted and analyzed using the MATLAB R2024b environment. Furthermore, the real-world distance matrix for Türkiye's provinces was sourced from the official dataset provided by the General Directorate of Highways of Türkiye [48].

This work addresses the TSP across Türkiye's 81 provinces, using traffic license plate codes to model the optimal route. Given the vast combinatorial search space, metaheuristic approaches such as ACO, TSA, and ABCA have been employed to find efficient solutions. The specific solution representation, neighborhood structures, and algorithmic strategies used to navigate this discrete optimization challenge have been detailed in the following paragraphs.

Solution representation: The representation of solutions is encoded in the MATLAB environment as an 81-element array in the form: [2 63 47 21 72 56 73 30 65 13 49 12 23 44 62 24 29 69 25 4 76 36 75 8 53 61 28 52 55 57 37 78 74 67 14 81 54 41 34 59 39 22 17 10 16 77 11 26 43 3 64 45 35 9 48 20 32 15 7 42 70 68 40 71 6 18 66 19 5 60 58 38 50 51 33 1 80 46 31 79 27]. The integers here correspond to the Turkish traffic license plate codes of each city. For example, they are coded as Adana: 1, Adıyaman: 2, Osmaniye: 80 and Düzce: 81. The beginning of the solution indicates traveling from the city with plate code 2 (Adıyaman) to the city with plate code 63 (Şanlıurfa), and signifies that the distance between them can be calculated according to the distance matrix as shown in Table 1.

Table 1. City distance matrix for Turkish cities.

	City Codes	01	02	...	80	81
City Codes	City Names	Adana	Adıyaman	...	Osmaniye	Düzce
01	Adana		337	...	89	734
02	Adıyaman	337		...	244	966
...	...	...	...	...	...	...
80	Osmaniye	89	244	...		824
81	Düzce	734	966	...	824	

Initial population: The initial populations of all algorithms are generated randomly. However, the implementations are executed 20 times using seed values ranging from 1 to 20. Consequently, identical results will be produced for the same seed values.

Neighborhood structure: The neighborhood of ACO is the set of "Feasible Cities". The ants select from this set with a weighted probability based on pheromone and distance. In TSA, neighborhood structure is constructed using swap, reversion, and insertion operators. In ABCA, employed bees and onlooker bees explore neighboring solutions of the solution currently held in their memory. The operators utilized in TSA, namely swap, reversion, and insertion, are also utilized in ABC. But, in ABC, a random neighbor solution is generated.

Search space and bounds: The search space of this TSP includes all permutations of the 81 cities, representing all feasible Hamiltonian cycles. Because the problem is modeled as a discrete combinatorial optimization problem, the bounds are not defined by continuous numerical ranges, but by permutation constraints. Specifically, the solution is an array of size 81, where each element must be a unique integer corresponding to a city's traffic license plate code (ranging from 1 to 81). Hence, the search space size is 80!/2 The tours are undirected, which constitutes a huge combinatorial space. The algorithm explores this discrete space by modifying the sequence of cities while maintaining the validity of the permutation (i.e., each city is visited exactly once). The bee does not immediately accept the generated solution. If the new solution is more successful than the old one, the new solution is memorized; otherwise, the bee continues to keep the old solution.

TSA addresses the problem of cycling through its Tabu List and uses aspiration criteria to escape local optima. But, it relies mostly on the quality of the initial solution. ACO tackles the TSP by constructing paths probabilistically. However, to overcome stagnation and slow convergence, it needs pheromone evaporation and hybridization with local search. ABCA addresses the discrete nature of TSP by placing algebraic formulas with permutation-based operators (swap, inversion, and reversion) and uses "Scout Bee" mechanism to abandon stagnant solutions.

ACO results:

Table 2 displays the parameters used for ACO Algorithm in this work. These ACO applications have been implemented 20 times in the MATLAB with seed values ranging from 1 to 20. This ensures that the same results can be reproduced when these simulations have been run. Test implementations have been conducted using the values shown in parentheses under the "Setting Values" heading. For instance, attempts have been performed for "max iterations" values of 500, 750, and 1000; however, the most successful result was obtained with 1000. Similarly, formulas are provided in parentheses under "Setting Values", and the values are calculated based on these formulas. For instance, regarding the *tauInit* parameter, tests were carried out using (10*Q/(*ps**mean(distances))) (*yielding* 1.6420e-04).

Table 2. The parameters for ACO.

ACO Parameters	Descriptions	Setting Values
Attempts	The number of attempts for random seed=1,2,...,20	20
Max Iterations	Stopping condition on loop	1000 (500, 750, 1000)
Initial solutions	How to produce initial solutions	Random seed (1, 2,..., 20)
n	The number of city	81
ps	Population size (The number of ants)	100
Q	Pheromone Deposit value	1
tauInit	Initial pheromone	1.6420e-04 (10*Q/(*ps**mean(distances)))
alpha	Exponential weight of pheromone	1.2
beta	Exponential weight of heuristic	3
rho	The rate of evaporation	0.09
eta	Heuristic information matrix	1/distance matrix
tau	Pheromone matrix	*tauInit* x ones matrix

As seen in Table 3, the province codes and province names for the shortest tour obtained with the system created by the ACO are given in sequential order. It is understood that this tour starts from Ankara, proceeds finally to Bolu, and from there returns to Ankara.

This system has been implemented 20 times, and the shortest distance and elapsed time for each trial are

given in Table 4 and Figure 5. It can be seen here that the shortest distance was 10193 km in the 10th trial, with an elapsed time of 150.3 seconds.

Figure 4 [49] visually displays the drawing of the shortest tour on the map of Turkish provinces by the ACO system, which was created with MATLAB software.

This system attempts to find the most successful tour over 1000 iterations. The shortest distances obtained at the end of every iteration are displayed in the graph in Figure 6.

Table 3. The best tour of province codes and names for ACO.

Codes of the best tour
6, 18, 71, 40, 66, 19, 5, 60, 58, 38, 50, 68, 51, 70, 42, 3, 64, 43, 26, 11, 16, 77, 41, 54, 34, 59, 39, 22, 17, 10, 45, 35, 9, 48, 20, 15, 32, 7, 33, 1, 80, 46, 27, 79, 31, 2, 63, 47, 21, 72, 56, 73, 30, 65, 13, 49, 12, 23, 44, 62, 24, 29, 69, 25, 4, 76, 36, 75, 8, 53, 61, 28, 52, 55, 57, 37, 78, 74, 67, 81, 14
Names of the best tour
Ankara-Çankırı-Kırıkkale-Kırşehir-Yozgat-Çorum-Amasya-Tokat-Sivas-Kayseri-Nevşehir-Aksaray-Niğde-Karaman-Konya-Afyonkarahisar-Uşak-Kütahya-Eskişehir-Bilecik-Bursa-Yalova-Kocaeli-Sakarya-İstanbul-Tekirdağ-Kırklareli-Edirne-Çanakkale-Balıkesir-Manisa-İzmir-Aydın-Muğla-Denizli-Burdur-Isparta-Antalya-Mersin-Adana-Osmaniye-Kahramanmaraş-Gaziantep-Kilis-Hatay-Adıyaman-Şanlıurfa-Mardin-Diyarbakır-Batman-Siirt-Şırnak-Hakkari-Van-Bitlis-Muş-Bingöl-Elazığ-Malatya-Tunceli-Erzincan-Gümüşhane-Bayburt-Erzurum-Ağrı-Iğdır-Kars-Ardahan-Artvin-Rize-Trabzon-Giresun-Ordu-Samsun-Sinop-Kastamonu-Karabük-Bartın-Zonguldak-Düzce-Bolu

Table 4. The best tour distances and elapsed time for ACO.

Attempts	**Best Tour Distances (Kilometers)**	**Elapsed Time (Seconds)**
1	10313	149.3
2	10432	151.4
3	10304	149.8
4	10491	152.7
5	10406	152.9
6	10314	152.7
7	10283	151.1
8	10434	149.9
9	10336	150.8
10	**10193**	150.3
11	10412	151.9
12	10312	150.4
13	10479	152.5
14	10461	150.7
15	10463	150.4
16	10431	151.7
17	10276	150.4
18	10263	150.5
19	10275	151.9
20	10453	150.2
Min	**10193**	149.3
Max	10491	152.9
Mean	10366.6	151.1
Std	86.1	1.05

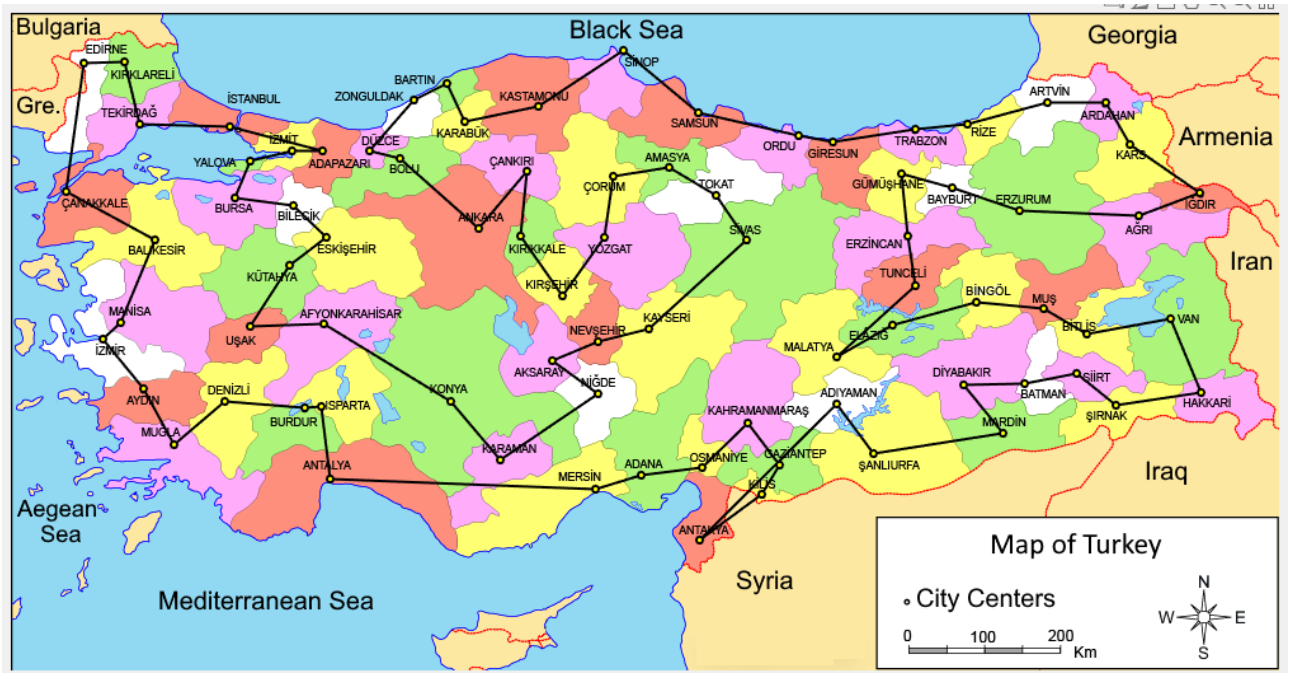

Figure 4. The best tour drawn on Türkiye province map for ACO [49].

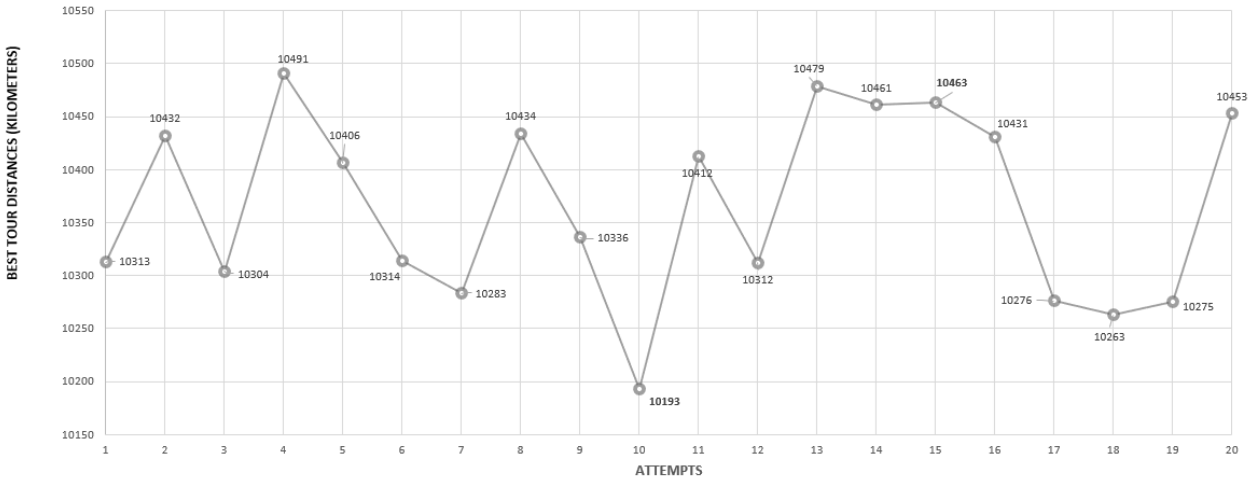

Figure 5. The tour distances of twenty attempts for ACO.

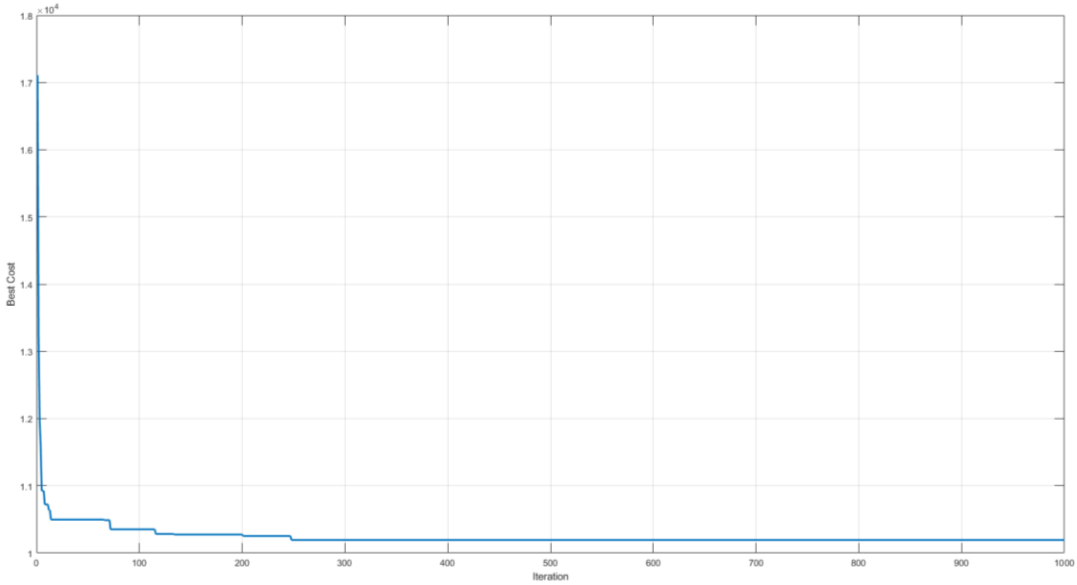

Figure 6. The best costs of 1000 iterations for ACO.

ABCA results:

In Table 5, the parameters of ABCA Algorithm have been presented in this study. These ABCA implementations have been executed 20 times in the MATLAB with seeds (1, 2, ..., 20). It is provided that the same results can be reproduced when the simulations have been run. Test implementations have been conducted using the values shown in parentheses under the "Setting Values" heading.

Table 5. The parameters for ABCA.

ABCA Parameters	Descriptions	Setting Values
Attempts	The number of attempts for random seed=1,2,...,20	20
Max Iterations	Stopping condition on loop	1000 (500, 750, 1000)
Initial solutions	How to produce initial solutions	Random seed (1, 2,..., 20)
n	The number of city	81
NumScoutBee	Total scout bees	200
radDamp	The rate of neighborhood damp	0.95
rad	The radius of neighborhood	8.1 (*n*/10)
numSelectedSiteBee	The number of recruited bees for selected sites	100 (*NumScoutBee*/2)
numEliteSiteBee	The number of recruited bees for elite sites	200 (*numSelectedSiteBee* x 2)
numSelectedSite	The number of selected sites	140 (*NumScoutBee* x 0.7)
numEliteSite	The number of elite sites	84 (*numSelectedSite* x 0.6)

The most successful route found by the system developed using the ABCA is presented in Table 6, along with the province codes and names. According to this route, the tour starts from Muğla, passes through various provinces in sequence, reaches Aydın, and then returns to Muğla.

The system's performance has been tested by running it 20 times. The shortest distances obtained in each trial are shown in Figure 8 and Table 7. The elapsed times to determine these distances are detailed in Table 7. Among the trials conducted, it is seen that the best result was achieved in the 15th trial with a shortest distance of 10456 km, and this tour took 118.1 seconds.

Furthermore, the shortest tour found by the ABCA, visualized on the map of Türkiye using MATLAB, is presented in Figure 7.

This system performs 1000 iterations to find the most optimal route. The shortest distances found as a result of each iteration are shown in the graph in Figure 9.

Table 6. The best tour of province codes and names for ABCA.

Codes of the best tour:
48, 20, 64, 3, 32, 15, 7, 42, 70, 33, 1, 31, 79, 27, 63, 2, 46, 80, 51, 38, 50, 68, 40, 66, 71, 6, 26, 43, 11, 16, 77, 41, 54, 67, 74, 78, 37, 57, 55, 52, 28, 61, 53, 8, 75, 36, 76, 4, 25, 69, 29, 24, 62, 12, 49, 13, 65, 30, 73, 56, 72, 47, 21, 23, 44, 58, 60, 5, 19, 18, 14, 81, 34, 59, 39, 22, 17, 10, 45, 35, 9
Names of the best tour:
Muğla-Denizli-Uşak-Afyonkarahisar-Isparta-Burdur-Antalya-Konya-Karaman-Mersin-Adana-Hatay-Kilis-Gaziantep-Şanlıurfa-Adıyaman-Kahramanmaraş-Osmaniye-Niğde-Kayseri-Nevşehir-Aksaray-Kırşehir-Yozgat-Kırıkkale-Ankara-Eskişehir-Kütahya-Bilecik-Bursa-Yalova-Kocaeli-Sakarya-Zonguldak-Bartın-Karabük-Kastamonu-Sinop-Samsun-Ordu-Giresun-Trabzon-Rize-Artvin-Ardahan-Kars-Iğdır-Ağrı-Erzurum-Bayburt-Gümüşhane-Erzincan-Tunceli-Bingöl-Muş-Bitlis-Van-Hakkari-Şırnak-Siirt-Batman-Mardin-Diyarbakır-Elazığ-Malatya-Sivas-Tokat-Amasya-Çorum-Çankırı-Bolu-Düzce-İstanbul-Tekirdağ-Kırklareli-Edirne-Çanakkale-Balıkesir-Manisa-İzmir-Aydın

Table 7. The best tour distances and elapsed time for ABCA.

Attempts	**Best Tour Distances (Kilometers)**	**Elapsed Time (Seconds)**
1	10664	112.6
2	11244	112
3	11207	115
4	10721	117.4
5	11094	113.5
6	10816	112.3
7	10604	113.1
8	11112	112.6
9	10525	113.8
10	11304	116.5
11	11068	116.5
12	11300	114
13	10757	112.6
14	10965	113
15	**10456**	118.1
16	10885	114.7
17	10637	113.2
18	11184	113.4
19	11556	113.6
20	11300	112.5
Min	**10456**	112
Max	11556	118.1
Mean	10970	114
Std	302.8	1.74

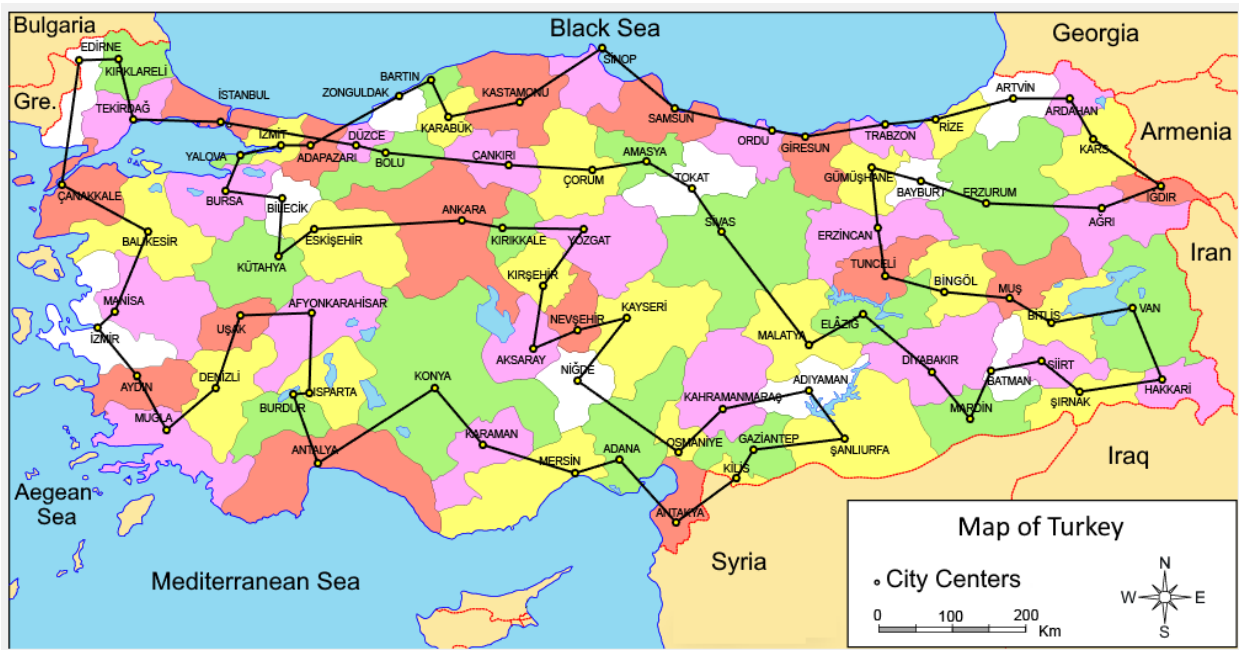

Figure 7. The best tour drawn on Türkiye province map for ABCA [49].

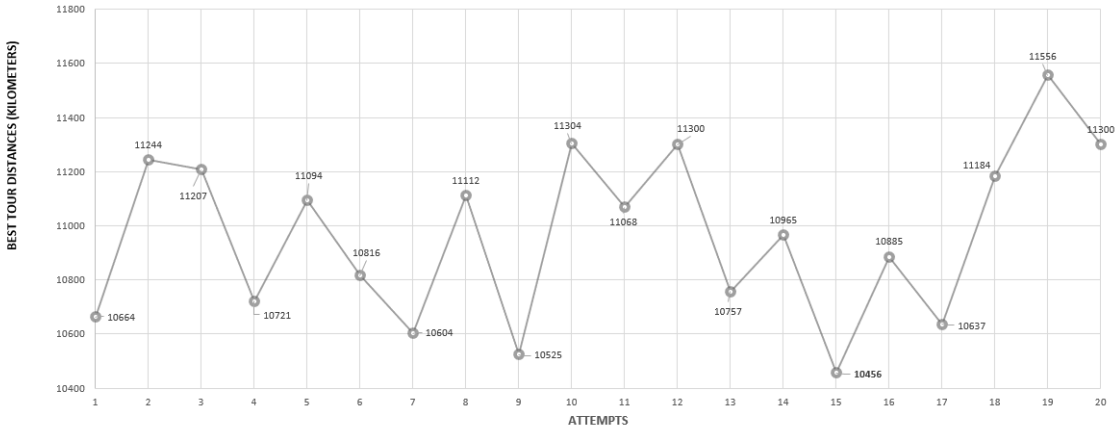

Figure 8. The tour distances of twenty attempts for ABCA.

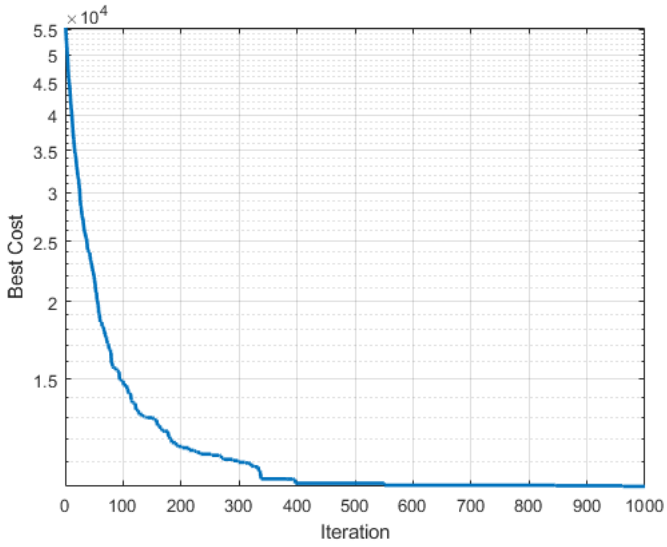

Figure 9. The best costs of 1000 iterations for ABCA.

TSA results:

Table 8 presents the parameters used for the implementation of the TSA method in this study. These TSA implementations were executed 20 times with seed values ranging from 1 to 20. This ensures that the same results can be reproduced when these simulations are run. Tests were also conducted using the values shown in parentheses under the "Setting Values" heading. For instance, trials were performed for "max iterations" values of 500, 750, and 1000; however, the most successful result was obtained with 1000. Similarly, formulas are provided in parentheses under "Setting Values", and the values are calculated based on these formulas. For example, regarding the "*n_swap*" parameter, tests were carried out using $n(n\text{-}1)/2$ (*yielding* 3240), $n(n\text{-}1)/3$ (*yielding* 2160), and $n(n\text{-}1)/4$ (*yielding* 1620). Since the value of 3240 proved to be the most successful, the results are presented based on this value.

Table 8. The parameters for TSA.

TSA Parameters	Descriptions	Setting Values
Attempts	The number of attempts for random seed=1,2,…,20	20
Max Iterations	Stopping condition on loop	1000 (500, 750, 1000)
Initial solutions	How to produce initial solutions	Random seed (1, 2,..., 20)
Criteria of aspiration	The condition of overriding tabu status	Improving the best solution
n	The number of city	81
n_swap	The number of swapping	3240 ($n(n\text{-}1)/2$=3240, $n(n\text{-}1)/3$=2160, $n(n\text{-}1)/4$=1620))
n_reversion	The number of reversion	3240 ($n(n\text{-}1)/2$=3240, $n(n\text{-}1)/3$=2160, $n(n\text{-}1)/4$=1620))
n_insertion	The number of insertion	6561(n^2= 6561, $n^2/2$= 3280 $n^2/3$= 2187)
n_action	The number of actions (Neighborhoods)	13041 (13041, 9760, 5427) (*n_swap* +

		n_reversion + *n_insertion*)
TL	Tabu length	1304 (1304 , 976, 542)(*n_action*/10)

Table 9 shows the optimal travel route determined by the system developed based on the TSA, along with the corresponding province codes and names. This route, starting from Mersin, covers various provinces before arriving at Karaman as the final stop and then returns to the starting point of Mersin.

The results of 20 separate trials conducted to measure the system's effectiveness are presented in Table 10 and Figure 11 with the shortest distance and elapsed time data. The most successful result achieved from these trials was recorded in the 11th trial with a distance of 9936 kilometers, and the completion time for this tour was measured as 52.2 seconds. A visual representation of this shortest route, generated on the map of Türkiye by the TSA system is located in Figure 10. The system undergoes a process of 1000 iterations to identify the most ideal tour. The sequence of the shortest distances obtained at each step of this process is shown in detail in the graph in Figure 12.

Table 9. The best tour of province codes and names for TSA.

Codes of the best tour:
33, 1, 80, 31, 79, 27, 46, 2, 63, 47, 21, 72, 56, 73, 30, 65, 13, 49, 12, 44, 23, 62, 24, 29, 69, 25, 4, 76, 36, 75, 8, 53, 61, 28, 52, 55, 57, 19, 5, 60, 58, 38, 50, 51, 68, 40, 66, 71, 6, 18, 37, 78, 74, 67, 14, 81, 54, 11, 16, 77, 41, 34, 59, 39, 22, 17, 10, 45, 35, 9, 48, 20, 64, 43, 26, 3, 32, 15, 7, 42, 70
Names of the best tour:
Mersin-Adana-Osmaniye-Hatay-Kilis-Gaziantep-Kahramanmaraş-Adıyaman-Şanlıurfa-Mardin-Diyarbakır-Batman-Siirt-Şırnak-Hakkari-Van-Bitlis-Muş-Bingöl-Malatya-Elazığ-Tunceli-Erzincan-Gümüşhane-Bayburt-Erzurum-Ağrı-Iğdır-Kars-Ardahan-Artvin-Rize-Trabzon-Giresun-Ordu-Samsun-Sinop-Çorum-Amasya-Tokat-Sivas-Kayseri-Nevşehir-Niğde-Aksaray-Kırşehir-Yozgat-Kırıkkale-Ankara-Çankırı-Kastamonu-Karabük-Bartın-Zonguldak-Bolu-Düzce-Sakarya-Bilecik-Bursa-Yalova-Kocaeli-İstanbul-Tekirdağ-Kırklareli-Edirne-Çanakkale-Balıkesir-Manisa-İzmir-Aydın-Muğla-Denizli-Uşak-Kütahya-Eskişehir-Afyonkarahisar-Isparta-Burdur-Antalya-Konya-Karaman

Table 10. The best tour distances and elapsed time for TSA.

Attempts	**Best Tour Distances (Kilometers)**	**Elapsed Time (Seconds)**
1	10119	51.8
2	10041	51.6
3	10028	51.2
4	10036	51.4
5	10007	51.1
6	10135	51.2
7	10026	52.2
8	9960	52.2
9	10185	52.2
10	10123	51.9
11	**9936**	52.2
12	9942	52.1
13	10185	52.1
14	10197	51.6
15	10000	51.3
16	10093	51.1
17	9955	51.4
18	9963	51.3
19	10225	51.5
20	10119	51.5
Min	**9936**	51.1
Max	10225	52.2
Mean	10063.8	51.6
Std	90.5	0.4

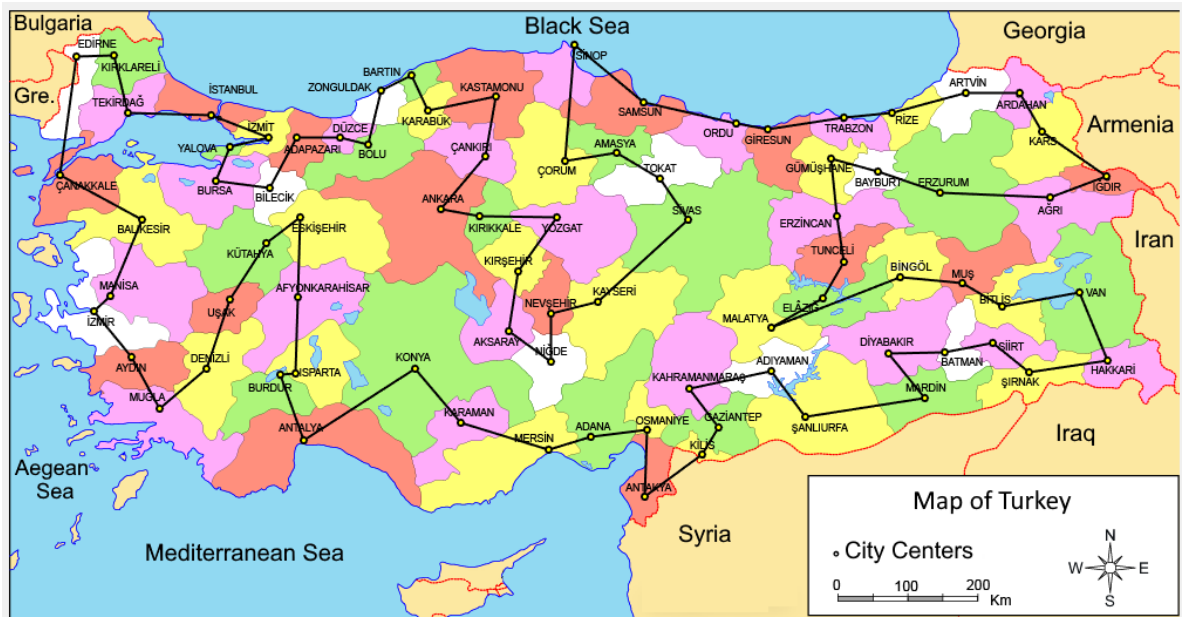

Figure 10. The best tour drawn on Türkiye province map for TSA [49].

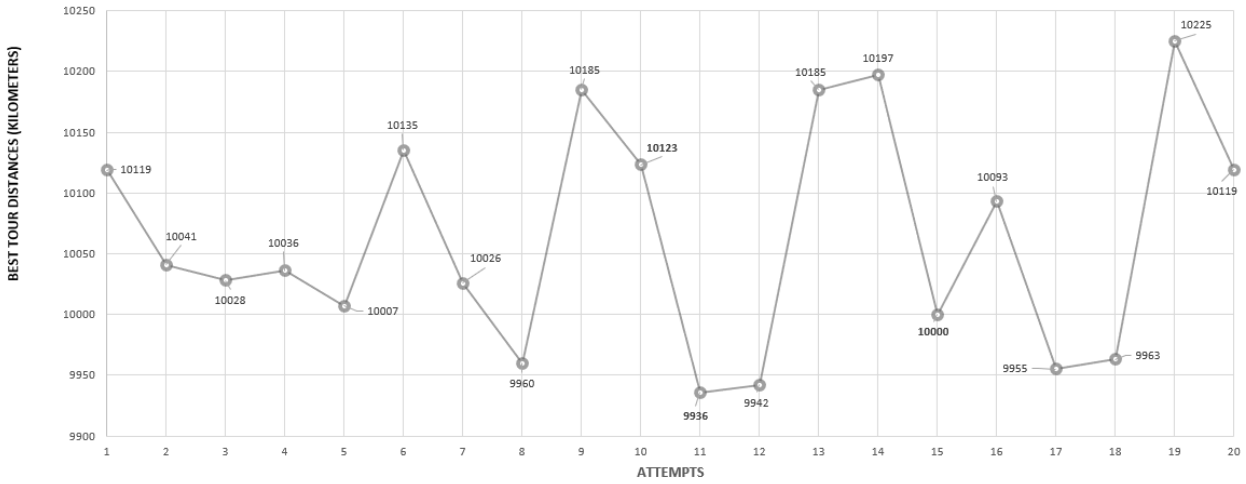

Figure 11. The tour distances of twenty attempts for TSA.

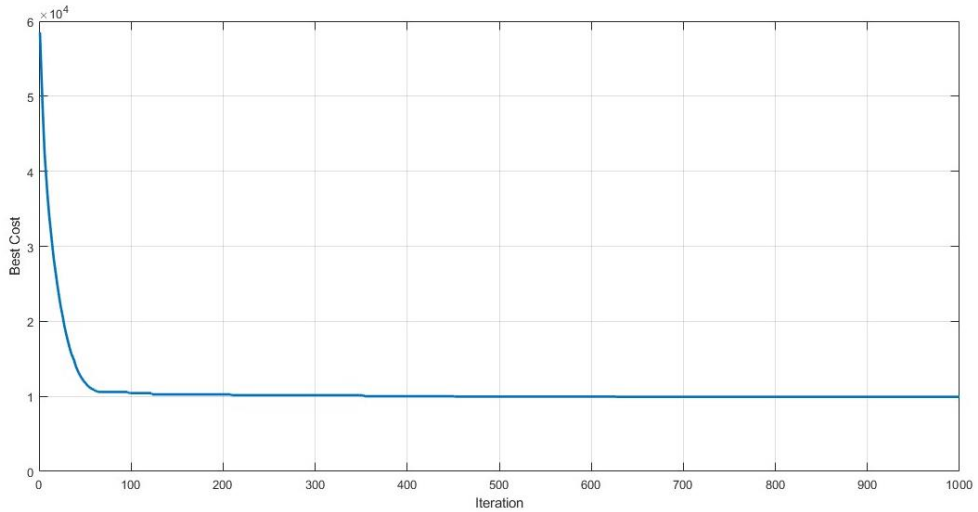

Figure 12. The best costs of 1000 iterations for TSA.

TSSIA results:

Table 11 shows the parameters utilized in the implementation of the TSSIA method. These TSSIA implementations have been executed 20 times with seed values ranging from 1 to 20. As can be seen, the implementations have been managed using the same parameters as the TSA method. However, due to SI module, it includes additional parameters such as segment length, total number of segments, and maximum number of segments.

Table 11. The parameters for TSSIA.

TSSIA Parameters	Descriptions	Setting Values
Attempts	The number of attempts for random seed=1,2,…,20	20
Max Iterations	Stopping condition on loop	1000 (500, 750, 1000)
Initial solutions	How to produce initial solutions	Random seed (1, 2,…, 20)
Segment Length	The length of segments for improving the solution	19 (5-20)
num_segments	Total number of segments for improvements	63 (*n-segment length* +1)
num_seg_insertions	The number of max number insertions in segments	300 (50, 100, 200, 300)
Criteria of aspiration	The condition of overriding tabu status	Improving the best solution
n	The number of city	81
n_swap	The number of swapping	3240 ($n(n-1)/2$=3240, $n(n-1)/3$=2160, $n(n-1)/4$=1620))
n_reversion	The number of reversion	3240 ($n(n-1)/2$=3240, $n(n-1)/3$=2160, $n(n-1)/4$=1620))
n_insertionn	The number of insertions	6561(n^2= 6561, $n^2/2$= 3280 $n^2/3$= 2187)
n_action	The number of actions (Neighborhoods)	13041 (13041, 9760, 5427) (*n_swap* + *n_reversion* + *n_insertion*)
TL	Tabu length	1304 (1304 , 976, 542)(*n_action*/10)

As shown in Table 12, the province codes and names for the shortest tour obtained by the system created with the TSSIA are given in sequential order. It can be seen that this tour starts from Sakarya, proceeds finally to Düzce, and from there returns to Sakarya.

This developed system has been run 20 times, and the shortest distance and elapsed time for each trial are given in Table 13 and Figure 14. It is observed here that the shortest distance is 9906 km from the 9th trial, with an elapsed time of 140 seconds.

Figure 13 visually displays the drawing of the shortest tour on the map of Turkish provinces by the TSSIA system. This system attempts to find the most successful tour over 1000 iterations. The shortest distances obtained at the end of each iteration are presented in the graph in Figure 15.

Table 12. The best tour of province codes and names for TSSIA

Codes of the best tour:
54, 41, 34, 59, 39, 22, 17, 10, 16, 77, 11, 26, 43, 3, 64, 45, 35, 9, 48, 20, 32, 15, 7, 42, 70, 33, 1, 80, 31, 79, 27, 46, 2, 63, 47, 21, 72, 56, 73, 30, 65, 13, 49, 12, 23, 44, 62, 24, 29, 69, 25, 4, 76, 36, 75, 8, 53, 61, 28, 52, 55, 57, 19, 5, 60, 58, 38, 50, 51, 68, 40, 66, 71, 6, 18, 37, 78, 74, 67, 14, 81
Names of the best tour:
Sakarya-Kocaeli-İstanbul-Tekirdağ-Kırklareli-Edirne-Çanakkale-Balıkesir-Bursa-Yalova-Bilecik-Eskişehir-Kütahya-Afyonkarahisar-Uşak-Manisa-İzmir-Aydın-Muğla-Denizli-Isparta-Burdur-Antalya-Konya-Karaman-Mersin-Adana-Osmaniye-Hatay-Kilis-Gaziantep-Kahramanmaraş-Adıyaman-Şanlıurfa-Mardin-Diyarbakır-Batman-Siirt-Şırnak-Hakkari-Van-Bitlis-Muş-Bingöl-Elazığ-Malatya-Tunceli-Erzincan-Gümüşhane-Bayburt-Erzurum-Ağrı-Iğdır-Kars-Ardahan-Artvin-Rize-Trabzon-Giresun-Ordu-Samsun-Sinop-Çorum-Amasya-Tokat-Sivas-Kayseri-Nevşehir-Niğde-Aksaray-Kırşehir-Yozgat-Kırıkkale-Ankara-Çankırı-Kastamonu-Karabük-Bartın-Zonguldak-Bolu-Düzce

Table 13. The best tour distances and elapsed time for TSSIA.

Attempts	**Best Tour Distances (Kilometers)**	**Elapsed Time (Seconds)**
1	10119	137.2
2	10099	138.1
3	9935	141.6
4	10093	138.1
5	10000	137.5
6	10180	136.8
7	9993	137.5
8	10041	140.2
9	**9906**	140.0
10	9960	138.6
11	10129	140.9
12	10125	140.1
13	10128	140.6
14	10015	141.1
15	10070	140.5
16	9961	140.8
17	10072	140.6
18	10049	140.2
19	9993	140.8
20	9958	140.2
Min	**9906**	136.8
Max	10180	141.6
Mean	10041.3	139.6
Std	74.8	1.47

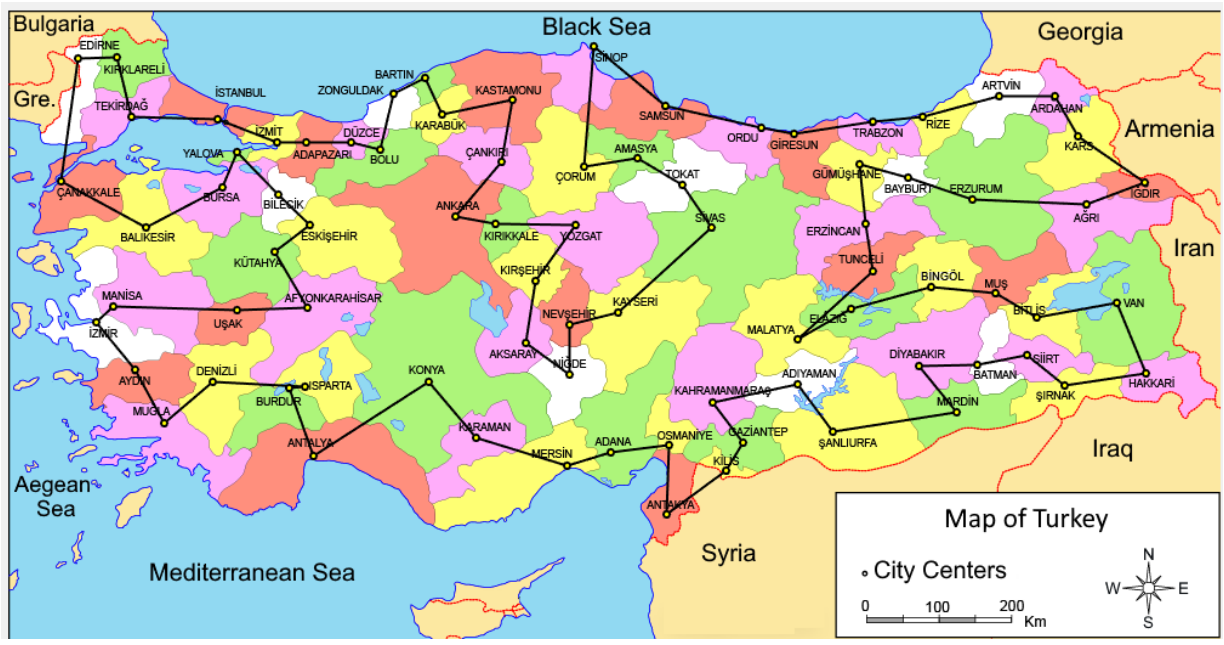

Figure 13. The best tour drawn on Türkiye province map for TSSIA [49].

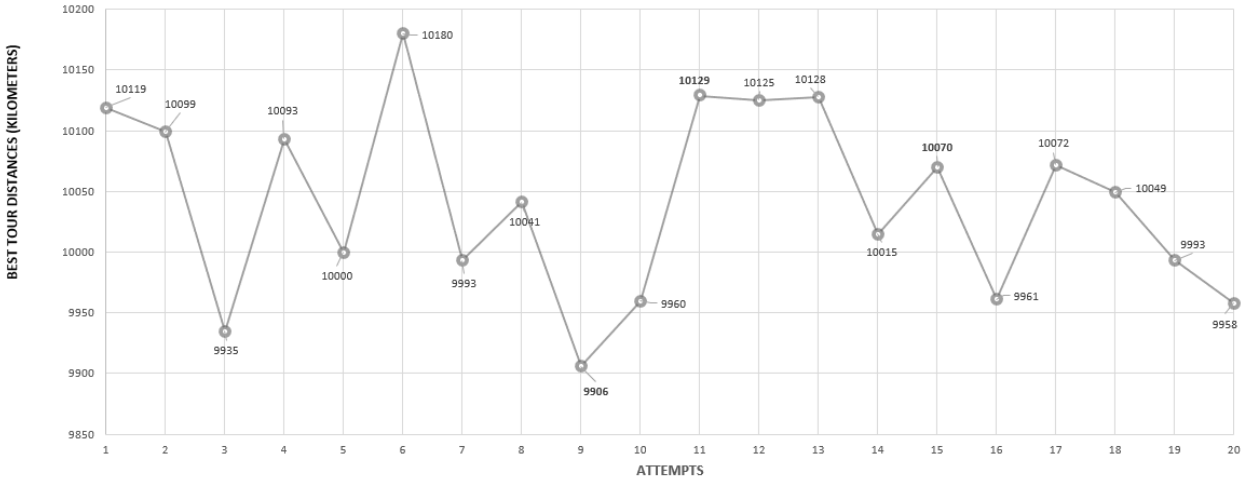

Figure 14. The tour distances of twenty attempts for TSSIA.

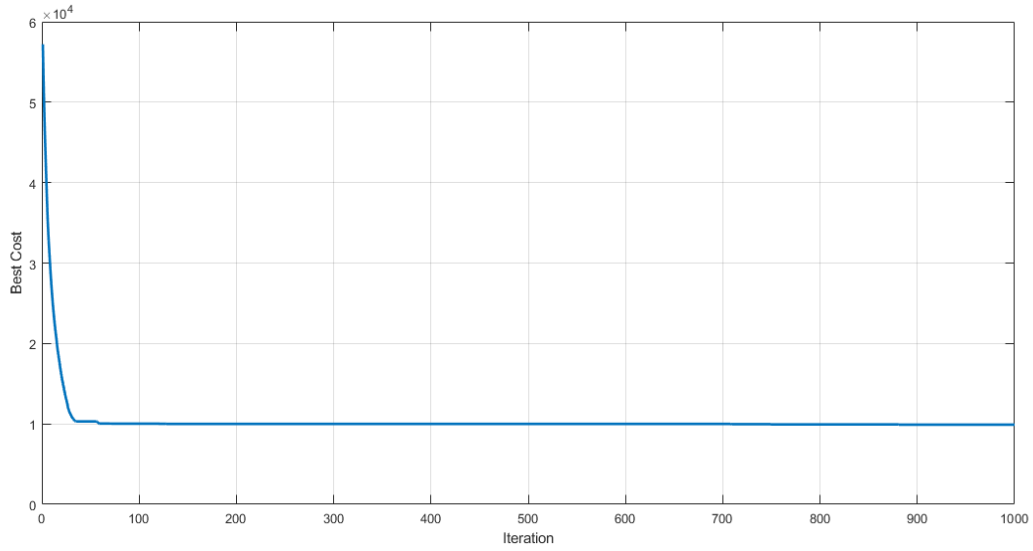

Figure 15. The best costs of 1000 iterations for TSSIA.

As can be seen in Table 14, the most successful tours and algorithms from studies conducted to date on the TSP for Turkish provinces are presented. Some of the studies here created systems by considering only specific provinces. Other studies, however, were conducted covering all provinces (81 provinces) of Türkiye. Looking at the results of the studies using 81 provinces, the most successful tour from studies conducted so far was found to be 9966 km with the ACO. However, in this study, the most successful tour was obtained with 9906 km using the TSSIA.

Studies [13] and [14] focused on finding the best tours for 20 provinces using ACO and 21 provinces using Genetic Algorithm, respectively. Study [13] stated a best tour of 2529, whereas Study [14] reported 4034. These results show that ACO outperformed the Genetic Algorithm in these specific cases. Nevertheless, as the current study involves 81 provinces, a direct comparison with Studies [13] and [14] would not be accurate due to the significant difference in city size.

While bio-inspired methods like ABCA and ACO rely heavily on stochastic processes and probabilistic transitions, TSA utilizes a deterministic approach supported by a short-term memory structure (Tabu list). This allows TSA to escape local optima more efficiently by explicitly forbidding recent moves, ensuring a more diverse exploration of the 81-province route without relying on random chance.

Table 14. The best tours and methods of the studies for Turkish cities

Studies	Methods	Best Tours
[4]	TSA (81 provinces)	10121
[12]	Imperialist Competitive Algorithm (81 provinces)	10394
[13]	ACO (20 provinces)	2529
[14]	Genetic Algorithm (21 provinces)	4034
[2]	Population-Based SA Algorithm (81 provinces)	9998
[5]	ACO (81 provinces)	9966
This study	**TSA (81 provinces)**	**9936**
This study	**TSSIA (81 provinces)**	**9906**

In Table 15, the statistical values (minimum, maximum, average, and standard deviation) for the best tour and elapsed times obtained from 20 runs of the ACO, ABCA, TSA, and TSSIA methods have been computed. The values marked in bold in the table indicate the most successful results. Accordingly, it is clearly evident that TSSIA is the method achieving the best tour, with minimum, maximum, average, and standard deviation values of 9906, 10180, 10041.3, and 74.8, respectively. The ABCA method obtained the worst tour with 10456. Furthermore, observing the table, the TSA method is the fastest with an average of 51.6 seconds, whereas the ACO method is the slowest with an average of 151.1 seconds.

Table 15. The statistical results for ACO, ABCA, TSA and TSSIA.

Methods	Tours and Time	Min	Max	Mean	Std
ACO	Best Tour Distances (km)	10193	10491	10366.6	86.1
	Elapsed Time (Seconds)	149.3	152.9	151.1	1.05
ABCA	Best Tour Distances (km)	10456	11556	10970	302.8
	Elapsed Time (Seconds)	112	118.1	114	1.74
TSA	Best Tour Distances (km)	9936	10225	10063.8	90.5
	Elapsed Time (Seconds)	**51.1**	**52.2**	**51.6**	**0.4**
TSSIA	Best Tour Distances (km)	**9906**	**10180**	**10041.3**	**74.8**
	Elapsed Time (Seconds)	136.8	141.6	139.6	1.47

While bio-inspired methods like ABCA and ACO rely heavily on stochastic processes and probabilistic transitions, TSA utilizes a deterministic approach supported by a short-term memory structure (Tabu list). This allows TSA to escape local optima more efficiently by explicitly forbidding recent moves, ensuring a more diverse exploration of the 81-province route without relying on random chance.

The logic of developing the Segment Improvement (SI) technique relies on this: it has been observed in studies that optimization algorithms create erroneous routes in mutually close regions even in many of the best solutions (best routes) obtained. In order to minimize these obviously erroneous solutions, at each iteration, it takes every segment of a certain length from the best solution (when the segment length is 19 in the best solution of this study, 63 segments are formed for 81 cities) and detects the incorrect orderings in this segment. Thus, SI corrects the incorrect orderings to obtain a better solution in regions close to every city of the best solution at each iteration.

For example, a 6-city segment of the best solution in the ACO is as follows. The length of this segment is 708 km, as seen below:

"…-Osmaniye-Kahramanmaraş-Gaziantep-Kilis-Hatay-Adıyaman-…", Segment Cost(105-78-63-146-316)=708 km

If segment optimization had been applied here, the probability of finding the solution would have been higher by focusing on specific segments, and the 636 km segment below would likely have been found as part of the solution. The following segment was found with TSSIA. Thus, a significant improvement of 72 km was achieved.

4. Conclusion

In this study, the metaheuristic systems have been developed for the Traveling Salesman Problem of Türkiye's 81 provinces and the solutions of the systems have been produced. ACO, ABCA, and TSA have been used in the construction of these systems. Of these algorithms, the most successful was the TSA. Furthermore, the most successful result has been achieved with the TSSIA, which was designed by making an improvement to the TSA. While the most successful method from studies to date was the ACO with 9966 km, with this study the shortest tours were identified as 9936 km with the Standard TSA and 9906 km with the developed TSSIA.

In future studies, new systems will be designed and implemented with different metaheuristic algorithms and with algorithms that will be formed by making improvements on them in order to obtain the best results.

Declaration of Ethical Code

In this study, I undertake that all the rules required to be followed within the scope of the "Higher Education Institutions Scientific Research and Publication Ethics Directive" are complied with, and that none of the actions stated under the heading "Actions Against Scientific Research and Publication Ethics" are not carried out.

References

[1] Goyal, S. 2010. A survey on travelling salesman problem. 43rd Midwest Instruction and Computing Symposium, Eau Claire, Wisconsin, USA, 16-17.

[2] Aşlıyan, R. 2022. Traveling Salesman Problem Solution with Ant Colony Optimization and

Simulated Annealing: A Case Study for Turkish Provinces, 2nd International Symposium of Scientific Research and Innovative Studies (ISSRIS'22), 2-5 March, 1-10, 583-592.

[3] Matai, R., Singh, S., Mittal, M.L. 2010. Traveling Salesman Problem: an Overview of Applications, Formulations and Solution Approaches. Editor: Davendra D. Traveling salesman problem, theory and applications, Landon, United Kingdom, InTech, 1-24.

[4] Aşlıyan, R. 2021. Comparison of Traveling Salesman Problem with Metaheuristic Algorithms for Turkish Provinces. International Congress on Scientific Advances (ICONSAD'21), 22-25 December, Türkiye, 660-668.

[5] Dikmen, H., Dikmen, H., Elbir, A., Ekşi, Z. 2014. Gezgin Satıcı Probleminin Karınca Kolonisi ve Genetik Algoritmalarla Eniyilemesi ve Karşılaştırılması. Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 18(1), 8-13.

[6] Glover, F., Laguna, M. 1997. Tabu Search. Kluwer Academic Publishers.

[7] Gendreau, M., Potvin, J. Y. (Eds.). 2010. Handbook of Metaheuristics. Second edition. Springer.

[8] Dantzig, G. B., Fulkerson, D. R., Johnson, S. M. 1954. Solution of a large-scale traveling-salesman problem. Operations Research, 2(4), 393-410.

[9] Miller, C. E., Tucker, A. W., Zemlin, R. A. 1960. Integer Programming Formulation of Traveling Salesman Problems. Journal of the ACM, 7(4), 326-329.

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

[11] Padberg, M., Sung, T. Y. 1991. An analytical comparison of different formulations of the traveling salesman problem. Mathematical Programming, 52(1-3), 315-357.

[12] Köse, O., Erdoğmuş, P. 2019. K-Gezgin Satıcı Probleminin Emperyalist Rekabetçi Algoritması ile Kümeleme Tabanlı Optimizasyonu. Bartın Üniversitesi Mühendislik ve Teknoloji Bilimleri Dergisi, 7(1), 1-13.

[13] Şenaras, E. A., İnanç, Ş. 2017. GSP Çözümü için Karınca Kolonisi Optimizasyonu. Sosyal Bilimler Metinleri, 58-67.

[14] Ertuğrul, İ., Özçil, A. 2016. Siyasi parti mitinglerinin gezgin satıcı problemi yaklaşımı ile analizi. Siyaset, Ekonomi ve Yönetim Araştırmaları Dergisi, 4(4), 223-238.

[15] Hasan, D. O., Aladdin, A. M. 2025. Real-World Applications of Metaheuristic Algorithms: A Comprehensive Review of the State-of-the-Art. Diyala Journal of Engineering Sciences, 1-27.

[16] Almufti, S. M. 2025. Metaheuristics Algorithms: Overview, Applications, and Modifications.

[17] Rashid, N. S., Zebari, I. M. I. 2025. A comprehensive review of metaheuristic algorithms for combinatorial optimization problems. International Journal of Scientific World, 11(1), 83-92.

[18] Pop, P. C., Cosma, O., Sabo, C., Pop Sitar, C. 2024. A comprehensive survey on the generalized traveling salesman problem. European Journal of Operational Research, 314(3), 819-835.

[19] Soni, V., Sharma, A., Singh, V. 2021. A Critical Review on Nature Inspired Optimization Algorithms. 1099(1), 012055.

[20] Liu, Y., Chen, X., Dib, O. 2023. Application of Metaheuristic Algorithms and Their Combinations to Travelling Salesman Problem. Springer International Publishing, 3-18.

[21] Jiang, C., Wan, Z., Peng, Z. 2020. A new efficient hybrid algorithm for large scale multiple traveling salesman problems. Expert Systems with Applications, 139, 112867.

[22] Yahia, W. B., Al-Neama, M. W., Arif, G. E. 2020. A hybrid optimization algorithm of ant colony search and neighbour-joining method to solve the travelling salesman problem. Advanced Mathematical Models & Applications, 5(1).

[23] Qamar, M. S., Tu, S., Ali, F., Armghan, A., Munir, M. F., Alenezi, F., Alnaim, N. 2021. Improvement of traveling salesman problem solution using hybrid algorithm based on best-worst ant system and particle swarm optimization. Applied Sciences, 11(11), 4780.

[24] Kanna, S. R., Sivakumar, K., Lingaraj, N. 2021. Development of deer hunting linked earthworm optimization algorithm for solving large scale traveling salesman problem. Knowledge-Based Systems, 227, 107199.

[25] Sarucan, A., Berkaya, M. F. 2024. Combining 3-Opt and Improved Discrete Cuckoo Search Algorithm for the Traveling Salesman Problem. Discrete Dynamics in Nature and Society, 2024(1).

[26] Bai, Z., Snášel, V., Vo, B., Kong, L., Wang, X. 2024. A Hybrid Algorithm ACS-GA for Solving the Traveling Salesman Problem. Springer Science+Business Media, 71-82.

[27] Yang, L., Wang, X., He, Z., Wang, S. Lin, J. 2024. Review of Traveling Salesman Problem Solution Methods. Communications in computer and information science, 3-16.

[28] Murali Krishna, M., Panda, N., Majhi, S. K. 2021. Solving traveling salesman problem using

hybridization of rider optimization and spotted hyena optimization algorithm. Expert Systems With Applications, 183, 115353.

[29] Thongpiem, J., Punkong, N., Ratanavilisagul C., Kosolsombat, S. 2024. Hybrid Genetic Algorithm and Ant Colony Algorithm for Solving Travelling Salesman Problem. IEEE 9th International Conference on Computational Intelligence and Applications (ICCIA), Haikou, China, 2024, 69-73.

[30] Kothari, A., Nikhil, V. R. N. S., Rohit, N., Sahay, A. 2024. Harnessing Meta-Heuristic Algorithms to Optimize Travelling Salesman Problem. 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kamand, India, 1-9.

[31] Abid, M., El Kafhali, S., Amzil, A., Hanini, M. 2023. An Efficient Meta-Heuristic Methods for Travelling Salesman Problem. In: Hassanien, A.E., et al. The 3rd International Conference on Artificial Intelligence and Computer Vision (AICV2023), Lecture Notes on Data Engineering and Communications Technologies, Springer, vol. 164.

[32] Glover, F. 1986. Future Paths for Integer Programming and Links to Artificial Intelligence. Computers & Operations Research, 13(5), 533-549.

[33] Glover, F. 1989. Tabu Search-Part I. ORSA Journal on Computing, 1(3), 190-206.

[34] Glover, F. 1990. Tabu Search-Part II. ORSA Journal on Computing, 2(1), 4-32.

[35] Malek, M., Guruswamy, M., Pandya, M., Owens, H. 1989. Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. Annals of Operations Research, 21(1), 59-84.

[36] Wu, Y., Zhou, X. 2008. Meliorative Tabu Search Algorithm for TSP Problem. Journal of Computer Engineering and Applications, 44(1), 57-59.

[37] Gendreau, M., Laporte, G., Semet, F. 1998. A tabu search heuristic for the undirected selective travelling salesman problem. European Journal of Operational Research, 106(2-3), 539-545.

[38] Dorigo, M., Gambardella, L. M. 1997. Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, 1(1), 53-66.

[39] Dorigo, M., Maniezzo, V., Colorni, A. 1996. Ant System: Optimization by a colony of cooperating agents. IEEE Trans Syst, Man Cybernet Part B, 26(1), 29-41.

[40] Dorigo, M., Stützle, T. 2004. Ant Colony optimization. Cambridge, MA, MIT Press, 305s.

[41] Dorigo, M., Gambardella, L. M. 1997. Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, 1(1), 53-66.

[42] Shang, G. 2005. The chaotic ant colony algorithm to solve TSP problem. Journal of Systems Engineering-theory & Practice, 9,100.

[43] Akça, M. R. 2011. Yapay arı koloni algoritması kullanılarak gezgin satıcı probleminin Türkiye'deki il ve ilçe merkezlerine uygulanması. Master thesis, Institute of Natural and Applied Science, Konya, Turkey.

[44] Karaboğa, D. 2005. An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.

[45] Karaboğa D., Baştürk, B. 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Global. Optim, 39, 459-471.

[46] Karaboğa, D., Baştürk Akay, B., Öztürk, C. 2007. Artificial Bee Colony (ABC) Optimization Algorithm for Training Feed-Forward Neural Networks. In: LNCS: Modeling Decisions for Artificial Intelligence, Springer-Verlag, Berlin Heidelberg 4617/2007, 318-319.

[47] Kalınlı, A., Karaboğa, N., Karaboğa D. 2001. A Modified Touring Ant Colony Optimization Algorithm for Contounous Functions. 16th International Symposium on Computer and Informatıon Science (ISCIS XVI), Işık University, Antalya, 437-44.

[48] Türkiye Karayolları Genel Müdürlüğü. https://www.kgm.gov.tr/Sayfalar/KGM/SiteTr/Root/Uzakliklar.aspx (Erişim Tarihi: 20.06.2025).

[49] Cografya Harita. http://cografyaharita.com. (Acces Date: 17.06.2025).