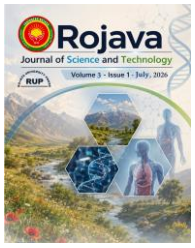
Rojava Journal of Science and Technology

2026, VOL. 3, No 1, [121-136]



Solving the Traveling Salesman Problem using an improved ant colony algorithm based on the nearest neighbor algorithm

Mohammed Salim Hassan[a], Yasmine Abouelseoud*[b] and Ahmed F. Tayel[b]

[a] Department of Mathematics, Institute of Science and Modern Technology, University of Rojava, Syria

[b] Department of Engineering Mathematics and Physics, Faculty of Engineering, Alexandria University, Egypt

Abstract

In this paper, we address the Traveling Salesman Problem (TSP), which is an NP-hard problem. Many meta-methods have been applied to solve such a problem in the literature. In this work, the Ant Colony Optimization (ACO) algorithm is applied to TSP, which is one of the population-based competitive algorithms used to solve many generative optimization problems in various engineering fields. By optimizing its problem-dependent parameters setting, the overall performance of the Ant Colony algorithm is improved, and the execution time is reduced. We have found through experiments that the performance of the ACO algorithm needs to be enhanced. Therefore, we have developed a new algorithm (NN-ACO) by hybridizing the ACO algorithm with the nearest neighbor algorithm, which is a greedy algorithm that finds the candidate solution to the TSP using simple approximate heuristics. The results of the ACO algorithm and the hybrid ACO algorithm are compared together in terms of solution quality and execution time. Moreover, the result based on the linear programming formulation of the test problem is used to evaluate the quality of the solutions obtained using the applied meta-heuristic methods. In addition, the algorithm is applied to a benchmark set of TSP problems and its performance is compared with some other hybrid algorithms in the literature. The experimental results show that the proposed algorithm outperforms other hybrid algorithms in terms of solution quality.

ARTICLE HISTORY

Received 12 Jan. 2026

Revised 3 Jun. 2026

Accepted 4 Sep. 2026

Keywords

Ant Colony Optimization (ACO), Parameters Tuning, Linear Programming (LP), Nearest Neighbor (NN) algorithm, Route Optimization, Traveling Salesman Problem (TSP).

*Corresponding Corresponding Author yasmine.abouelseoud@alexu.edu.eg,

1. Introduction

The Traveling Salesman Problem (TSP) is an NP-hard problem, which means that there is no known polynomial-time solution technique that can provide an exact solution to a given problem instance. TSP can be envisioned as an abstraction of problems encountered in the fields of logistics, computer science, controlling industrial robots, drilling holes in electrical circuits, etc. It aims to achieve a trip to all cities - visiting each only once - before returning to the city from which it started with the least cost. The cost can be the minimum distance, or the minimum travel time, or any other cost function dependent on the problem context. A solution of the TSP is simply a permutation of the cities. So, the number of possible solutions to a TSP instance is **n!**, where n is the number of cities. When n is large, it is difficult to find the minimum cost solution in polynomial time. Therefore, we resort to other methods to find approximate solutions to such problems that give near optimal solutions. These approximate techniques are either based on Problem-based heuristics such as the nearest neighbor algorithm [1-2] or nature-inspired heuristics that serve as general frameworks to guide the search procedure in the solution space. Examples of the latter class of algorithms include genetic algorithm (GA) [3-6], ant colony algorithm (ACO) [7-9], neural networks [10-11], and Artificial Bee Colony (ABC) algorithm [12]. Moreover, other solution techniques have been invented, such as combining two algorithms which are referred to as hybrid algorithms in order to find effective solutions to problems that are difficult to solve using traditional methods. For example, in [13], the authors proposed hybridization of the genetic algorithm, the simulated annealing algorithm, and ant colony optimization scheme as well as particle swarm optimization technique. According to the conducted study using this hybrid algorithm, experimental results obtained for the TSPLIB (a library for TSP test instances) showed that both the mean solution and the deviation ratio of the mean solution are improved compared to other methods in literature. The use of self-organizing maps to solve TSP has been considered by several researchers [14-15]. Masotti and de Castro considered enhancing self-organizing maps performance for solving TSP based on ideas from the immune system [16], extending the work in [17]. Durbin and Willshaw proposed a solution to TSP based on elastic net involving time-dependent parameters that aid in moving neurons more rapidly towards cities positions and attracting neurons to their neighbors on the path in order to minimize the total path length [18].

Dong et al. in [19] presented a Cooperative Genetic Ant System (CGAS) for solving TSPs. In order to improve the performance of the basic ACO algorithm, a genetic algorithm (GA) is combined with it to obtain good solutions within an acceptable time period. According to the study, it was found that CGAS has superior performance over both GA and ACO algorithms in solving TSP in terms of its ability to find the global optimal solution. Yannis Marinakis and Magdalene Marinakis in [20] presented a hybrid algorithm that combines a genetic algorithm and particle swarm optimization to solve another related combinatorial optimization problem, which is the vehicle routing problem. There was an improvement in the exploration capabilities of their hybrid algorithm, thus, yielding better solutions to the problem instances considered in their study. In [21], a hybrid algorithm to solve the TSP problem that combines deer hunting optimization algorithm (DHOA) and earthworm algorithm (EWA) has been presented. The proposed (EW-DHOA) algorithm aims to find the optimal solution of large-scale problems. Experimental results showed that the convergence of the proposed hybrid optimization was better in finding optimal solutions compared to the performance of both DHOA and EWA, when applied separately. The authors in [22] presented a modification of the artificial bee colony algorithm. Using multiple update rules and the K-opt process, the algorithm generally revives stagnant solutions. The 3-opt process is used to optimize any stagnant solution obtained in the sequel. All experimental results indicated that the performance of the proposed approach is superior compared to other existing methods in the literature in terms of accuracy, efficiency and consistency.

The authors in [23] introduced a particle swarm optimization (PSO) algorithm to improve the performance of the ACO algorithm and called their new hybrid optimization algorithm (HPSACO), which

was used to solve the TSP problem. The HPSACO algorithm takes advantage of the exploratory ability of the PSO algorithm and the stochastic ability of the ACO algorithm. The experimental results showed that their proposed HPSACO algorithm can obtain the best solution quality and better stability compared to the other researches included in their study. In [24], a hybrid algorithm to solve the traveling salesman problem, that combines a heuristic algorithm and a repetitive nearest neighbor (RNN) mechanism, was presented. The proposed algorithm consists of two main stages. In the first stage, the RNN algorithm is used to build a set of possible paths step by step. In the second stage, it improves these paths in an iterative optimization process based on the simulated annealing (SA) algorithm. Experimental results obtained for TSP instances taken from the TSPLIB library showed that the proposed algorithm (RNN-SA) is more effective than both RNN and SA. The authors in [25] proposed a neural network to dynamically adapt the two parameters of the ACO algorithm (α and β). Determining the best combination of ACO parameters has a critical impact on the performance of the algorithm. The authors in [26] proposed an evolutionary hybrid approach, combining the ant colony algorithm and attention-based neural networks (ACO-RNN), to address the traveling salesman problem. Results on TSP instances from the TSPLIB [27] library showed that the hybrid approach outperformed the classical ACO algorithm.

In our research, linear programming is used to obtain the exact optimal solution to moderate size TSP problems [28–29]. This is important in evaluating the performance of the studied approximate algorithms for solving the traveling salesman problem. Our study focuses on meta-heuristic algorithms for the traveling salesman problem, with a particular emphasis on the ant colony optimization (ACO) algorithm. The ACO algorithm is inspired from the natural searching behavior of ants, where pheromones play a crucial role in guiding other ants along short paths, a concept discussed in detail in Section 3. The above review of the literature suggests that hybrid algorithms tend to provide superior solutions. Therefore, we designed a hybrid algorithm (NN-ACO) that combines the ACO algorithm and the nearest neighbor (NN) algorithm to achieve efficient solutions within an acceptable time frame. The effectiveness of this hybrid approach is demonstrated by the results presented in Section 5. The ACO algorithm has been extensively analyzed and its parameters have been carefully tuned in order to improve its performance and adapt it to solve the TSP problem with different number of cities. Our research highlights the potential of ACO algorithm to solve the travelling problem, which in turn plays an important role in several practical planning problems. These practical problems include logistic planning, road planning, printed circuit board wiring [30] and urban design. Moreover, the efficiency of public transportation can be improved reducing congestion and pollution based on modelling the planning problem as a TSP.

The contributions of this research can be summarized as follows:

1- Tuning the parameter settings of the ACO algorithm, which depend on the specific problem instance to be solved.
2- Building a new hybrid algorithm (NN-ACO) that effectively combines the greedy nearest neighbor algorithm and the ACO algorithm.
3- Reaching near optimal solutions within a small number of iterations and thus reducing the computational time.

4- Comparing the results of the ACO and NN algorithms with the hybrid NN-ACO algorithm, in addition to comparing the NN-ACO algorithm with another recent hybrid algorithm.

The rest of the paper is organized as follows. Section 2 provides a description of the travelling salesman problem. Section 3 describes the algorithms used in this work, namely linear programming, nearest neighbor algorithm, and ACO algorithm. Section 4 introduces the proposed hybrid algorithm. The fifth section presents the experimental results supported by tables and graphs. Section 6 presents the concluding remarks.

2. Problem description

TSP is an NP-hard problem that can be described as follows. Given n cities and d_{ij} is the distance between cities i and j, the goal is to determine the shortest tour of the group of n cities so that it passes

by each city once and returns to the starting city. An instance of a TSP problem can be described as a weighted graph $G(V, E)$, where V represents the set of nodes (cities), and E represents the set of edges (paths between cities). It is weighted according to the distances between cities d_{ij}. For the ACO algorithm, the ants will move randomly through the graph, i.e., each ant starts its tour from a certain city, let it be i, to the next city, j, until all the cities are visited once. To ensure that a city is not visited twice by each ant, a blocked list is kept for this purpose. Choosing the next city depends on a certain probability affected by the amount of pheromone, called the transition probability $p_{ij}(t)$. This depends on the distribution of pheromone τ and visibility η, which is inversely proportional to the distance, in order to attract ants towards the nearest cities. Pheromones are chemicals secreted by ants in order to attract other ants and direct them to a food source. The amount of pheromone depends on the pathways emanating from them. Thus, both visibility and pheromone distribution play a major role in directing the ACO algorithm in order to find effective solutions to the traveling salesman problem.

3. Description of algorithms used in this work

In this section, the linear programming formulation of the travelling salesman problem used is described first. Next, the nearest neighbor algorithm and the basic ACO algorithm that are implemented to solve TSP in our experiments are described.

3.1. Linear programming

Linear programming is an efficient and accurate method to solve the traveling salesman problem. It depends on representing the problem directly using binary variables (0 or 1) and applying appropriate linear constraints to ensure the feasibility of the obtained solution. The importance of this approach lies in its ability to yield exact solutions to TSP instances of moderately large sizes efficiently, making it a popular and preferred technique in many computational and engineering applications.

There are several integer programming formulations for the TSP problem. They differ in the number of decision variables and constraints. These formulations generally use natural (city-to-city) modeling variables and focus on explicit constraints intended to prevent subtours from automated solutions, known as "natural" or "subtour elimination" formulations. There are two prominent formulations: the Miller–Tucker–Zemlin formulation (MTZ) [31] and the Dantzig–Fulkerson–Johnson (DFJ) formulation [32-33].

In order to formulate the LP for the TSP problem, we consider a set of cities indexed as 1, . . ., n, and denote the distance between cities i and j by $\boldsymbol{c_{ij}}$. The decision variables of the problem are defined by the binary variables

$$x_{ij} = \begin{cases} 1 & optimal\ path\ includes\ the\ edge\ from\ city\ i\ to\ j\ city \\ 0 & otherwise \end{cases} \quad (1)$$

The goal of a linear program is to minimize the length of the complete tour.

$$\boldsymbol{D(x)} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \sum_{\boldsymbol{i\neq j, j=1}}^{\boldsymbol{n}} \boldsymbol{c_{ij}\, x_{ij}} \quad (\mathbf{2})$$

The following set of constraints must be included in the formulation to ensure that each city has only one incoming edge and one outgoing edge:

$$\sum_{\boldsymbol{i=1, i\neq j}}^{\boldsymbol{n}} \boldsymbol{x_{ij}} = \mathbf{1} \ \ \forall \boldsymbol{j} \quad (\mathbf{3})$$

$$\sum_{\boldsymbol{j=1, i\neq j}}^{\boldsymbol{n}} \boldsymbol{x_{ij}} = \mathbf{1} \quad \forall \boldsymbol{i} \quad (\mathbf{4})$$

Equations (3) and (4) specify that each city must be visited and left exactly once. This formulation is invalid in the sense that the solution is not necessarily a complete tour as desired. Generally, the solution

consists of several sub-tours and must be completed. It is formulated such that subtours are prevented, and it is arguable that this universal requirement is what makes TSP a difficult problem. This can be done conventionally in two ways, proposed respectively by DFJ (Danzig, Fulkerson, & Johnson) and MTZ (Miller, Tucker, & Zemlin). In this research, the MTZ formula is adopted to solve the TSP problem. The number of constraints and variables are a polynomial function of the number of vertices (cities).

Miller–Tucker–Zemlin formulation This formulation excludes subtours. In addition to the variables x_{ij} as mentioned above, new variables u_i , where $i = 2, ..., n$, are used in the MTZ formulation, with n denoting the maximum number of nodes that can be visited by any salesman. The variable u_i keeps track of the order in which cities are visited. The interpretation of $u_i < u_j$ means that city i was visited before city j for a given tour.

Hence, the MTZ formulation of the TSP to exclude sub-tours adds the following set of constraints

$$u_i - u_j + 1 \leq (n-1)(1-x_{ij}) \quad 2 \leq i \neq j \leq n \qquad (5)$$

$$1 \leq u_i \leq n-1 \quad 2 \leq i \leq n \qquad (6)$$

The constraints in (5) ensure that if the salesman travels from i to j, then the position of node j is one more than that of node i. Together with the bounds in (6), they ensure that each non-depot (intermediate) node is in a unique position.

3.2. Nearest Neighbor

The nearest neighbor (NN) algorithm is a greedy algorithm used to solve TSP, which yields solutions that are close to the optimal solution. J. G. Skellam [34] introduced the first NN algorithm. Work on NN was continued by P.J. Clark and F.C. Evans [34]. A tour for the street vendor is created on the principle of choosing the move with the lowest remaining cost in the sequence. The vendor starts with a random city, visits the city closest to the current city, and adds it to the route. The process is repeated until all cities are visited, and upon completion, it returns to the starting city. The NN algorithm can be summarized as follows:

1) Randomly choose the starting city as the current city.

2) Search for the closest unvisited city to the current city, move to it, and assign it as the current city.

3) Repeat the process until all cities have been visited.

4) If all cities have been visited, return to the first city and complete the procedure.

5) Display the path and calculate the total length of the path.

Improvement to the NN Heuristic using 2-change heuristic

Once we have an initial tour using the NN algorithm, it is optimized using the "2-opt" algorithm. This algorithm performs exchanges between edges in the tour to reduce the total path length. These exchanges are done such that two pairs of edges are replaced with each other to optimize the path length. The edge with the largest cost or weight is chosen for replacement. Once the two exchanges are completed, the tour with the smallest length is chosen as the final solution to the TSP problem. This process is repeated several times (determined by the value of M) to increase the chances of finding the optimal or near-optimal solution.

In Figure 1, the total length of the initial tour is 30 units. The edges selection and replacement in the 2-opt exchange process are also shown in Figure 1. For example, v5-v4 with the largest edge weight 9 and v3-v2 with edge weight 5 are selected, and the modified tour after 2-opt exchange process with new edge weights is shown in Figure 2. They are replaced by v5-v2 with edge weight 3 and v4-v3 with edge weight 5, so the modified tour is v1-v3-v4-v6-v2-v5-v1; with a total tour length of 24, as shown in Figure 2. In our experiments, the impact of repeating the 2- opt exchange process on the obtained tour cost and computational time are demonstrated.

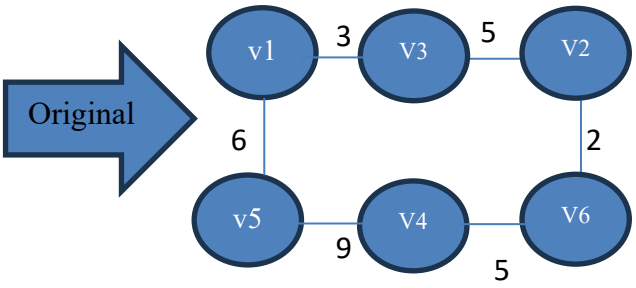


Figure 1: TSP network of 6 cities with their edge weights and original tour obtained using NN algorithm. Total tour weight: 24

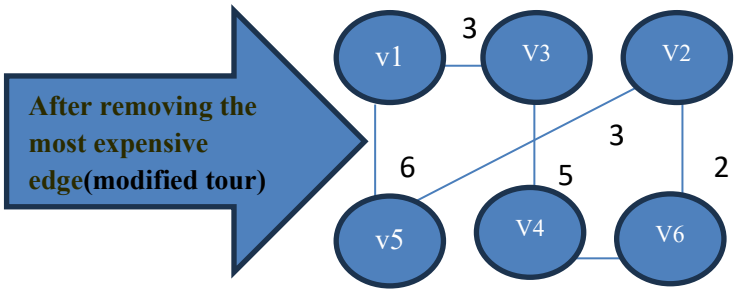


Figure 2: The 6-city TSP tour after exchanging a pair of its edges according to 2-opt exchange process.

3.3. Ant Colony Optimization

M. Dorige introduced Ant Colony Optimization (ACO) in 1991[35], which is a nature-inspired technique for solving difficult combinatorial problems represented as graphs. It is a population-based algorithm, where the population consists of a group of ants that by working together search for food sources and return to their nests. Pheromones (chemicals) secreted by ants and dissipated along the path visited in the graph play an important role in directing them to a food source. There are some important strategies, such as positive feedback, incorporated in ACO. The positive feedback strategy through use of pheromones helps ACO find better results because ants are attracted to paths that have an increase in pheromone concentration, and thus the pheromone distribution affects the discovery of areas that contain good solutions. However, the positive feedback technique alone is not sufficient and the algorithm may converge early, get stuck in local solutions, and stall without finding the optimal solution. To counter this effect, a negative feedback mechanism, namely evaporation, is also needed. Evaporation is an important feature of pheromones, which prevents excessive accumulation of pheromones on the edges in each cycle, avoiding getting trapped into local optimal solutions.

ACO is one of the most famous heuristic algorithms for solving the TSP problem, which is implemented by simulating ants on a graph. It starts with an initial amount of pheromones on each path between cities and this is followed by randomly distributing ants to different nodes. ACO involves two main steps after the initialization step. The first step is to construct solutions: each ant constructs its route based on a probability rule that depends on the current amount of pheromones and the distance between cities. The second step is to update the pheromones. When paths are built at iteration $\boldsymbol{t+1}$, each ant moves to a new node and the pheromones are updated. Shorter, more efficient paths receive larger amounts of pheromones. The pheromone is vaporized after ants complete their tours to prevent excessive accumulation and to explore other solution areas. When the ant decides to choose the next city, it does so with a probability that depends on the distance to that node and the intensity of the pheromone. The inverse of the distance to the next node is known as the visibility (visibility of city j from city i), and the visibility η_{ij} is defined as

$$\eta_{ij} = \frac{\mathbf{1}}{\boldsymbol{d_{ij}}} \quad (7)$$

where $\boldsymbol{d_{ij}}$ is the distance between nodes i and j.

3.2.1. Possibility of transfer

The probability of moving, $\boldsymbol{p_{ij}(t)}$, is the probability of choosing city j from city i at time t based on the pheromone distribution τ, visibility η, and permissible choices determined by previously visited nodes. $\boldsymbol{p_{ij}(t)}$ involves two parameters α and β. When α is large and β is small, the choice is more directed by pheromones, and in the opposite case, the choice is more directed by distance. The following formula is employed to

compute the probability of transition, where X is the set of unvisited cities for ant $\boldsymbol{k}$:

$$\boldsymbol{p_{ij}}(t) = \begin{cases} \frac{[\boldsymbol{\tau_{ij}}(t)]^{\propto} \cdot [\eta_{\boldsymbol{ij}}(t)]^{\beta}}{\sum_{k \epsilon X} [\boldsymbol{\tau_{ij}} t)]^{\propto} \cdot [\eta_{\boldsymbol{ij}}(t)]^{\beta}} & if \ \ k \in X \\ 0 & otherwise \end{cases} \quad (8)$$

α and β are two parameters that determine the relative influence of the pheromone and distance. With α = 0 and β = 1, you get a purely greedy algorithm, while if α = 1 and β = 0, you get a completely randomly controlled algorithm.

3.3.2 Pheromone trail update rule

First, ants are placed randomly at the $\boldsymbol{n}$ cities. Next, each ant $\boldsymbol{k}$ goes from city $\boldsymbol{i}$ to city $\boldsymbol{j}$ with the probability $\boldsymbol{p_{ij}}^{(k)}$, given by Eq. (8). Each ant completes its tour after $\boldsymbol{n}$ iterations. Moreover, each ant $\boldsymbol{k}$ leaves pheromone according to $\boldsymbol{Q}$ /$\boldsymbol{L_k}$, where $\boldsymbol{Q}$ is a constant and $\boldsymbol{L_k}$ is the length of its tour.
This way ants with shorter tours will leave more pheromone than ants with longer tours. Additionally, the pheromone evaporates with time as it will be indicated later by the evaporation factor $\boldsymbol{\rho}$.

$\boldsymbol{\tau_{ij}}(\boldsymbol{t})$ describes the pheromone density at time t on edge (i, j). The distribution of the pheromone is affected by the ants' previous path choices, and affects the future path choices. After each cycle of the algorithm, the pheromone density becomes:

$$\boldsymbol{\tau_{ij}}(\boldsymbol{t+1}) = \boldsymbol{\rho} * \boldsymbol{\tau_{ij}}(\boldsymbol{t}) + \boldsymbol{\Delta \tau_{ij}} \quad (9)$$

This formula applies to all edges of the graph with density $\boldsymbol{\rho}$ between zero and one.

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \quad (10)$$

where m is the number of ants and $\Delta\tau_{ij}^{k}$ is the amount of pheromone deposited by ant k onto the edge from i to j at the current iteration. ρ is known as the evaporation coefficient.

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & if \ ant \ \boldsymbol{k} \ used \ \ edge \ (i,j) \ in \ its \ tour \\ 0 & otherwise \end{cases} \quad (11)$$

Here, $\mathbf{Q}$ is a constant and $\boldsymbol{L_K}$is the length of the tour of the $\boldsymbol{k}$ th ant.

3.2.2. ACO Algorithm Description

Based on the aforementioned reference [36], the algorithm can be summarized as follows:

- Initialize the pheromone distribution $\boldsymbol{\tau}$
- Each ant is randomly placed in a city at the beginning of the algorithm
- Build trajectories and update pheromones up to a specified number of iterations $\boldsymbol{t_{max}}$
- Construct a path for each ant using calculated transition probabilities
- Calculate tour costs
- Update the best solution, if any
- Determine $\boldsymbol{\Delta\tau_K(t+1)}$
- Updated pheromone distribution based on new pathways
- Printing the best solution.

3.3.4. How is each city visited once on each route?

In the ant colony optimization (ACO) algorithm used to solve the traveling salesman problem, duplication of a city on each path chosen by the ant is avoided by modifying the visibility matrix at each iteration by setting the visibility of the visited city to zero, dynamically during the tour. The effect of this modification prevents the ant from choosing this city again because the visibility becomes non-existent, making the possibility of choosing it in the following probability calculations impossible.

4. The Proposed Hybrid ACO Algorithm

The TSP problem is one of the most famous problems studied in the field of combinatorial optimization. It has been proven to be an NP-hard problem, and no algorithm has been found to obtain the optimal solution in polynomial time with the increase in the number of cities. TSP plays an important role in modeling numerous real-life problems, thus, the need for an algorithm to solve large-scale TSP problems has emerged.

In our research, a new hybrid algorithm is developed that combines the ACO algorithm and the nearest neighbor algorithm. The latter has a role in improving the speed of convergence and effectiveness of the ACO algorithm towards the optimal solution.

Figure 3 shows the flowchart that combines ACO and NN and how they are combined to solve the TSP problem. First, the parameters are determined through experimentation. The initial population is built by generating Ngreedy solutions using NN and optimizing these nearest neighbor paths using 2-opt exchange process. The rest of the initial population consisting of m - Ngreedy solutions (where m represents the population size) is randomly generated to ensure careful exploration of the search space. After this step, in each iteration, probabilistic solutions are built. If the stopping condition is met, the best path is printed. However, if the stopping condition is not met, the algorithm updates its parameters based on the new information from the current solutions and repeats the iterative process.

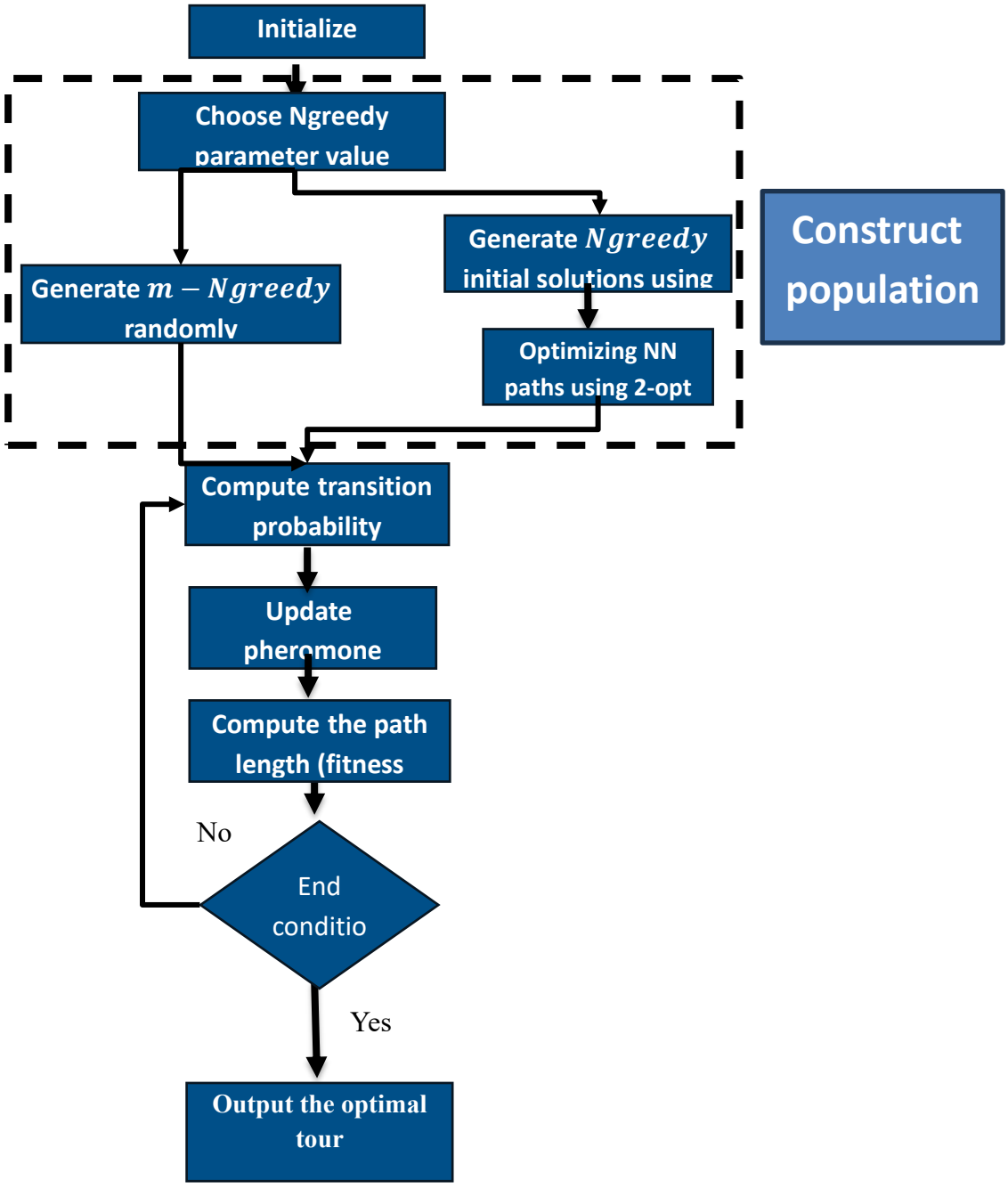


Figure 3: A flowchart combining the ACO and NN algorithms

5. Numerical Results

In this section, the performance of the proposed hybrid algorithm is evaluated and compared with the basic ACO algorithm and NN algorithms. The right set of parameters is determined by experimenting with problem instances of different sizes. This helped us identify the best parameter values that yield competitive results. The ACO parameters used in generating the results in this section are presented in Table 1.

Table 1: The values of the parameters included in the ACO algorithm.

parameters	α	B	evaporation coefficient (ρ)
Value	2	3	0.15

The results of the modification to the basic ACO algorithm show that using the nearest neighbor heuristic algorithm to generate part of the initial population improves the results in terms of reducing time consumed and quality of the obtained solutions, as shown in Tables 2 and 3, where Ngreedy = 3 and M is taken equal to 1 (recall that M represents the number of times the NN algorithm is executed to generate TSP tours, where each time the algorithm starts at a new random solution).

The performance of the hybrid algorithm and the performance of the ACO algorithm are evaluated by executing each algorithm 15 times to obtain the best solutions, the relative percentage error (%) is calculated using equation (12) to compare the performance of the two algorithms and their convergence.

$$Err_{ACO,NN-ACO}(\%) = \frac{Y_{ACO} - Y_{NN-ACO}}{Y_{NN-ACO}} \times 100\% \qquad (12)$$

Table 2 shows a comparison of four algorithms for a moderate number of cities ranging from 10 to 70. The four algorithms included in the comparison are the basic ACO algorithm, the proposed hybrid NN-ACO algorithm, the NN algorithm and the linear programming based approach (LP). The number of ants was set to m = 500 ants. The distance values for up to 30 cities are identical for the four algorithms. However, when the number of cities exceeds 30, slight differences start to occur. The results show that the proposed hybrid NN-ACO offers an improvement in performance through the effective role played by NN compared to the original ACO algorithm as the number of cities increases.

Figures 4 and 5 show convergence curves of the NN-ACO algorithm and ACO algorithms for a large-scale problem instance with 220 cities, respectively. In both figures, the first point at which an algorithm reaches a stable solution is highlighted and designated by X (number of iterations) and Y (corresponding cost or path length). These values (X and Y) are reported in Table 3.

In Table 3, as the number of cities increases ranging from 100 to 220, the performance difference between the ACO and the hybrid algorithm becomes more pronounced, with the relative percentage difference growing from 7.6954% for 100 cities to 14.2915% for 220 cities. For this larger size problem instances, the number of ants was set to be five times the number of cities to ensure diversity in solutions and a small number of iterations is used. The maximum number of iterations was set to be 20, after monitoring the convergence curves of the algorithms, as depicted in Figures 4 and 5. The proposed hybrid approach helps to improve the initial population generated and thus ACO builds solutions on it and thus skips situations in which the algorithm may fail, as shown in the results presented in Table 3 and Figure 4.

In the proposed hybrid algorithm, it is noteworthy that the greedy algorithm only participates in the initialization step. The remaining iterations are purely ACO procedures. This makes the complexity of each iteration of the proposed algorithm the same as that of the ACO algorithm alone. In this way, the number of iterations until a stable performance (X value) is reached can be considered as a measure of the time consumed by the algorithms, as shown in Tables 3 and 4. In Table 3, the cost (Y) achieved by the proposed hybrid algorithm is much lower than that of the standalone ACO algorithm for approximately the same number of iterations. Moreover, as the number of cities increases, the gap between the two algorithms. in terms of cost continues to widen. In addition, the

quality of the produced solution by the hybrid algorithm closely approximates the exact solution obtained by the LP-based approach for all tested problem instances, demonstrating the efficiency of the hybrid algorithm in reaching near-optimal solutions while maintaining a computational complexity similar to that of the basic ACO algorithm.

Table 2: *A comparison between the proposed algorithm and other algorithms for solving TSP*

Iteration = 500, m = 500				
N. Cites	ACO Ngreedy=0	NN-ACO Ngreedy=3	NN	LP
	Distance	distance	Distance	distance
10	29.5451	29.5451	29.5451	29.5451
20	35.3447	35.3447	35.3447	35.3447
30	45.5921	45.5921	45.5921	45.5921
40	49.8462	49.8314	50.3574	49.8314
50	56.7955	56.6001	57.1995	56.5524
60	65.7239	63.9752	63.9752	63.9752
70	68.9364	66.9294	67.0011	66.7672

Table 3: *The minimum distance obtained by ACO and NN-ACO and the least number of iterations required to converge to it*

Maximum Number of Iterations = 20							
No. of Cites	m	ACO Ngreedy=0		NN-ACO Ngreedy=3		NN	LP
		X	Y	X	Y	Y	Distance
100	500	4	85.6724	2	79.5506	80.5414	77.9675
150	750	3	105.298	5	94.6435	94.9129	93.5338
160	800	3	114.484	4	100.6107	101.3478	99.1409
170	850	3	114.612	3	101.667	101.837	98.0148
180	900	4	115.019	4	102.672	102.9092	98.7619
190	950	3	124.736	3	108.2763	109.7153	106.3494
220	1100	3	130.7052	2	114.3612	115.4124	111.0998

Figure 4: *Minimum tour length for 220 cities problem instance obtained using NN-ACO algorithm with Ngreedy=3*

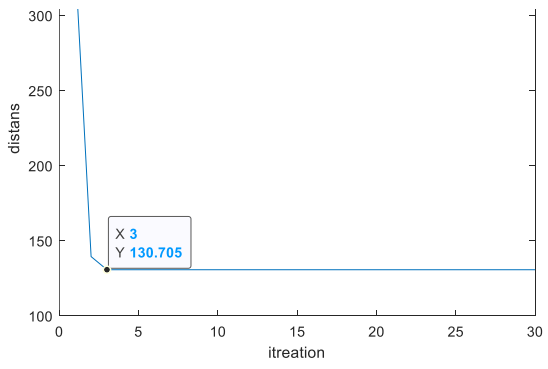


Figure 5: *Minimum tour length for 220 cities problem instance obtained by ACO algorithm*

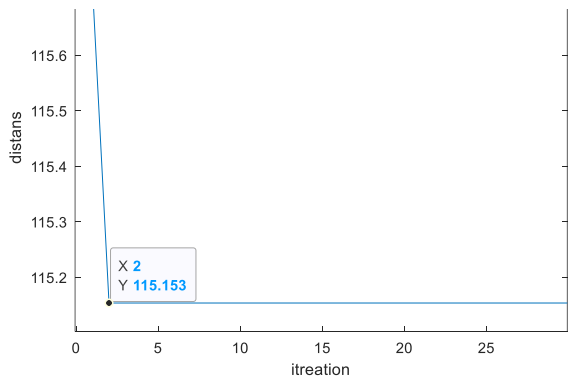


Table 4: Solutions corresponding to different values of M, M=1,2,3,4 for the case of 150 cities

	M	X	Y	Time of initial population construction in seconds
Ngreedy=3	M=1	4	94.6435	0.011957
	M=2	4	94.6435	0.018705
	M=3	2	94.593	0.027437
	M=4	2	94.3254	0.033969

In our experiments, we investigated the impact of the parameter M of the NN algorithm on the quality of the obtained solutions. The results are shown in Table 4. The preliminary results corresponding to different values of M for the 150 cities problem instance, with m = 750 and a maximum iteration limit= 30, show that the solutions obtained at the values M = 1 and M = 2 are identical. However, a slight difference in the cost at the values M = 3 and M = 4 have been observed. Yet, it is noteworthy that all solutions are close with a relative difference of no more than 0.3%, so that the relative percentage error of the distance difference corresponding to the two values M = 2 and M = 3 is 0.05338%, and the relative percentage error at M = 3 and M = 4 is 0.28369%. The best solution is 94.3254 at M = 4, but this comes at the cost of computational time increase. As the value of M increases, the time needed to build the initial population increases, which in turn affects the computational cost. The time difference corresponding to two different values of M is expressed by the relative time increase measure $\boldsymbol{Err_{M,M+1}}$, which is calculated as:

$$\boldsymbol{Err_{M,M+1}}(\%) = \frac{\boldsymbol{T_{M+1} - T_M}}{\boldsymbol{T_M}} \times \mathbf{100\%} \qquad \boldsymbol{M = 1, 2, 3} \tag{13}$$

Hence, we find $\boldsymbol{Err_{1,2}}$ =56.435%, $\boldsymbol{Err_{2,3}}$ = 46.6827%, and $\boldsymbol{Err_{3,4}}$ = 23.807%.

In this study, the hybrid algorithm NN-ACO is compared with the hybrid algorithm presented in [26], which is the ACO-RNN algorithm that combines the ant colony algorithm and attention-based neural networks. Twenty benchmark TSP problem instances from the TSPLIB library are used to test the effectiveness of our proposed hybrid NN-ACO method. In this part of our experiments, the number of ants was taken as 100 and the number of iterations was 200. The obtained results are shown in Table 5, together with the exact optimal solutions for both small and large problems. Moreover, the best solutions yielded by the ACO-RNN algorithm as reported in [26] are included for sake of comparison. From Table 5, it is apparent that the proposed NN-ACO algorithm succeeded to give high quality solutions compared to the optimal solution. The results indicate that the NN-ACO algorithm outperforms the ACO-RNN algorithms in most of the problem instances. It is worth mentioning that the LP-based approach failed to solve large-scale problems with 400 or more cities.

From Table 6, according to the results of Equation 14, it can be seen that the proposed NN-ACO algorithm produces solutions that are very close to the optimal solution and the relative error is mostly very small. This means that the proposed NN-ACO algorithm is stable to compute a satisfactory solution to such problems, with errors falling within the range [0.03, 4.74].

$$\boldsymbol{Err}(\%) = \frac{\mathbf{Best - Optimal}}{\mathbf{Optimal}} \times \mathbf{100\%} \tag{14}$$

Instance	Optimal solution	Best solution NN-ACO	Best solution ACO-RNN	Instance	Optimal solution	Best solution NN-ACO	Best solution ACO-RNN
eil51	426	429.5303	429.3	**pr144**	58537	58687.7961	6300.1
Berlin52	7542	7544.3659	7583.8	**ch150**	6528	6576.334	6987.4
st70	675	677.1096	700.2	**kroA200**	29368	29562.278	33449.1
eil76	538	555.5332	560.4	**rd400**	15281	15638.6445	15900.5
pr76	108159	109193.4374	112000.1	**fl417**	11861	12232.3636	13503.4
rat99	1211	1229.02	1334.5	**pr439**	107217	112270.5673	112286.9
rd100	7910	8007.0342	8036.0	**pcb442**	50778	51903.633	53850.0
kroA100	21282	21320.957	21554.2	**rat575**	6773	7055.2899	7053.7
eil101	629	645.6245	650.8	**p654**	34643	35768.3278	35982.0
lin105	14379	14439.7026	14750	**u724**	41910	43427.875	44002.5
Ch130	6110	6233.3524	6300.1	**rat783**	8806	9223.8153	9335.8

Table 5: *Results of the proposed NN-ACO for small/large problem instances from TSPLIB library*

Table 6: Experimental results of the proposed NN-ACO optimization algorithm for the 22 similar benchmark datasets and their comparison by relative error percentage

Instance	Optimal solution	Best solution NN-ACO	Err (%)	Instance	Optimal solution	Best solution NN-ACO	Err (%)
eil51	426	429.5303	0.828709	**pr144**	58537	58687.7961	0.257608
Berlin52	7542	7544.3659	0.03137	**ch150**	6528	6576.334	0.740411
st70	675	677.1096	0.312533	**kroA200**	29368	29562.278	0.66153
eil76	538	555.5332	3.258959	**rd400**	15281	15638.6445	2.340452
pr76	108159	109193.4374	0.956404	**fl417**	11861	12232.3636	3.130964
rat99	1211	1229.02	1.488026	**pr439**	107217	112270.5673	4.713401
rd100	7910	8007.0342	1.226728	**pcb442**	50778	51903.633	2.216773
kroA100	21282	21320.957	0.183051	**rat575**	6773	7055.2899	4.167871
eil101	629	645.6245	2.643005	**p654**	34643	35768.3278	3.248356
lin105	14379	14439.7026	0.422161	**u724**	41910	43427.875	3.621749
Ch130	6110	6233.3524	2.018861	**rat783**	8806	9223.8153	4.744666

6. Concluding remarks

In this paper, a high-performance hybrid algorithm is developed, which combines ant colony optimization algorithm with the nearest neighbor algorithm to solve the Travel Salesman Problem (TSP). It consists of two main stages. In the first stage, the nearest neighbor algorithm is used to optimize the initial population and 2-opt exchange is used to further improve the obtained solutions. There is a possibility of a stagnant solution at this stage. In the second stage, ACO builds on the solutions obtained in the initialization stage and optimizes the paths through an iterative optimization process. The goal is to minimize the path length taken by the salesman to visit all the cities and returning back to the starting point of the tour. To maximize the enhancement to performance of the algorithm, the algorithm parameters were carefully tuned. The simulation results we obtained show that the proposed NN-ACO algorithm is more effective than the basic ACO algorithm and NN algorithm by providing a faster descent rate and a clear convergence process for solutions, especially when the number of cities increases. The experimental results proved that the proposed approach produces good solutions compared to another recently developed hybrid algorithm in literature (ACO- RNN) and is thus effective for solving large-scale problems, making it a practical solution for TSP.

Conflict of interest

The authors have no conflicts of interest to declare.

Ethics approval

This study did not require ethical approval as it does not involve any human or animal case studies.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data availability

The data that support the findings of this study are available on request from the corresponding author.

Authors contribution

Mohammed Salim Hassan: Formal Analysis, Investigation, Software, Visualization, Writing – Original Draft Preparation. **Yasmine Abouelseoud:** Conceptualization, Investigation, Supervision, Writing – Review & Editing. **Ahmed F. Tayel:**

Investigation, Methodology, Software, Supervision, Writing – Review & Editing.

Acknowledgement

Not applicable.

Human Participants and/or Animals

Not applicable.

References

1. Kızılates, G., & Nuriyeva, F. (2013). A parametric hybrid method for the traveling salesman problem. *Mathematical and Computational Applications*, *18*(3), 459-466.
2. Taiwo, O. S., Josiah, O., Taiwo, A., Dkhrullahi, S., & Sade, O. K. (2013). Implementation of heuristics for solving travelling salesman problem using nearest neighbour and nearest insertion approaches.
3. Kaabi, J., & Harrath, Y. (2019). Permutation rules and genetic algorithm to solve the traveling salesman problem. *Arab journal of basic and applied sciences*, *26*(1), 283-291.
4. Nurdiawan, O., Pratama, F. A., Kurnia, D. A., & Rahaningsih, N. (2020, March). Optimization of Traveling Salesman Problem on Scheduling Tour Packages using Genetic Algorithms. In *Journal of Physics: Conference Series* (Vol. 1477, No. 5, p. 052037). IOP Publishing.
5. Erol, A. H., & Bulkan, S. (2012, July). New genetic algorithm for the travelling salesman problem. In *Proceedings of the 2012 International Conference on Industrial Engineering and Operations Management, Istanbul, Turkey*.
6. Mahmoudinazlou, S., & Kwon, C. (2024). A hybrid genetic algorithm for the min–max Multiple Traveling Salesman Problem. *Computers & Operations Research*, *162*, 106455.
7. Vescan, A., & Pintea, C. M. (2007). Ant colony component-based system for travelling salesman problem. *Applied Mathematical Sciences*, *1*(28), 1347-1357.
8. Brezina Jr, I., & Čičková, Z. (2011). Solving the travelling salesman problem using the ant colony optimization. *Management Information Systems*, *6*(4), 10-14
9. Yang, J., Shi, X., Marchese, M., & Liang, Y. (2008). An ant colony optimization method for generalized TSP problem. *Progress in Natural Science*, *18*(11), 1417-1422.
10. Shi, Y., & Zhang, Y. (2022). The neural network methods for solving Traveling Salesman Problem. *Procedia Computer Science*, *199*, 681-686.
11. Cao, Y., Sun, Z., & Sartoretti, G. (2022, November). Dan: Decentralized attention-based neural network for the minmax multiple traveling salesman problem. In *International Symposium on Distributed Autonomous Robotic Systems* (pp. 202-215). Cham: Springer Nature Switzerland.
12. Karaboga, D., & Ozturk, C. (2011). A novel clustering approach: Artificial Bee Colony (ABC) algorithm. *Applied soft computing*, *11*(1), 652-657.
13. Chen, S. M., & Chien, C. Y. (2011). Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. *Expert Systems with Applications*, *38*(12), 14439-14450.
14. Angeniol, B., Vaubois, G. D. L. C., & Le Texier, J. Y. (1988). Self-organizing feature maps and the travelling salesman problem. *Neural Networks*, *1*(4), 289-293.
15. Somhom, S., Modares, A., & Enkawa, T. (1997). A self-organising model for the travelling salesman problem. *Journal of the Operational Research Society*, *48*(9), 919-928.
16. Masutti, T. A., & de Castro, L. N. (2009). A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem. *Information Sciences*, *179*(10), 1454-1468.
17. Pasti, R., & de Castro, L. N. (2006, July). A neuro-immune network for solving the traveling salesman problem. In *The 2006 IEEE International Joint Conference on*

Neural Network Proceedings (pp. 3760-3766). IEEE.

18. Durbin, R., & Willshaw, D. (1987). An analogue approach to the travelling salesman problem using an elastic net method. Nature 326, 689-691.
19. Dong, G., Guo, W. W., & Tickle, K. (2012). Solving the traveling salesman problem using cooperative genetic ant systems. *Expert systems with applications*, *39*(5), 5006-5011.
20. Marinakis, Y., & Marinaki, M. (2010). A hybrid genetic–Particle Swarm Optimization Algorithm for the vehicle routing problem. *Expert Systems with Applications*, *37*(2), 1446-1455.
21. 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.
22. Khan, I., & Maiti, M. K. (2019). A swap sequence based artificial bee colony algorithm for traveling salesman problem. *Swarm and evolutionary computation*, *44*, 428-438.
23. Jia, H. (2015). A novel hybrid optimization algorithm and its application in solving complex problem. *International Journal of Hybrid Information Technology*, *8*(2), 1-10.
24. Rahman, Md.A., & Parvez, H. (2021). Repetitive Nearest Neighbor Based Simulated Annealing Search Optimization Algorithm for Travelling Salesman Problem, Open Access Library Journal, 8:e7520.
25. Erol, A. H., Er, M., & Bulkan, S. (2012). Optimizing the ant colony optimization algorithm using neural network for the traveling salesman problem. In *Actas de la Conferencia Internacional de* (pp. 83-89).
26. Soh, M., & Likeufack, A. N. (2024). A New Hybrid Algorithm Based on Ant Colony Optimization and Recurrent Neural Networks with Attention Mechanism for Solving the Traveling Salesman Problem.
27. Stützle, T., & Dorigo, M. (1999). ACO algorithms for the traveling salesman problem. *Evolutionary algorithms in engineering and computer science*, *4*, 163-183.
28. Fereidouni, S. (2011). Solving traveling salesman problem by using a fuzzy multi-objective linear programming. *Afr J Math Comput Sci Res*, *4*(11), 339-349.
29. Benhida, S., & Mir, A. (2017). Solving the Traveling Salesman Problem with Exact Methods, International Journal of Enhanced Research in Science, Technology and Engineering, 6(7), 61-66.
30. Katagiri, H., Wu, H., Kakiuchi, Y., Hamori, H., & Kato, K. (2017). A route optimization problem in electrical pcb inspections with alignment operations. *Scientiae Mathematicae Japonicae*, *80*(3), 219-232.
31. C.E. Miller, A.W. Tucker & R.A. Zemlin (1960) Integer programming formulations and travelling salesman problems. J. Ass. Comp. Mach., 7, 326–329.
32. Pataki, G. (2000). The bad and the good-and-ugly: formulations for the traveling salesman problem. *Department of Industrial Engineering and Operations Research. Columbia University*.
33. Sharma, M. K., Chaudhary, S., Rathour, L., & Mishra, V. N. (2024). Modified genetic algorithm with novel crossover and mutation operator in travelling salesman problem. *Sigma Journal of Engineering and Natural Sciences*, *42*(6).
34. AlSalibi, B. A., Jelodar, M. B., & Venkat, I. (2013). A comparative study between the nearest neighbor and genetic algorithms: A revisit to the traveling salesman problem. *International Journal of Computer Science and Electronics Engineering (IJCSEE)*, *1*(1), 110-123.
35. Gao, W. (2020). New ant colony optimization algorithm for the traveling salesman problem. *International Journal of Computational Intelligence Systems*, *13*(1), 44-55.
36. Blum, D., & Dortmund, U. (2013). Ant colony optimization (ACO). *Journal of Interdisciplinary Mathematics*, *8*, 157-168.