BAREKENG: Journal of Mathematics and Its Applications
December 2025 Volume 19 Issue 4 Page 3087–3104
P-ISSN: 1978-7227 E-ISSN: 2615-3017
https://doi.org/10.30598/barekengvol19iss4pp3087-3104

TRAVELING SALESMAN PROBLEM INTEGRATED WITH FUZZY LOGIC ON TOURISM IN D.I. YOGYAKARTA

Uskar Sabilil Mukminin[1*], Irma Sari Yulianti[2], Budi Surodjo[3]

[1,2,3]*Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada*
Jln. Geografi, Bulaksumur Sekip Utara, Yogyakarta, 55281, Indonesia

*Corresponding author's e-mail: * **sabilmuk@gmail.com***

Article History:
Received: 17th April 2025
Revised: 15th May 2025
Accepted: 16th June 2025
Available online: 1st September 2025

Keywords:
Fuzzy logic;
Minimum Spanning Tree;
Nearest Neighbor;
Tourist route optimization;
Traveling salesman problem.

ABSTRACT

The optimization of tourist travel routes has become a crucial factor in enhancing travel efficiency, reducing costs, and optimizing the overall tourist experience. This study focuses on the innovative integration of fuzzy logic with the Traveling Salesman Problem (TSP) to determine the optimal path for visiting several major tourist destinations in the Special Region of Yogyakarta, a methodological approach not previously explored in existing literature. Initially, we perform data fuzzification, followed by fuzzy inference, to obtain fuzzy outputs. These output values are subsequently used to determine the shortest route using TSP. Several algorithms are utilized, including Minimum Spanning Tree (MST) and Nearest Neighbor (NN). The results show that the Prim algorithm in MST generates the most optimal route, with a travel distance of 223.1 km and a travel time of 442 minutes. Integrating fuzzy logic into the TSP framework effectively addresses uncertainties in distance and time, offering a solid foundation for improved travel route planning.

How to cite this article:

U. S. Mukminin, I. S. Yulianti and B. Surodjo., "TRAVELING SALESMAN PROBLEM INTEGRATED WITH FUZZY LOGIC ON TOURISM IN D.I. YOGYAKARTA", *BAREKENG: J. Math. & App.*, vol. 19, iss. 4, pp. 3087-3104, December, 2025.

Journal homepage: *https://ojs3.unpatti.ac.id/index.php/barekeng/*
Journal e-mail: *barekeng.math@yahoo.com; barekeng.journal@mail.unpatti.ac.id*

Research Article · Open Access

1. INTRODUCTION

As one of Indonesia's primary tourist destinations, Yogyakarta must address the logistical challenges posed by its approximately 30 million annual visitors (BPS, 2024). The concentration of travel between major attractions such as Prambanan Temple and Parangtritis Beach often results in inefficient time management and increased costs, ultimately degrading the tourist experience. Therefore, well-planned trips considering both time efficiency and travel distances are essential. Choosing the best route can help save costs and maximize the travel experience by visiting various tourist attractions. Thus, effective coordination between tourism development by the government and individual travel planning is key to achieving maximum benefits for the regional economy and a satisfying travel experience for tourists. This study proposes an optimized travel route system for Yogyakarta tourist destinations by integrating fuzzy logic with the Traveling Salesman Problem (TSP).

In graph and optimization theory, determining the best route falls under the shortest path problem, commonly known as the TSP. The goal is to provide routing solutions that minimize delays in goods delivery and optimize transportation facilities. The TSP was first introduced in the 1930s by Karl Menger, a mathematician and economist, who referred to it as the "Messenger Problem," a problem faced by mail carriers and many travelers. Several studies of TSP, such as Bandara, have applied TSP to an ABC Company in Sri Lanka, which supplies air conditioners throughout the country [1]. Narwadi and Subiyanto have applied an improved genetic algorithm for the traveling salesman problem on Android Google Maps [2]. Numerous studies have also introduced various methods for addressing the TSP that integrate multiple algorithms. For instance, Hao et al. employed a hybrid approach that merges an advanced ant colony optimization technique with a significantly refined simulated annealing method, utilizing clustering as part of their strategy [3], hybrid simulated annealing and tabu search algorithms [4], [5] and [6] using hybrid genetic algorithm in their research.

Studies on the Traveling Salesman Problem (TSP) have continued to evolve in recent years. Placido et al. developed a genetic algorithm based on 2-opt and cone programming to solve the Close-Enough Traveling Salesman Problem (CETSP). They applied it to scheduling solar panel diagnostic missions [7]. Subsequently, Muren et al. proposed a fast and stable mixed steepest descent algorithm to address the TSP in the context of air logistics and emergency scenarios [8]. Gharehgozli et al. introduced a polynomial-time algorithm for solving the high multiplicity asymmetric TSP with a feedback vertex set, which was applied in automated storage and retrieval systems [9]. Zhang and Yang modified the cuckoo search algorithm in a discrete form to solve the TSP and applied it to cutting path optimization in the glass manufacturing industry [10]. Bock et al. conducted a comprehensive survey on various TSP variants in warehouse operations, analyzing their computational complexity and identifying future research directions in the era of warehouse automation [11].

Generally, selecting the best route is based solely on the distance between one location and another. However, road conditions involve more than just distance—other aspects, such as travel time, play an important role. Combining multiple factors, such as distance and travel time, can introduce uncertainties in determining road weight values.

Fuzzy logic is used to address this uncertainty. Fuzzy logic is employed to model the quantities of input received. One method within fuzzy logic is the Tsukamoto fuzzy model, which provides fairly good efficiency values. The output results of fuzzy logic can be used as input for other algorithms, including those for determining the shortest path. In 2018, Anwar et al. presented a fuzzy Tsukamoto for a decision support system for selecting scholarship recipient students [12].

Researchers have previously explored the integration of fuzzy logic with the Traveling Salesman Problem to solve various issues, such as Mukminin et. al, who used dynamic programming and fuzzy logic to solve the traveling salesman problem on Semarang tourism [13]. Kim G applied a dynamic vehicle routing problem framework incorporating fuzzy customer responses [14]. Almahasneh et al. developed an interval-valued intuitionistic fuzzy set approach to optimize the time-dependent TSP [15]. Cheikhrouhou introduced FL-MTSP, a fuzzy logic-based method to solve the multi-objective multiple traveling salesman problem within multi-robot systems [16]. Yang et al. propose a vehicle routing optimization method with fuzzy demand and flexible time windows using a credibility theory-based chance-constrained programming model and a hybrid simulated annealing-genetic algorithm to minimize total logistics costs [17].

In light of the preceding discussion, this research addresses optimizing route selection for multiple tourist attractions within the Special Region of Yogyakarta. In this investigation, we employ the Tsukamoto

fuzzy logic model, considering the interplay between distance and average travel duration to ascertain the weights of the edges within the graph. Subsequently, we implement algorithms such as the Minimum Spanning Tree and Nearest Neighbor to identify the most efficient route. The findings of this research are anticipated to aid travelers in navigating to tourist sites, thereby enhancing the efficiency of their excursions and serving as a foundation for subsequent scholarly inquiries.

2. RESEARCH METHODS

In this chapter, we discuss several aspects of the research methodology. First, regarding data collection, we used secondary data from the Tourism Office of the Special Region of Yogyakarta to identify the leading tourist destinations in this province. Then, we measured the travel distances and average travel times using Google Maps and Google Earth. According to the data, we identified 15 leading tourist destinations in the Special Region of Yogyakarta, with the initial departure point from Yogyakarta International Airport (YIA).

Subsequently, we obtained the travel distance matrix and average travel times between the tourist destinations. This data was then processed using Fuzzy Logic and the Traveling Salesman Problem (TSP) to determine the optimal route. We discuss the fuzzy Tsukamoto to determine the decision variable outputs, which are subsequently used to identify the shortest route. The algorithms we employed include Minimum Spanning Tree and Nearest Neighbor. These algorithms were then compared to each other to obtain the minimum route.

2.1 Fuzzy Logic

Fuzzy logic is a field that incorporates varying degrees of membership within a set, allowing for values that are not restricted to a simple true or false dichotomy. The idea of fuzzy sets was initially proposed by Professor Lotfi A. Zadeh in 1965 as a broadening of the traditional mathematical notion of sets. A fuzzy set consists of a spectrum of values, each assigned a degree of membership that falls within the range of [0,1] [18].

A fuzzy set A in a universe of discourse is represented by a membership function μ_A which expressed as:

$$\mu_A: U \rightarrow [0,1] \tag{1}$$

The fuzzy set A in the universe of discourse U is typically represented as a collection of element pairs $x \in U$ with their corresponding of membership value:

$$A = \{(x, \mu_A(x)) | x \in U\} \tag{2}$$

In general, fuzzy logic is applied to problems that involve uncertainty. The foundation of fuzzy logic is the theory of fuzzy sets, which represent certain conditions within a fuzzy variable, such as the set of intelligent people, the set of tall students, or the set of employees with high salaries. Within fuzzy logic, there are two main processes: fuzzification and defuzzification. Fuzzification transforms a crisp set into a fuzzy set, which involves defining the fuzzy variables and their corresponding fuzzy sets, and then determining the degree of membership between input data and the predefined fuzzy sets for each input variable in the fuzzy rule set. On the other hand, defuzzification converts a fuzzy set into a specific value within the domain of that fuzzy set, resulting in a crisp set [19].

In general, the inference steps of the Tsukamoto fuzzy method are formally represented in Pseudocode 1.

Pseudocode 1. Fuzzy Logic

```
FUNCTION main():
Data input: distance and time matrices between destinations
distance_matrix; time_matrix; fuzzy_outputs
   FOR EACH pair IN distance_matrix AND time_matrix:
      Fuzzification of distance and time
        mu_Distance = fuzzification(pair.distance, "Travel Distance")
        mu_Time = fuzzification(pair.time, "Travel Time")
      Fuzzy inference
```

```
        alpha_predicate, output_values = fuzzy_inference(mu_Distance, mu_Time)
      Defuzzification
        Z_star = defuzzification(alpha_predicate, output_values)
      fuzzy_outputs.append(Z_star)
END FUNCTION
```

2.2 Traveling Salesman Problem

The Traveling Salesman Problem (TSP) seeks to identify the most efficient route for a salesman, requiring him to visit multiple destinations while ensuring that no location is revisited during the journey [20].

The Traveling Salesman Problem in dynamic programming is defined as follows:

$$\min z = \sum_{j=1}^{n} \varphi_{ij}\delta_{ij}\,; \quad i,j = 1,\dots,n \tag{3}$$

s.t.

$$\sum_{i=1}^{n} \delta_{ij} = 1, \quad j = 1,\dots,n \tag{4}$$

$$\sum_{i=1}^{n} \delta_{ji} = 1, \quad j = 1,\dots,n \tag{5}$$

where,

$$\delta_{ij} = \begin{cases} 1, \text{if salesman travels from } i \text{ to } j \\ 0, \qquad\qquad \text{others} \end{cases} \tag{6}$$

2.3 Minimum Spanning Tree (Prim-Kruskal Algorithm)

The Minimum Spanning Tree (MST) of a weighted graph is a tree that consists of a subgraph of the weighted graph with the minimal total weight [21]. There are two standard methods for finding the MST: Prim's and Kruskal's algorithms.

Applying Kruskal's algorithm to obtain a Minimum Spanning Tree (MST) for solving the Traveling Salesman Problem (TSP) is formally represented in Pseudocode 2.

Pseudocode 2. Minimum Spanning Tree (Kruskal's Algoritm)

```
Procedure KRUSKAL_MST(dist_matrix):
1. Initialize:
  - n = size(dist_matrix)
  - edges = []              (List to store all edges)
  - parent = [0..n-1]       (Each node is its own parent initially)
  - rank = [0,..,0]         (For union by rank)
  - mst_edges = []          (To store MST edges)
2. Collect all possible edges
  For i = 0 to n-1 do:
    For j = i+1 to n-1 do:
      If dist_matrix[i,j] ≠ ∞ then:
        Append (dist_matrix[i,j], i, j) to edges
      End If
    End For
  End For
3. Sort edges by weight in ascending order
4. Define FIND(u):  ▷ Path compression
  If parent[u] ≠ u then:
    parent[u] = FIND(parent[u])
  End If
  Return parent[u]
```

```
5. Define UNION(u, v):  ▷ Union by rank
   root_u = FIND(u)
   root_v = FIND(v)
   If root_u ≠ root_v then:
      If rank[root_u] > rank[root_v] then:
         parent[root_v] = root_u
      Else If rank[root_u] < rank[root_v] then:
         parent[root_u] = root_v
      Else:
         parent[root_v] = root_u
         rank[root_u] += 1
      End If
   End If
6. Build MST
   For each (weight, u, v) in edges do:
      If FIND(u) ≠ FIND(v) then:
         UNION(u, v)
         Append (u, v) to mst_edges
      End If
   End For
7. Return mst_edges
End Procedure
Procedure MST_TO_PATH(mst_edges, num_nodes):
1. Initialize:
   - visited = [False,..,False]   (Size num_nodes)
   - path = []
2. Define DFS(u):                 (Depth-first search)
   a. Append u to path
   b. visited[u] = True
   c. For each (u,v) in mst_edges do:
      If not visited[v] then:
         DFS(v)
      End If
   End For
   d. For each (v,u) in mst_edges do:
      If not visited[v] then:
         DFS(v)
      End If
   End For
3. DFS(0)                         (Start from node 0)
4. Return path
End Procedure
```

On the other hand, Prim's algorithm are formally represented in Pseudocode 3.

Pseudocode 3. Minimum Spanning Tree (Prim's Algoritm)

```
Procedure PRIM_MST(graph):
1. Initialize:
   - num_nodes = size(graph)
   - visited = [False, ..., False]        (Track visited nodes)
   - mst_edges = []                       (Store MST edges)
   - start_node = 0                       (Begin from node 0)
   - visited[start_node] = True
2. For _ = 1 to num_nodes-1 do:     (Need n-1 edges for MST)
   a. min_edge_weight = ∞
   b. min_edge = NULL
   c. For u = 0 to num_nodes-1 do:  (Find minimum edge)
      If visited[u] then:
         For v = 0 to num_nodes-1 do:
            If (not visited[v]) and (graph[u][v] < min_edge_weight) then:
               min_edge_weight = graph[u][v]
               min_edge = (u, v)
            End If
```

```
        End For
      End If
    End For
    d. Append min_edge to mst_edges
    e. visited[min_edge.1] = True       (Mark second node as visited)
End For
3. Return mst_edges
End Procedure
Procedure MST_TO_PATH(mst_edges, num_nodes):
1. Initialize:
   - visited = [False, ..., False]
   - path = []
2. Define DFS(u):                       (Nested depth-first search)
   a. Append u to path
   b. visited[u] = True
   c. For each edge in mst_edges do:
      If (edge.0 == u) and (not visited[edge.1]) then:
        DFS(edge.1)
      Else If (edge.1 == u) and (not visited[edge.0]) then:
        DFS(edge.0)
      End If
   End For
3. DFS(0)                               (Start DFS from node 0)
4. Return path
End Procedure
```

2.4 Nearest Neighbor (NN)

The Nearest Neighbor algorithm exemplifies a Greedy approach in problem-solving. This algorithm identifies the optimal choice based solely on the currently available information, without considering the complete dataset. Its simplicity makes it easy to comprehend and implement swiftly [22]. The steps for addressing the Traveling Salesman Problem with the Nearest Neighbor algorithm are formally represented in Pseudocode 4.

Pseudocode 4. Nearest Neighbor

```
Procedure NEAREST_NEIGHBOR_TSP(dist_matrix, n):
1. Initialize:
    visited = [False, ..., False]
    best_path = [0]
    visited[0] = True
    current_node = 0
    min_distance = 0
2. While |best_path| < n do:
    nearest_node = NULL
    nearest_dist = ∞
    For j = 0 to n-1 do:
     If (not visited[j]) and (dist_matrix[current_node][j] < nearest_dist) then:
       nearest_node = j
       nearest_dist = dist_matrix[current_node][j]
     End If
   End For
    Append nearest_node to best_path
    visited[nearest_node] = True
    min_distance += nearest_dist
    current_node = nearest_node
End While
3. Complete the cycle
  min_distance += dist_matrix[current_node][0]
  Append 0 to best_path
4. Return (best_path, min_distance)
End Procedure
```

2.5 High-Level Workflow

In general, the inference steps of the Tsukamoto fuzzy method combined with MST and NN are formally represented in Pseudocode 5.

Pseudocode 5. Fuzzy Logic and Traveling Salesman Problem (MST and NN)

```
Procedure HYBRID_FUZZY_TSP():
Initialize:
  - distance_matrix  ▷ Raw distance data between nodes
  - time_matrix      ▷ Travel time data between nodes
  - fuzzy_matrix = []  ▷ To store combined fuzzy scores
Phase 1: Fuzzy Evaluation
  For each (i,j) in distance_matrix AND time_matrix do:
    a. mu_Distance = FUZZIFICATION(distance_matrix[i,j], "Distance")
    b. mu_Time = FUZZIFICATION(time_matrix[i,j], "Time")
    c. (alpha, outputs) = FUZZY_INFERENCE(mu_Distance, mu_Time)
    d. fuzzy_matrix[i,j] = DEFUZZIFICATION(alpha, outputs)
  End For
Phase 2: TSP Solution Generation
  Algorithm 1: Prim-based Solution
  mst_edges_prim = PRIM_MST(fuzzy_matrix)
  prim_path = MST_TO_PATH(mst_edges_prim, num_nodes)
  prim_distance = CALCULATE_DISTANCE(prim_path, fuzzy_matrix)
  Algorithm 2: Kruskal-based Solution
  mst_edges_kruskal = KRUSKAL_MST(fuzzy_matrix)
  kruskal_path = MST_TO_PATH(mst_edges_kruskal, num_nodes)
  kruskal_distance = CALCULATE_DISTANCE(kruskal_path, fuzzy_matrix)
  Algorithm 3: Nearest Neighbor Solution
  nn_path, nn_distance = NEAREST_NEIGHBOR(fuzzy_matrix)
Phase 3: Result Evaluation
  results = [
    ("Algorithm 1 (Prim)", prim_path, prim_distance),
    ("Algorithm 2 (Kruskal)", kruskal_path, kruskal_distance),
    ("Algorithm 3 (NN)", nn_path, nn_distance)
  ]
  best_solution = ARGMIN(results.distances)
Return (results, best_solution)
End Procedure
```

3. RESULTS AND DISCUSSION

In this study, we define the starting point with the code "*D1*", which refers to Yogyakarta International Airport as the point of departure. Subsequently, the tourist destinations to be visited are denoted as *P1, P2, ..., P15*, with a total of 15 tourist destinations to be explored. The details of the visited tourist destinations are represented in **Table 1**.

Table 1. Tourist Destination Code

Code	Tourist Destination
D1	Yogyakarta International Airport
T1	Prambanan Temple
T2	Ratu Boko Temple
T3	Heha Skyview
T4	Obelix Hills
T5	Breksi Rock Cliff
T6	Gembira Loka Zoo
T7	Parangtritis Beach
T8	Kids Fun
T9	Ngayogyakarta Hadiningrat Palace
T10	Malioboro

Code	Tourist Destination
T11	Jogja Kembali Memorial Monument
T12	Ibarbo Park
T13	Merapi Museum
T14	Ullen Sentalu Museum
T15	Klangon Hills

Table 2 and **Table 3** present the data regarding the travel distances and average travel times of all nodes according to Google Maps and Google Earth. We assume that each node, except for itself, is accessible from any other node.

Table 2. Travel Distance Each Destination

From - to	D1	T1	T2	T3	T4	…	T15
D1	-	65.5	66.3	62.5	69.3	…	75.6
T1	64	-	5.6	15.1	10.1	…	21.9
T2	65.4	5.5	-	12.9	7.7	…	28.2
T3	58.6	14.9	12.9	-	14	…	35.9
T4	67.2	11.2	7.7	11.4	-	…	33.9
⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
T15	74.9	21.2	28.2	35.5	32.5	⋯	-

Table 3. Average Travel Time Each Destination

From - to	D1	T1	T2	T3	T4	…	T15
D1	-	102	105	93	114	…	124
T1	103	-	13	30	25	…	43
T2	103	14	-	26	22	…	52
T3	91	28	25	-	27	…	64
T4	110	28	22	28	-	…	65
⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
T15	118	42	52	62	63	⋯	-

3.1 Determination of Fuzzy Output Based on Fuzzy Tsukamoto

In this study, we employ fuzzy logic to obtain fuzzy output values, which are subsequently processed using several Traveling Salesman Problem (TSP) algorithms. The fuzzy logic data processing is based on the Tsukamoto fuzzy inference system. This inference method transforms linguistic variable values into fuzzy linguistic terms, then makes inference decisions from a set of fuzzy rules.

We begin the data processing by fuzzifying the travel distance data into fuzzy sets and determining their membership values. In this case, the travel distance variable is divided into four fuzzy sets: "Near", "Medium", "Far", and "Very Far". The fuzzification results for the travel distance variable are represented in **Table 4**. The membership function is represented by **Equation (7)** - **Equation (10)**, and the graph of the membership function is depicted in **Figure 1**.

$$\mu_{A_1}(x) = \begin{cases} 1, & \text{if} \quad x \leq 15.12 \\ \dfrac{30.24 - x}{15.12}, & \text{if} \quad 15.12 \leq x \leq 30.24 \\ 0, & \text{if} \quad \text{others} \end{cases} \tag{7}$$

$$\mu_{A_2}(x) = \begin{cases} \dfrac{x - 15.12}{15.12}, & \text{if} \quad 15.12 \leq x \leq 30.24 \\ \dfrac{45.36 - x}{15.12}, & \text{if} \quad 30.24 \leq x \leq 45.36 \\ 0, & \text{if} \quad \text{others} \end{cases} \tag{8}$$

$$\mu_{A_3}(x) = \begin{cases} \dfrac{x - 30.24}{15.12}, & \text{if} \quad 30.24 \leq x \leq 45.36 \\ \dfrac{60.48 - x}{15.12}, & \text{if} \quad 45.36 \leq x \leq 60.48 \\ 0, & \text{if} \quad \text{others} \end{cases} \tag{9}$$

$$\mu_{A_4}(x) = \begin{cases} \dfrac{x - 45.36}{15.12}, & \text{if} \quad 45.36 \leq x \leq 60.48 \\ 1, & \text{if} \quad x \geq 60.48 \\ 0, & \text{if} \quad \text{others} \end{cases} \tag{10}$$

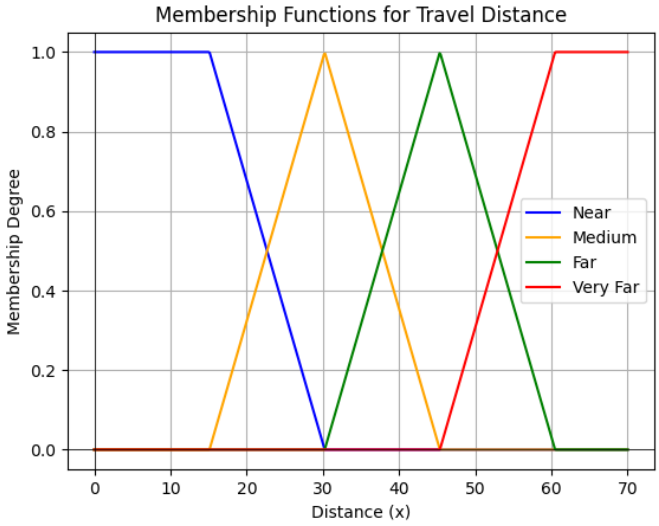

Figure 1. Membership Function of Travel Distance

Table 4. Travel Distance Fuzzification

From - to	D1	T1	T2	T3	T4	…	T15
D1	-	A_4	A_4	A_4	A_4	…	A_4
T1	A_4	-	A_1	A_1	A_1	…	A_1
T2	A_4	A_1	-	A_1	A_1	…	A_2
T3	A_4	A_1	A_1	-	A_1	…	A_3
T4	A_4	A_1	A_1	A_1	-	…	A_2
⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
T15	A_4	A_1	A_2	A_2	A_2	⋯	-

Then, we perform fuzzification for the average travel time data into fuzzy sets. Here, we divide the average travel time variable into four sets: "Short", "Medium", "Long", and "Very Long". The fuzzification results for the average travel time variable are shown in **Table 5**. The membership function is represented by **Equation (11)** - **Equation (14)**, and the graph of the membership function is depicted in **Figure 2**.

$$\mu_{B_1}(x) = \begin{cases} 1, & \text{if} \quad x \leq 10 \\ \dfrac{38.5 - x}{28.5}, & \text{if} \quad 10 \leq x \leq 38.5 \\ 0, & \text{if} \quad \text{others} \end{cases} \tag{11}$$

$$\mu_{B_2}(x) = \begin{cases} \frac{x-10}{28.5}, & \text{if } 10 \le x \le 38.5 \\ \frac{67-x}{28.5}, & \text{if } 38.5 \le x \le 67 \\ 0, & \text{if others} \end{cases} \quad (12)$$

$$\mu_{B_3}(x) = \begin{cases} \frac{x-38.5}{28.5}, & \text{if } 38.5 \le x \le 67 \\ \frac{95.5-x}{28.5}, & \text{if } 67 \le x \le 95.5 \\ 0, & \text{if others} \end{cases} \quad (13)$$

$$\mu_{B_4}(x) = \begin{cases} \frac{x-67}{28.5}, & \text{if } 67 \le x \le 95.5 \\ 1, & \text{if } x \ge 95.5 \\ 0, & \text{if others} \end{cases} \quad (14)$$

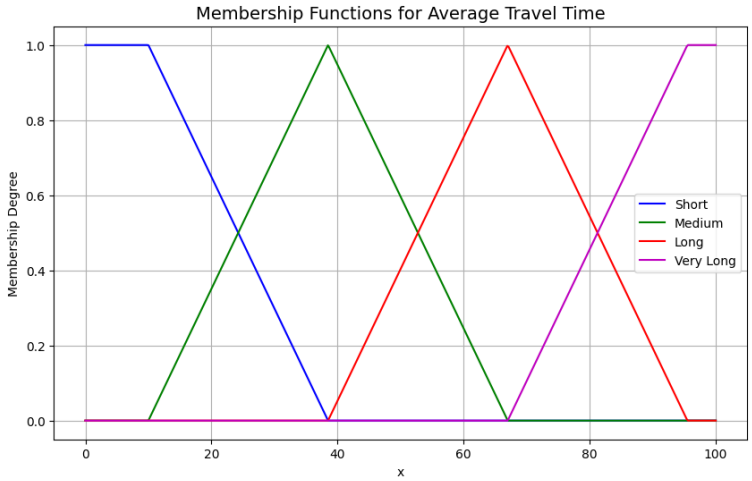

Figure 2. Membership Function of Average Travel Time

Table 5. Average Travel Time Fuzzification

From - to	D1	T1	T2	T3	T4	...	T15
D1	-	B_4	B_4	B_3	B_4	...	B_4
T1	B_4	-	B_1	B_2	B_2	...	B_2
T2	B_4	B_1	-	B_2	B_1	...	B_2
T3	B_4	B_2	B_2	-	B_2	...	B_3
T4	B_4	B_2	B_1	B_2	-	...	B_3
⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
T15	B_4	B_2	B_2	B_3	B_3	...	-

After fuzzifying the distance and average travel time data, we proceeded to formulate the fuzzy rules that will be used for fuzzy inference. We established 16 fuzzy rules, as represented in **Table 6**. The allocation of the 16 rules is derived from the combination of 4 fuzzy sets A and 4 fuzzy sets B, resulting number of possible events $4^2 = 16$ rules [23].

Table 6. Fuzzy Rules

Rules	Travel Distance	Travel Time	Output
R1	Near	Short	Very Very Near
R2	Near	Medium	Very Near
R3	Near	Long	Near
R4	Near	Very Long	Medium

Rules	Travel Distance	Travel Time	Output
R5	Medium	Short	Near
R6	Medium	Medium	Medium
R7	Medium	Long	Far
R8	Medium	Very Long	Far
R9	Far	Short	Medium
R10	Far	Medium	Far
R11	Far	Long	Very Far
R12	Far	Very Long	Very Far
R13	Very Far	Short	Far
R14	Very Far	Medium	Very Far
R15	Very Far	Long	Very Far
R16	Very Far	Very Long	Very Very Far

Then we define $\alpha - predicate$ for all fuzzy sets of travel distance data and average travel time data. For example, we define $\alpha - predicate$ of T1 to T2's fuzzy sets. Its travel distance fuzzy set is A_1 and average travel time fuzzy set is B_1.

The criteria of each rule z_i are shown as follows:
Very Very Near = 0.14
Very Near = 0.29
Near = 0.43
Medium = 0.57
Far = 0.71
Very Far = 0.86
Very Very Far = 1

Table 7 evaluates T1 to T2's fuzzy set according to fuzzy rules. The evaluation is processed until all fuzzy sets in **Table 4** and **Table 5** are determined.

Table 7. Evaluation of T1 to T2's Fuzzy Set According to Fuzzy Rules

R_i	Travel Distance	Travel Time	Membership Value in Travel Distance	Membership Value in Travel Time	$\alpha - pred_i$ (Minimum)	Z_i
R1	Near	Short	1	0.89	0.89	0.14
R2	Near	Medium	1	0.1	0.1	0.29
R3	Near	Long	1	0	0	0.43
R4	Near	Very Long	1	0	0	0.57
R5	Medium	Short	0	0.89	0	0.43
R6	Medium	Medium	0	0.1	0	0.57
R7	Medium	Long	0	0	0	0.71
R8	Medium	Very Long	0	0	0	0.71
R9	Far	Short	0	0.89	0	0.57
R10	Far	Medium	0	0.1	0	0.71
R11	Far	Long	0	0	0	0.86
R12	Far	Very Long	0	0	0	0.86
R13	Very Far	Short	0	0.89	0	0.71
R14	Very Far	Medium	0	0.1	0	0.86
R15	Very Far	Long	0	0	0	0.86
R16	Very Far	Very Long	0	0	0	1

After obtaining the $\alpha - predicate$, we then determine the value of Z^* by employing the weighted average method, as represented by **Equation (15)**.

$$Z^* = \frac{\sum_{i=1}^{n} \alpha_i z_i}{\sum_{i=1}^{n} \alpha_i} \tag{15}$$

Since only [R1] and [R2] which has $\alpha - predicate$ value, so we define fuzzy output value of T1 to T2's fuzzy set only by using [R1] and [R2]. It is represented by calculation as follows:

$$Z^* = \frac{0.89 \times 0.14 + 0.1 \times 0.29}{0.89 + 0.1} = 0.551$$

The Z^* value is the output derived from the fuzzy set that has undergone the inference process based on fuzzy rules. This process is carried out until all fuzzified data have determined their respective output values. The resulting output is presented in **Table 8**.

Table 8. Fuzzy Outputs

From - to	D1	T1	T2	T3	T4	…	T15
D1	-	1	1	0.87	1	…	1
T1	1	-	0.16	0.25	0.22	…	0.45
T2	1	0.16	-	0.22	0.20	…	0.58
T3	0.95	0.23	0.22	-	0.23	…	0.78
T4	1	0.23	0.20	0.23	-	…	0.73
⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
T15	1	0.44	0.58	0.73	0.71	…	-

Following the acquisition of fuzzy outputs in **Table 8**, we analyze them to determine the most efficient route using the Traveling Salesman Problem (TSP) principles. This analysis employs various algorithms, such as Minimum Spanning Tree methods (including Prim's and Kruskal's algorithms) and Nearest Neighbor (NN). The outcomes derived from the data processing of each algorithm will be examined in the subsequent subsection.

3.2 Minimum Spanning Tree (Prim and Kruskal Algorithm)

Based on calculations using Prim's and Kruskal's Algorithm with the help of Python, we get the optimal route as shown in **Table 9** and **Figure 3** - **Figure 4**.

Table 9. Optimal Route of Minimum Spanning Tree

Prim's Algorithm				Kruskal's Algorithm			
Start	**End**	**Distance**	**Time**	**Start**	**End**	**Distance**	**Time**
D1	T9	42.5	68	D1	T9	42.5	68
T9	T6	6.5	21	T9	T6	6.5	21
T6	T8	6	13	T6	T8	6	13
T8	T3	8.3	16	T8	T3	8.3	16
T3	T2	12.9	25	T3	T5	12.8	26
T2	T5	3.5	10	T5	T2	3.5	12
T5	T4	4.2	12	T2	T1	5.5	14
T4	T1	11.2	28	T1	T4	10.1	25
T1	T11	19.3	36	T4	T11	24.6	50
T11	T12	9.3	15	T11	T12	9.3	15
T12	T13	16.7	31	T12	T13	16.7	31
T13	T14	2.8	10	T13	T14	2.8	10
T14	T15	15.2	33	T14	T15	15.2	33
T15	T10	36.6	72	T15	T10	36.6	72
T10	T7	28.1	52	T10	T7	28.1	52
TOTAL		**223.1**	**442**	**TOTAL**		**228.5**	**458**

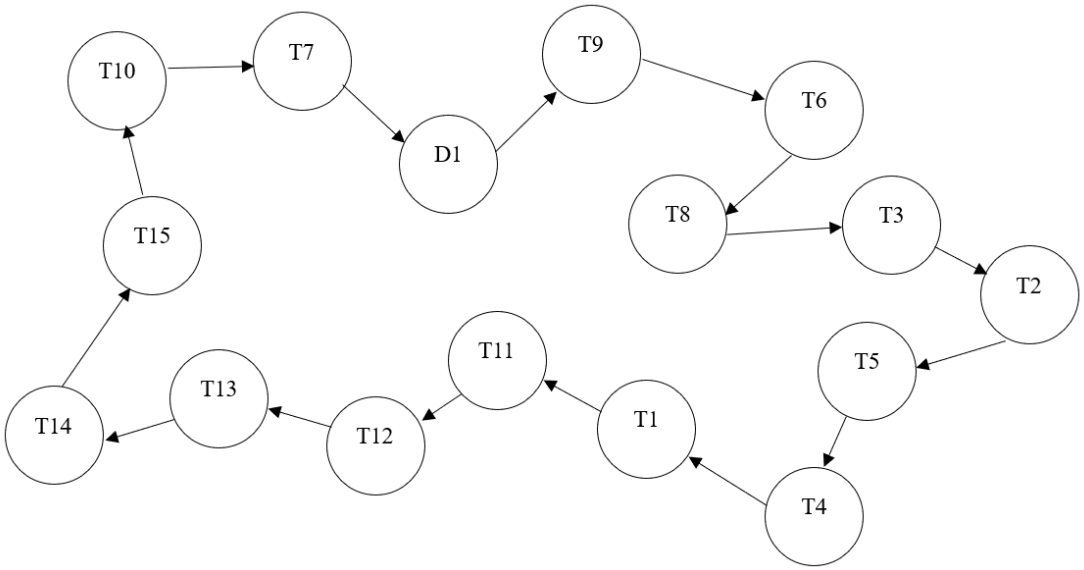

Figure 3. Prim's Algorithm Graph Representation

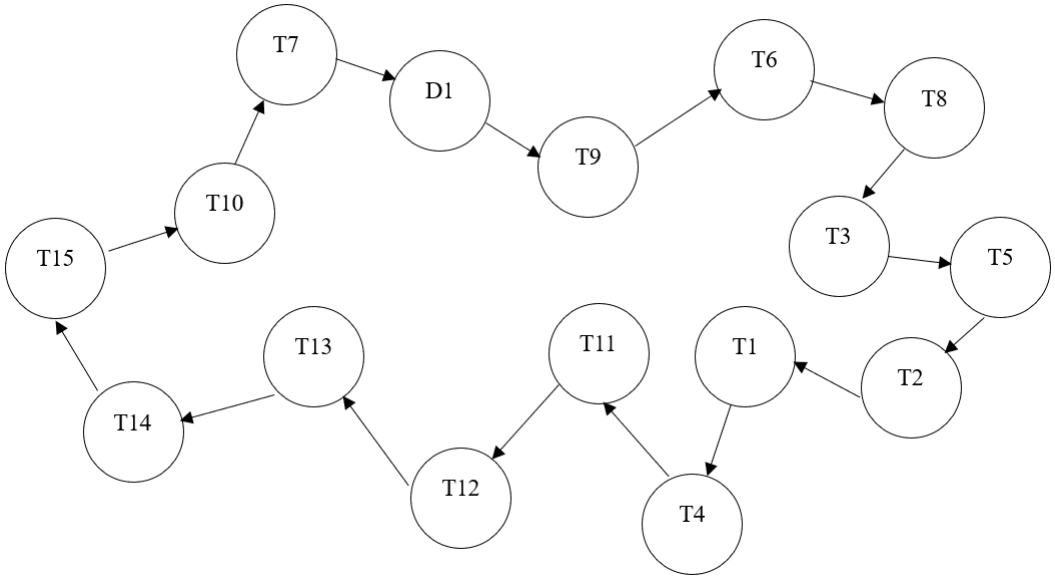

Figure 4. Kruskal's Algorithm Graph Representation

Based on the data analysis employing Prim's algorithm, as represented in **Table 9** and **Figure 3**, the resulting optimal travel route is as follows: commencing from the Depot (D1 – Yogyakarta International Airport), followed sequentially by T9 (Ngayogyakarta Hadiningrat Palace), T6 (Gembira Loka Zoo), T8 (Kids Fun), T3 (Heha Skyview), T2 (Ratu Boko Temple), T5 (Breksi Rock Cliff), T4 (Obelix Hills), T1 (Prambanan Temple), T11 (Jogja Kembali Memorial Monument), T12 (Ibarbo Park), T13 (Merapi Museum), T14 (Ullen Sentalu Museum), T15 (Klangon Hills), T10 (Malioboro), and concluding at T7 (Parangtritis Beach) before back to Depot D1 (YIA). This algorithm achieves the total optimal travel distance of 223.1 kilometers, with a corresponding optimal travel time of 442 minutes.

Furthermore, by applying Kruskal's algorithm, the optimal route obtained—illustrated in **Table 9** and **Figure 4**—comprises the following sequence: beginning from the Depot (D1 – Yogyakarta International Airport), then proceeding to T9 (Ngayogyakarta Hadiningrat Palace), T6 (Gembira Loka Zoo), T8 (Kids Fun), T3 (Heha Skyview), T5 (Breksi Rock Cliff), T2 (Ratu Boko Temple), T1 (Prambanan Temple), T4 (Obelix Hills), T11 (Jogja Kembali Memorial Monument), T12 (Ibarbo Park), T13 (Merapi Museum), T14 (Ullen Sentalu Museum), T15 (Klangon Hills), T10 (Malioboro), and terminating at T7 (Parangtritis Beach) before back to Depot D1 (YIA). The total optimal travel distance resulting from this approach is 228.5 kilometers, with an average optimal travel time of 458 minutes.

3.3 Nearest Neighbor (NN)

Based on the calculation using the Nearest Neighbor (NN) Algorithm with the help of Python, we get the optimal route as shown in Table 10 and Figure 5.

Table 10. Optimal Route of Nearest Neighbor (NN)

Nearest Neighbor (NN)			
Start	**End**	**Distance**	**Time**
D1	T9	42.5	68
T9	T6	6.5	21
T6	T8	6	13
T8	T3	8.3	16
T3	T2	12.9	25
T2	T5	3.5	10
T5	T4	4.2	12
T4	T1	11.2	28
T1	T11	19.3	36
T11	T12	9.3	15
T12	T10	15.1	40
T10	T13	25.6	52
T13	T14	2.8	10
T14	T15	15.2	33
T15	T7	64	101
TOTAL		**246.4**	**480**

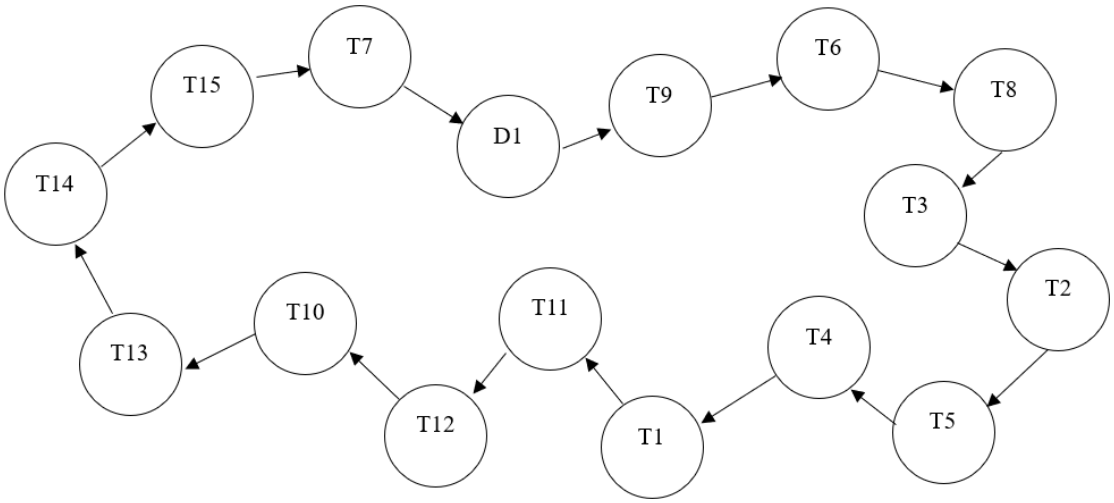

Figure 5. Nearest Neighbor (NN) Algorithm Graph Representation

Based on the data analysis utilizing the Nearest Neighbor (NN) Algorithm, as presented in Table 10 and Figure 5, the resulting optimal route is as follows: the journey commences from the Depot D1 (Yogyakarta International Airport), subsequently followed by T9 (Ngayogyakarta Hadiningrat Palace), T6 (Gembira Loka Zoo), T8 (Kids Fun), T3 (Heha Skyview), T2 (Ratu Boko Temple), T5 (Breksi Rock Cliff), T4 (Obelix Hills), T1 (Prambanan Temple), T11 (Jogja Kembali Memorial Monument), T12 (Ibarbo Park), T10 (Malioboro), T13 (Merapi Museum), T14 (Ullen Sentalu Museum), T15 (Klangon Hills), and concludes at T7 (Parangtritis Beach) before back to Depot D1 (YIA). The total optimal travel distance attained through this algorithm is 246.4 kilometers, with a corresponding average optimal travel time of 480 minutes.

3.4 Result Comparison

Based on the analysis of the optimal route calculations performed in the previous section, we present a comparison of the results among three methods and the comparison between the proposed method and only using single travel distance data, as shown in Table 11.

Table 11. Result Comparison

Algorithm		Proposed Method		Using Travel Distance Data	
		Total Distance	Total Time	Total Distance	Total Time
MST	Prim	223.1	442	284,8	567
	Kruskal	228.5	458	257,5	495
Nearest Neighbor (NN)		246.4	480	294.2	589

Based on Table 11, it can be concluded that integrating outputs from the Tsukamoto fuzzy logic system with various Traveling Salesman Problem (TSP) solution methods yields improved efficiency. Regarding travel distance, the Prim and Kruskal algorithms achieved travel efficiencies of 21.66% and 11.2%. Meanwhile, the Nearest Neighbor (NN) method resulted in an efficiency of 16.25%. Furthermore, regarding travel time, the three algorithms attained efficiencies of 22.4%, 7.48%, and 18.5%, respectively. These findings demonstrate that the proposed method significantly affects optimizing travel distance and travel time.

4. CONCLUSION

From the discussion we have conducted above, we can conclude that:

1. In this article, we conduct a study on integrating fuzzy logic and the Traveling Salesman Problem (TSP) to minimize the travel route for tourism in the Special Region of Yogyakarta. We process data, including travel distance and average travel time, by fuzzifying them into fuzzy sets. Subsequently, we perform a fuzzy inference process to obtain fuzzy outputs. These outputs are then processed using the TSP concept with three algorithms: Minimum Spanning Tree (MST), Prim's and Kruskal's Algorithms, and Nearest Neighbor (NN).
2. This study is expected to serve as an academic recommendation for various stakeholders in making decisions regarding the organization of visits to tourism destinations throughout the Special Region of Yogyakarta Province. The analysis presented herein is particularly suggested for use by travel agencies and tourists who seek to plan their journeys efficiently across multiple destinations within the province.
3. The data processing and analysis results indicate that the Prim algorithm produces the best total travel distance and travel time among the three methods. The integration of fuzzy logic with TSP is effective in generating optimal tourism route solutions.
4. However, additional variables, such as travel time and traffic density, should be incorporated alongside the consideration of travel distance alone. Future research is expected to further develop this concept by considering other variables, such as the time spent at tourist destinations, and so on, thus allowing for a more comprehensive evaluation of factors that were previously overlooked.

AUTHOR CONTRIBUTIONS

Uskar Sabilil Mukminin: Data Curation, Formal Analysis, Methodology, Software, Validation, Investigation. Irma Sari Yulianti: Project Administration, Resources, Visualization. Budi Surodjo: Data Writing - Review and Editing. All authors discussed the results and contributed to the final manuscript.

FUNDING STATEMENT

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

ACKNOWLEDGEMENT

The authors sincerely thank the Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada, for the financial support. The authors also gratefully acknowledge the contributions of all parties who assisted in facilitating this research.

CONFLICT OF INTEREST

The authors declare that no conflicts of interest exist in this study.

REFERENCES

[1] D. Bandara, "AN APPLICATION TO THE TRAVELLING SALESMAN PROBLEM," *Operations Research and Applications : An International Journal*, vol. 4, no. 3/4, pp. 09–20, Nov. 2017, doi: https://doi.org/10.5121/oraj.2017.4402.

[2] T. Narwadi and Subiyanto, "AN APPLICATION OF TRAVELING SALESMAN PROBLEM USING THE IMPROVED GENETIC ALGORITHM ON ANDROID GOOGLE MAPS," in *AIP Conference Proceedings*, American Institute of Physics Inc., Mar. 2017. doi: https://doi.org/10.1063/1.4976899.

[3] T. Hao, W. Yingnian, Z. Jiaxing, and Z. Jing, "STUDY ON A HYBRID ALGORITHM COMBINING ENHANCED ANT COLONY OPTIMIZATION AND DOUBLE IMPROVED SIMULATED ANNEALING VIA CLUSTERING IN THE TRAVELING SALESMAN PROBLEM (TSP)," *PeerJ Comput Sci*, vol. 9, no. 1, pp. 1–31, Oct. 2023, doi: https://doi.org/10.7717/peerj-cs.1609.

[4] İ. Küçükoğlu, R. Dewil, and D. Cattrysse, "HYBRID SIMULATED ANNEALING AND TABU SEARCH METHOD FOR THE ELECTRIC TRAVELLING SALESMAN PROBLEM WITH TIME WINDOWS AND MIXED CHARGING RATES," *Expert Syst Appl*, vol. 134, no. 1, pp. 279–303, May 2019, doi: https://doi.org/10.1016/j.eswa.2019.05.037.

[5] G. Soares, T. Bulhões, and B. Bruck, "AN EFFICIENT HYBRID GENETIC ALGORITHM FOR THE TRAVELING SALESMAN PROBLEM WITH RELEASE DATES," *Eur J Oper Res*, vol. 318, no. 1, pp. 31–42, Oct. 2024, doi: https://doi.org/10.1016/j.ejor.2024.05.010.

[6] M. Rabbani, H. Farrokhi-Asl, and H. Rafiei, "A HYBRID GENETIC ALGORITHM FOR WASTE COLLECTION PROBLEM BY HETEROGENEOUS FLEET OF VEHICLES WITH MULTIPLE SEPARATED COMPARTMENTS," *Journal of Intelligent and Fuzzy Systems*, vol. 30, no. 3, pp. 1817–1830, Mar. 2016, doi: https://doi.org/10.3233/IFS-151893.

[7] A. Di Placido, C. Archetti, and C. Cerrone, "A GENETIC ALGORITHM FOR THE CLOSE-ENOUGH TRAVELING SALESMAN PROBLEM WITH APPLICATION TO SOLAR PANELS DIAGNOSTIC RECONNAISSANCE," *Comput Oper Res*, vol. 145, Sep. 2022, doi: https://doi.org/10.1016/j.cor.2022.105831.

[8] Muren, J. Wu, L. Zhou, Z. Du, and Y. Lv, "MIXED STEEPEST DESCENT ALGORITHM FOR THE TRAVELING SALESMAN PROBLEM AND APPLICATION IN AIR LOGISTICS," *Transp Res E Logist Transp Rev*, vol. 126, pp. 87–102, Jun. 2019, doi: https://doi.org/10.1016/j.tre.2019.04.004.

[9] A. Gharehgozli, C. Xu, and W. Zhang, "HIGH MULTIPLICITY ASYMMETRIC TRAVELING SALESMAN PROBLEM WITH FEEDBACK VERTEX SET AND ITS APPLICATION TO STORAGE/RETRIEVAL SYSTEM," *Eur J Oper Res*, vol. 289, no. 2, pp. 495–507, Mar. 2021, doi: https://doi.org/10.1016/j.ejor.2020.07.038.

[10] Z. Zhang and J. Yang, "A DISCRETE CUCKOO SEARCH ALGORITHM FOR TRAVELING SALESMAN PROBLEM AND ITS APPLICATION IN CUTTING PATH OPTIMIZATION," *Comput Ind Eng*, vol. 169, Jul. 2022, doi: https://doi.org/10.1016/j.cie.2022.108157.

[11] S. Bock, S. Bomsdorf, N. Boysen, and M. Schneider, "A SURVEY ON THE TRAVELING SALESMAN PROBLEM AND ITS VARIANTS IN A WAREHOUSING CONTEXT," *Eur J Oper Res*, vol. 322, no. 4, pp. 1–14, Apr. 2024, doi: https://doi.org/10.1016/j.ejor.2024.04.014.

[12] M. Husain Anwar and M. Furqan, "DECISION SUPPORT SYSTEM FOR SELECTION OF SCHOLARSHIP RECIPIENT STUDENTS USING TSUKAMOTO METHOD FUZZY LOGIC," *International Journal of Information System & Technology Akreditasi*, vol. 8, no. 158, pp. 6–11, Apr. 2024, doi: https://doi.org/10.30645/ijistech.v8i1.341.

[13] U. S. Mukminin, B. Irawanto, B. Surarso, and Farikhin, "VEHICLE ROUTING PROBLEM ON DYNAMIC PROGRAMMING USING FUZZY LOGIC APPROACH FOR TOURIST DESTINATION ROUTE IN SEMARANG," in *AIP Conference Proceedings*, American Institute of Physics Inc., Jun. 2023. doi: https://doi.org/10.1063/5.0140408.

[14] G. Kim, "DYNAMIC VEHICLE ROUTING PROBLEM WITH FUZZY CUSTOMER RESPONSE," *Sustainability (Switzerland)*, vol. 15, no. 5, Mar. 2023, doi: https://doi.org/10.3390/su15054376.

[15] R. Almahasneh, B. Tüú-Szabó, L. T. Kóczy, and P. Földesi, "OPTIMIZATION OF THE TIME-DEPENDENT TRAVELING SALESMAN PROBLEM USING INTERVAL-VALUED INTUITIONISTIC FUZZY SETS," *Axioms*, vol. 9, no. 2, Jun. 2020, doi: https://doi.org/10.3390/axioms9020053.

[16] S. Trigui, O. Cheikhrouhou, A. Koubaa, U. Baroudi, and H. Youssef, "FL-MTSP: A FUZZY LOGIC APPROACH TO SOLVE THE MULTI-OBJECTIVE MULTIPLE TRAVELING SALESMAN PROBLEM FOR MULTI-ROBOT SYSTEMS," *Soft comput*, vol. 21, no. 24, pp. 7351–7362, Dec. 2017, doi: ttps://doi.org/10.1007/s00500-016-2279-7.

[17] T. Yang, W. Wang, and Q. Wu, "FUZZY DEMAND VEHICLE ROUTING PROBLEM WITH SOFT TIME WINDOWS," *Sustainability (Switzerland)*, vol. 14, no. 9, May 2022, doi: https://doi.org/10.3390/su14095658.

[18] R. Saatchi, "FUZZY LOGIC CONCEPTS, DEVELOPMENTS AND IMPLEMENTATION," *Information (Switzerland)*, vol. 15, no. 10, Oct. 2024, doi: https://doi.org/10.3390/info15100656.

[19] M. S. Rahman and M. H. Ali, "ADAPTIVE NEURO FUZZY INFERENCE SYSTEM (ANFIS)-BASED CONTROL FOR SOLVING THE MISALIGNMENT PROBLEM IN VEHICLE-TO-VEHICLE DYNAMIC WIRELESS CHARGING SYSTEMS," *Electronics (Switzerland)*, vol. 14, no. 3, Feb. 2025, doi: https://doi.org/10.3390/electronics14030507.

[20] F. S. Aisyah, W. Winarno, and D. N. Rinaldi, "OPTIMIZING CARTON PRODUCT DELIVERY BY SOLVING TRAVELLING SALESMAN PROBLEM AT PACKAGING COMPANIES," *BAREKENG: Jurnal Ilmu Matematika dan Terapan*, vol. 18, no. 3, pp. 1803–1816, Aug. 2024, doi: https://doi.org/10.30598/barekengvol18iss3pp1803-1816.

[21] S. M. H. Al Kafaji and M. A. K. Shiker, "OPTIMIZING THE MINIMUM SPANNING TREE (MST) AND ITS RELATIONSHIP WITH THE MINIMUM CUT," *Int J Health Sci (Qassim)*, vol. 6, no. 6, pp. 9347–9360, Sep. 2022, doi: https://doi.org/10.53730/ijhs.v6nS6.12436.

[22] S. Hougardy and M. Wilde, "ON THE NEAREST NEIGHBOR RULE FOR THE METRIC TRAVELING SALESMAN PROBLEM," *Discrete Appl Math (1979)*, vol. 195, no. 1, pp. 101–103, Apr. 2015, doi: https://doi.org/10.1016/j.dam.2014.03.012.

[23] R. Li, G. Rose, H. Chen, and J. Shen, "EFFECTIVE LONG-TERM TRAVEL TIME PREDICTION WITH FUZZY RULES FOR TOLLWAY," *Neural Comput Appl*, vol. 30, no. 9, pp. 2921–2933, Nov. 2018, doi: https://doi.org/10.1007/s00521-017-2899-6.