

Implementation of Ant Colony Optimization (ACO) Algorithm for Route Optimization of Tourist Paths in Takengon

Fitra Suryana^{1*}, Nurdin^{2**}, Defry Hamdhana^{3*}

* Department of Informatics, Universitas Malikussaleh, Lhokseumawe, Indonesia

** Department of Information Technology, Universitas Malikussaleh, Lhokseumawe, Indonesia
fitra.210170012@mhs.unimal.ac.id¹, nurdin@unimal.ac.id², defryhamdhana@unimal.ac.id³

Article Info

Article history:

Received 2025-06-02

Revised 2025-07-03

Accepted 2025-08-09

Keyword:

Geographic Information Systems (GIS),

Ant Colony Optimization (ACO),

Shortest Route Optimization,

Tourist Destinations,

Travel Planning System.

ABSTRACT

This study aims to design and implement a system for determining the shortest route between tourist destinations in Takengon using the Ant Colony Optimization (ACO) algorithm. The system is developed to assist travelers in obtaining efficient visitation routes based on distance and travel time. Experiments were conducted on 20 tourist locations, resulting in an optimized route with a total travel distance of 40.40 km and an estimated travel time of 81 minutes. The computation process took only 0.024001 seconds with a memory usage of 20.23 KB. The ACO algorithm was executed using 10 ants with key parameters set to alpha (α) = 1, beta (β) = 2, and rho (ρ) = 0.5. ACO demonstrated high effectiveness in exploring route combinations and iteratively generating near-optimal solutions. The chosen parameters were determined through experimentation to balance solution quality and convergence speed. In addition to generating the optimal visitation sequence, the system also provides complete turn-by-turn navigation instructions, including major roads such as Jalan Lintas Tengah Sumatera and Jalan Lebe Kader. The actual estimated travel route based on the generated navigation covers a distance of 97.4 km with a travel duration of approximately 2 hours and 42 minutes. The results indicate that ACO is an effective and efficient approach for solving medium- to large-scale tourist route optimization problems. The developed system can serve as a practical tool in the tourism sector and has the potential to be adapted and implemented in other tourist regions with similar routing challenges.



This is an open access article under the [CC-BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.

I. INTRODUCTION

The advancement of information and communication technology has driven the widespread adoption of Geographic Information Systems (GIS) for tourism management and planning. GIS enables the visualization, analysis, and interpretation of complex spatial data, supporting more accurate and informed decision-making in the development of tourist destinations [1][2].

Takengon, located in Aceh Province, Indonesia, is rich in natural attractions such as Lake Lut Tawar and scenic green hills. To maximize this potential, efficient route planning is essential to allow tourists to visit multiple destinations

within optimal time and distance while promoting environmental and operational sustainability [3][4][5].

Ant Colony Optimization (ACO) has emerged as a powerful meta-heuristic for solving shortest-path and other combinatorial optimization problems. Inspired by the collective behavior of ants in searching for food, ACO iteratively reinforces promising routes through pheromone updates, enabling adaptive exploration and robust convergence [6][7]. Successful applications range from logistics-such as LPG distribution in Medan, where ACO produced highly efficient delivery routes [8], to tourism, where ACO determined optimal paths for multiple attractions in Batam [9]. Despite these successes, previous

tourism studies have often been limited in scale and have rarely investigated the sensitivity of key ACO parameters.

Nonetheless, ACO offers unique advantages in terms of parallel exploration and adaptive optimization, particularly when parameter tuning is involved.

This study addresses existing gaps by implementing an ACO-based route planning system integrated with GIS to optimize travel across 20 tourist destinations in Takengon. The focus includes evaluating ACO's performance on medium-to-large-scale cases, analyzing the sensitivity of parameters α , β , and ρ , and providing interactive map visualizations and turn-by-turn navigation. The results are expected to support tourism development in Takengon and be adaptable to other regions with similar geographical and tourism characteristics[10].

This study addresses those gaps by integrating ACO with GIS to optimize travel across 22 tourist destinations in Takengon. The research evaluates ACO's performance on medium-to-large-scale route planning, analyzes the impact of parameters α , β , and ρ on solution quality, and provides interactive map visualizations with turn-by-turn navigation. The resulting system is expected to enhance tourism development in Takengon and offer a scalable template for other regions with similar geographical and routing challenges.

This study aims to implement the Ant Colony Optimization (ACO) algorithm for optimizing tourist routes in Takengon. The main focus of the research is to evaluate the performance of ACO in generating optimal routes based on distance and travel time.

This study specifically focuses on ACO. Comparisons with other algorithms such as Dijkstra, Greedy, or Brute Force are not included, as the scope of the research is directed toward examining the characteristics, convergence behavior, and parameter sensitivity of the ACO algorithm.

II. METHODOLOGY

In this study, the author uses the Waterfall method. The Waterfall method is an approach that provides a sequential or step-by-step flow of software development. The stages in the Waterfall method are as follows:

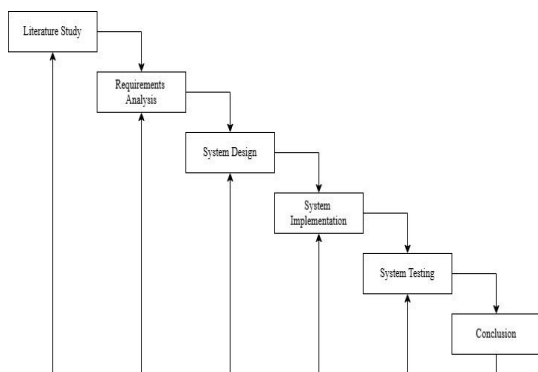


Figure 1. Research Flowchart

This study adopts the Waterfall method, which consists of several structured stages. The first stage is a literature review, conducted to examine the fundamental theories of Ant Colony Optimization (ACO), Geographic Information Systems (GIS), and related previous studies, serving as a foundation for system design. The next stage is the requirements analysis, which identifies the necessary hardware and software specifications. The hardware used includes a laptop with an Intel N4020 processor and 4GB of RAM, while the software includes Windows 11, XAMPP, Visual Studio Code, Google Chrome, and MySQL as the database management system. The system design phase involves defining the system architecture, ACO algorithm flow, database structure, and user interface layout.

This is followed by the system implementation phase, where a web-based application is developed, integrating the ACO algorithm with interactive map visualization using LeafletJS. The system is then evaluated through functional testing, measuring route accuracy, computation time, and route navigation display. The final stage involves drawing conclusions based on system performance and the effectiveness of the ACO algorithm in generating optimal routes, along with recommendations for further development.

A. Literatur Study

1) SIG

Geographic Information System (GIS) is a computer-based system used to collect, manage, analyze, and visualize geographic data. It integrates hardware, software, and spatial data to present information in the form of maps, charts, or reports. GIS is widely applied in fields such as urban planning, transportation, and resource management, serving as a valuable tool for location-based decision-making and problem-solving[11][12].

2) Leaflet

Leaflet is an open-source JavaScript library designed to simplify the creation of interactive and responsive web-based maps. It is known for its ease of use, even for developers with little GIS experience. Leaflet supports GeoJSON data, enabling clickable markers with detailed information. It also integrates with various map providers like OpenStreetMap and Google Maps, offering flexibility in displaying map layers. Created by Vladimir Agafonkin and later joined Mapbox in 2013, Leaflet has become a popular tool for building dynamic and customizable maps with minimal setup[13][14].

3) Graph

According to Arga et al. (2021), a graph is a mathematical structure consisting of a set of vertices (nodes) connected by edges or arcs. It is commonly represented as points (vertices) connected by lines (edges). A graph G can be defined as a pair of sets $G = (V, E)$, where V is the set of vertices and E is the set of edges that connect the vertices[15].

4) *Ant Colony Optimization Algorithm*

The history of Ant Colony Optimization (ACO) began in 1996, when the behavior of ants was observed to develop an algorithm known as Ant Colony Optimization or Ant Algorithm. ACO was designed as a multi-agent simulation that uses the natural metaphor of ants to provide solutions to problems in a physical space. The ant algorithm was first introduced by Manderick and Moyson and was later widely developed by Marco Dorigo. Dorigo utilized probabilistic techniques to solve computational problems and find the best path[16][17].

In 2004, Dorigo and Thomas stated that the algorithm was inspired by the observation of ants' behavior, which led to the design of a new algorithm to address optimization problems. ACO is based on the analogy of the behavior of ants finding a path from their nest to a food source. Ants have the ability to use their sensory tools to explore complex environments, find food, and return to their nest while leaving a pheromone trail along the path they take[18].

Ant Colony Optimization (ACO) is an effective heuristic approach to solving optimization problems by finding good solutions. This approach uses spreading calculations to avoid getting trapped in local optima, as well as the application of a greedy algorithm that can generate fairly good solutions in the early stages of the search[19].

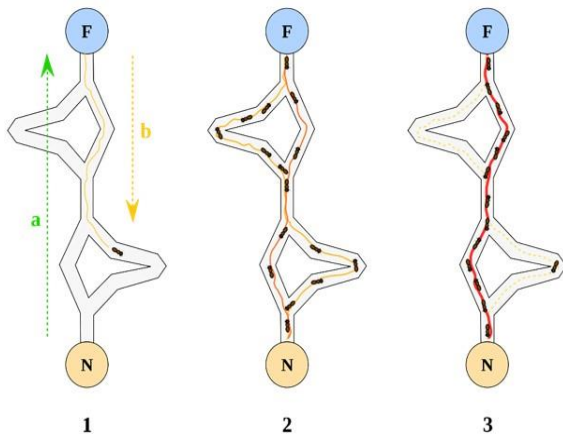


Figure 2. AntCO Algorithm Illustration

Ant Colony Optimization (ACO) is a heuristic method that mimics the behavior of an ant colony in finding the shortest path between the nest and the food source, known as the ant system. Naturally, an ant colony is able to find the shortest route from the nest to the food source. The colony of ants uses the pheromone trail left on the path they have traveled to discover this route. The more ants that pass through a path, the clearer the trail becomes. As a result, paths that are only passed by a few ants will be less frequently traveled and may eventually be abandoned. In contrast, paths that are frequently traversed by ants will become more densely populated by ants, and eventually, all ants will choose that path[20].

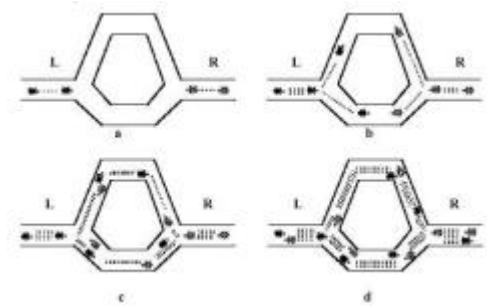


Figure 3. Ant Illustration

According to Dorigo Marco and Thomas (2004), "Informally, the ACO algorithm can be imagined as the interaction of three procedures: ConstructAntsSolutions, UpdatePheromones, and DaemonActions." Essentially, more ants pass through the lower path than the upper path. Meanwhile, the pheromone on the upper path has mostly evaporated, causing ants to avoid that route. The more ants that travel the lower path, the more ants will follow it. Conversely, the fewer ants that travel the upper path, the fewer pheromones are left, and they may even disappear entirely. This is how the shortest path between the nest and the food source is selected.

5) *Unified Modelling Language (UML)*

Unified Modeling Language (UML) is a standard language used to describe, explain, and build software systems. UML is a method applied in object-oriented system development and serves as an important tool in the development process. Some of the tools used in designing object-oriented systems with UML include Use Case Diagrams, Activity Diagrams, Sequence Diagrams, and Class Diagrams. These tools help in designing and visualizing various aspects of the system, from user interactions to internal structure and workflow[21].

B. *Requirements Analysis*

1) *Hardware Requirements Analysis*

Hardware analysis is a crucial step to ensure the system runs smoothly and efficiently. The selected components must meet both functional and technical needs. The required hardware includes:

- Advan Soulmate-A89ASFQ Laptop @ 1.10GHz
- Intel® N4020 Processor
- 4GB RAM
- 256GB SSD
- 128GB eMMC

2) *Software Requirements Analysis*

The software used in the development and operation of this application is essential to support its functionality and performance. The required software includes:

- Operating System: Windows 11
- Web Browser: Google Chrome
- Local Server: XAMPP v3.3.0
- Text Editor: Visual Studio Code
- Database Management System (DBMS): MySQL

C. System Design

1) Use Case Diagram

A use case is a description of the functions a system performs from the user's perspective. It outlines what processes the system will handle and the components involved. In a use case, scenarios are used to illustrate the sequence of steps that represent interactions between the user and the system, including both user actions toward the system and the system's responses[22].

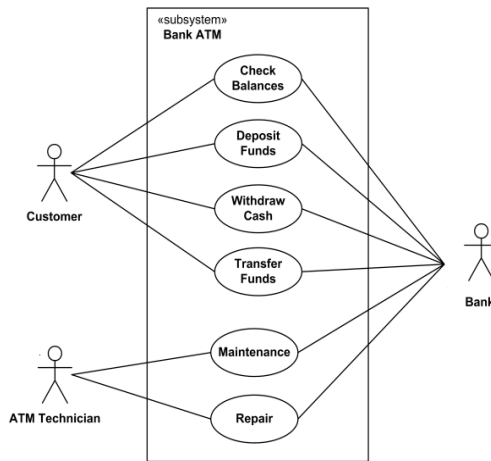


Figure 4. Use Case Diagram

2) Sequence Diagram

A Sequence Diagram is a type of UML diagram used to illustrate interactions between objects in a system over time. It shows how objects communicate through messages in a specific sequence. Each object is represented by a vertical line, and messages are shown as horizontal arrows indicating the order and direction of communication. Sequence diagrams help visualize dynamic processes and make it easier to identify critical points or potential errors in object interactions[23].

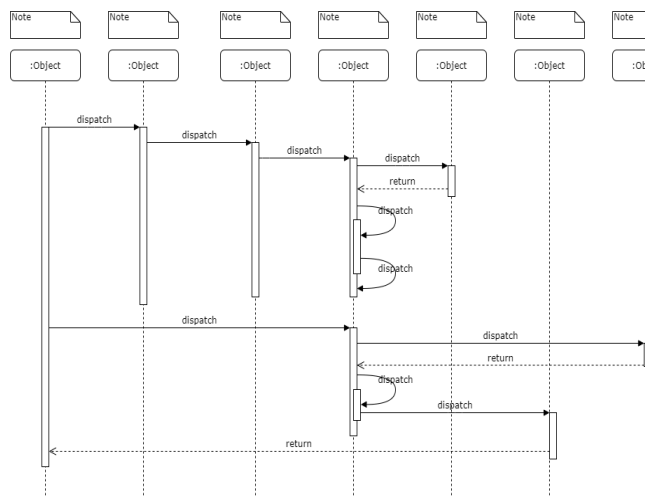


Figure 5. Sequence Diagram

3) Activity Diagram

An Activity Diagram is a visual tool used to represent workflows or processes within a system, showing how a sequence of actions is connected and ordered. It illustrates the steps taken by users or the system to complete a task, from start to finish. Activities are shown as boxes, and arrows indicate the flow between them. This diagram helps map business processes, system workflows, or component interactions, and supports the analysis of efficiency and identification of potential issues in the process flow[24].

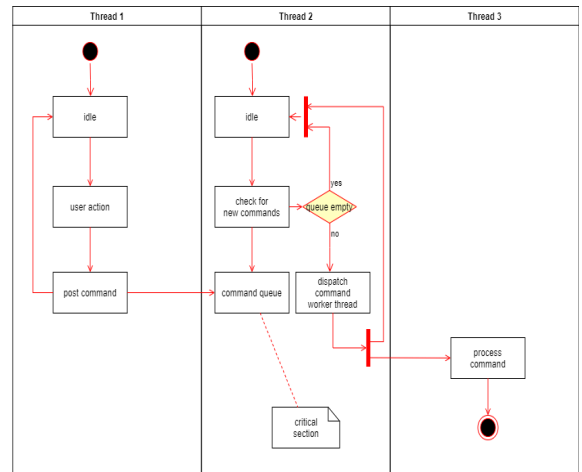


Figure 6. Activity Diagram

4) Class Diagram

A Class Diagram is a type of UML diagram used to represent the static structure of a system, focusing on the classes and their relationships. Each class is shown as a box divided into three parts: class name, attributes, and methods. The diagram also displays relationships such as inheritance, association, and aggregation or composition, illustrating how objects interact within the system. Class diagrams are essential for designing the foundational structure of software systems, providing a clear overview of how data and functions are organized and supporting efficient development and maintenance[25].

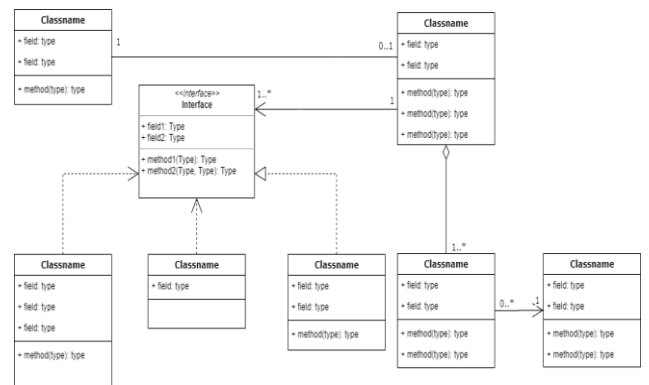


Figure 7. Class Diagrams

III. RESULT AND DISCUSSION

All results presented in this section are derived from the optimization process using the ACO algorithm. This study does not include comparisons with other algorithms, as the approach focuses on internally evaluating the performance of ACO, including result stability, computational time, and the influence of parameters α , β , and ρ .

A. Data Description

The data used in this study includes information about 22 tourist destinations in Takengon, consisting of various sites with details such as name, address, and geographical coordinates in the form of latitude and longitude. These destinations include Terminal Tipe A Paya Ilang, Pasar Inpres Takengon, Rembele Takengon Airport, Pantai Terong, and others spread across multiple sub-districts in Central Aceh Regency.

The geographical coordinates allow the system to accurately calculate the distances between locations and support route optimization. However, for the purpose of simulation and performance testing, the Ant Colony Optimization (ACO) algorithm was applied to a selected subset of 20 locations. This algorithm, inspired by the foraging behavior of ant colonies in finding the shortest paths, was used exclusively to determine the most efficient travel route based on distance and travel time.

TABLE I
RESEARCH DATASET

T	Location	Address	Latitude	Longitude
T1	Terminal Tipe A Paya Ilang	Blang Kolak II, Kec. Bebesen, Kabupaten Aceh Tengah, Aceh 24471	4.625518	96.835620
T2	Pasar Inpres Takengon	Takengon Tim., Kec. Lut Tawar, Kabupaten Aceh Tengah, Aceh 24519	4.623034	96.850663
T3	Bandar Udara Rembele Takengon	Bale Atu, Kec. Bukit, Kabupaten Bener Meriah, Aceh 24582	4.724459	96.849380
...
...
...
T22	Galeri Kopi Indonesia	Kayu Kul, Kec. Pegasing, Kabupaten Aceh Tengah, Aceh 24552	4.605249	96.818384

B. Ant Colony Optimization Algorithm Scheme

The scheme of the Ant Colony Optimization (ACO) algorithm stages can be described as follows:

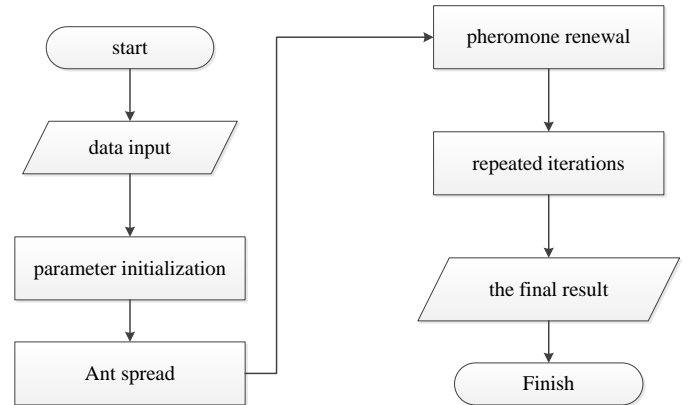


Figure 8. Ant Colony Optimization Algorithm Schema

1. Start

```

process.php X
pages > aco > process.php
1 <?php
2 include '../config/config.php';
3 include '../config/koneksi.php';
.
  
```

2. Input Data: The system receives input including graphdata, list of nodes, distances between nodes, and ACO parameters. This data is essential for determining the optimal path based on virtual ant movement.

```

// Validasi input
if (!isset($_POST['start_location']) || !isset($_POST['total_locations']) ||
    !isset($_POST['ant_count']) || !isset($_POST['alpha']) ||
    !isset($_POST['beta']) || !isset($_POST['rho'])) {
    throw new Exception("Semua field harus diisi!");
}
  
```

3. Parameter Initialization: The system initializes key parameters:

- Number of ants
- Pheromone evaporation rate
- α (pheromone influence) and β (heuristic influence)
- Initial pheromone value on each edge

All core parameters (a – d) are initialized at the beginning of the antColonyOptimization() function.

```

function antColonyOptimization($distances, $start, $numLocations,
    $antCount, $alpha, $beta, $rho, $maxIterations = 500) {
    $n = count($distances);
    $pheromone = array_fill(0, $n, array_fill(0, $n, 0.1));
    $bestDistance = PHP_FLOAT_MAX;
    $bestPath = [];
    $globalBestPath = [];
    $globalBestDistance = PHP_FLOAT_MAX;
    $noImprovement = 0;
    $maxNoImprovement = 50;
  }
  
```

- \$antCount → (a) jumlah semut
- \$rho → (b) tingkat penguapan feromon
- \$alpha, \$beta → (c) pengaruh feromon & heuristik
- \$pheromone matriks → (d) nilai awal tiap edge

TABLE II
TABLE OF ACO PARAMETER SENSITIVITY RESULT

Alpha (α)	Beta (β)	Rho (ρ)	Best distance (km)	Convergence Iteration
0.5	1	0.3	41.3	45
1.0	2	0.5	40.4	32
2.0	3	0.5	40.9	39

The results indicate that the combination of $\alpha = 1$, $\beta = 2$, and $\rho = 0.5$ yields the best outcome, with the shortest travel distance and the fastest convergence, and is therefore selected as the main configuration for the system.

4. Ant Distribution: Virtual ants begin exploring the graph from the starting node, selecting paths based on pheromone levels and distance heuristics. Paths with stronger pheromone and shorter distances are more likely to be chosen.

```
// Meningkatkan jumlah semut
for ($ant = 0; $ant < max(30, $antCount); $ant++) {
    $visited = array_fill(0, $n, false);
    $path = [$start];
    $visited[$start] = true;
    $currentDistance = 0;

    // Konstruksi jalur
    while (count($path) < $numLocations) {
        $current = end($path);
        $probabilities = [];
        $total = 0;

        for ($next = 0; $next < $n; $next++) {
            if (!$visited[$next]) {
                $distance = $distances[$current][$next];
                $visibility = 1.0 / ($distance > 0 ? $distance : 0.0001);
                $tau = pow($pheromone[$current][$next], $alpha);
                $eta = pow($visibility, $beta);
                $probabilities[$next] = $tau * $eta;
                $total += $probabilities[$next];
            }
        }
    }
}
```

5. Pheromone Update: After all ants complete their paths, pheromone levels are updated through:

- Evaporation: Reduces pheromone to prevent premature convergence

```
// Update feromon
for ($i = 0; $i < $n; $i++) {
    for ($j = 0; $j < $n; $j++) {
        $pheromone[$i][$j] *= (1 - $rho);
        if ($pheromone[$i][$j] < 0.1) $pheromone[$i][$j] = 0.1;
    }
}
```

- Deposition: Adds pheromone on shorter paths to reinforce good routes

```
// Tambahkan feromon baru
foreach ($antPaths as $index => $path) {
    $distance = $antDistances[$index];
    if ($distance > 0) {
        $contribution = 1.0 / $distance;
        $multiplier = ($distance === $bestDistance) ? 2.0 : 1.0;

        for ($i = 0; $i < count($path) - 1; $i++) {
            $from = $path[$i];
            $to = $path[$i + 1];
            $pheromone[$from][$to] += $contribution * $multiplier;
            $pheromone[$to][$from] += $contribution * $multiplier;
        }
    }
}
```

6. Iteration Loop: The process of ant distribution and pheromone update is repeated for a set number of

iterations or until convergence. The system continuously updates the best path found so far.

The entire process of distribution and pheromone updating is repeated in each iteration.

```
for ($iteration = 0; $iteration < $maxIterations && $noImprovement < $maxNoImprovement; $iteration++) {
    $antPaths = [];
    $antDistances = [];
}
```

7. Final Result: After all iterations, the path with the highest pheromone concentration is selected as the optimal route. The antColonyOptimization() function returns the best route and the total distance.

```
return [
    'path' => $globalBestPath,
    'distance' => $globalBestDistance
];
}
```

The route is then converted to IDs and stored in the hasil optimasi database table.

```
// Simpan hasil ke database
$pathStr = implode(',', $pathIds);
$sql = "INSERT INTO hasil_optimasi (
    algoritma,
    start_location,
    total_locations,
    ant_count,
    alpha,
    beta,
    rho,
    path,
    distance,
    execution_time,
    memory_usage,
    created_at
) VALUES (
    'ACO',
    ?,
    ?,
    ?,
    ?,
    ?,
    ?,
    $pathStr,
    ?,
    ?,
    ?,
    NOW()
)";
```

8. End

After saving the optimization results, the user is redirected to the results page.

```
$lastId = mysqli_insert_id($koneksi);
header("Location: hasil.php?id=$lastId");
exit();
```

C. System Implementations

This section presents the implementation of the Ant Colony Optimization (ACO) algorithm in a tourist route planning system for Takengon. Users input destination data including name, address, and geographic coordinates. The system processes this data using ACO to generate the shortest route, which is visualized through an interactive map. The interface is designed to be simple and informative, helping users plan their trips efficiently. This system supports travel

optimization and can be further developed into web or mobile applications in the future.

1) Login Page

This Login Page serves as the initial interface of the shortest route determination system that utilizes the Ant Colony Optimization (AntCO) algorithm.

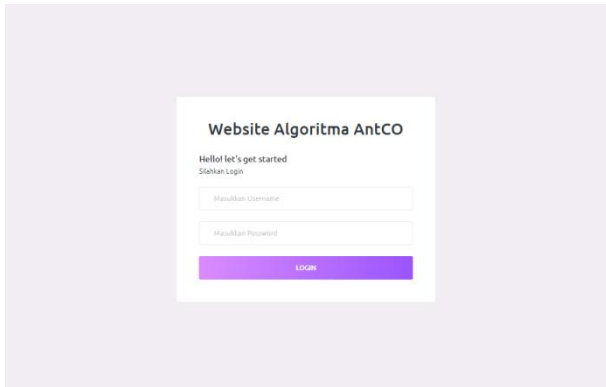


Figure 9. Login Page

2) Dashboard Page

This Dashboard Page provides a summary of the activities within the shortest route determination system that uses the Ant Colony Optimization (AntCO) algorithm.

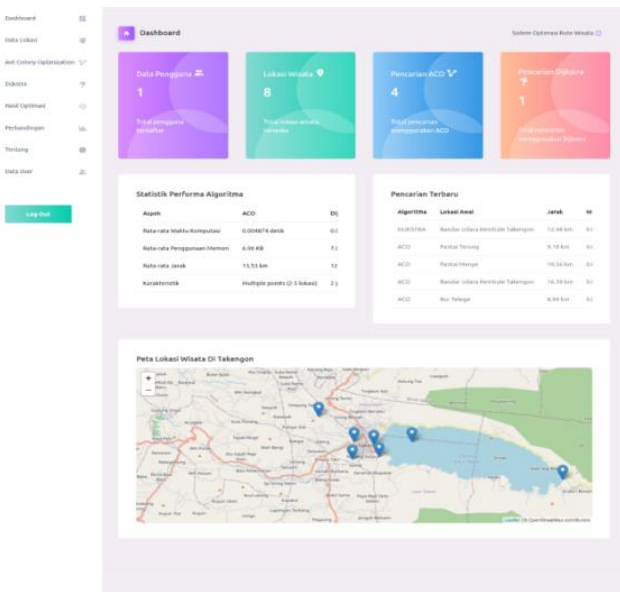


Figure 10. Dashboard Page

3) Location Data Page

The map displayed at the top shows the geographical positions of the entered tourist locations, making it easier for the Admin to visualize the distribution of those destinations. Although the map is interactive and supports zoom and pan functions using LeafletJS, it does not operate in real-time. The map displays static location data and does not reflect dynamic changes such as live user movement or real-time traffic conditions.

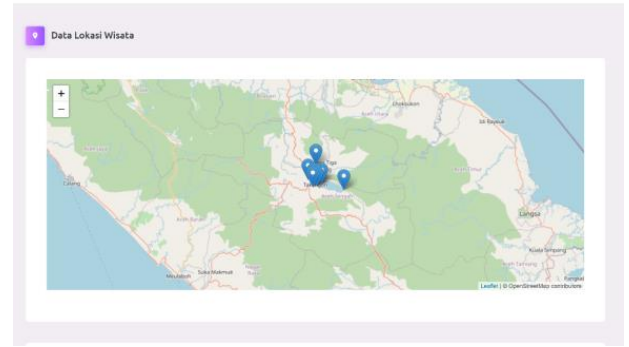


Figure 11. Location Data Page

4) Overview of the Location Data Page

The Location Data Page in this system contains a list of tourist location names used in route calculations. Through this page, users can manage existing location data and add new locations to support data updates and optimize route calculations.

No	ID Lokasi	Nama Lokasi	Alamat	Latitude	L
1	T1	Terminal Tipe A Paya Kang	Bilang Kolak II, Kec. Bebesen, Kabupaten Aceh Tengah, Aceh 24471	4,625518	5
2	T2	Pasar Inpres Takengon	Takengon Tim., Kec. Lut. Tawar, Kabupaten Aceh Tengah, Aceh 24519	4,623034	5
3	T3	Bandar Udara Rembule Takengon	Beir Atu, Kec. Bukit, Kabupaten Bener Meriah, Aceh 24582	4,724459	5
4	T4	Bur Telege	Hakim Bale Bujang, Kec. Lut. Tawar, Kabupaten Aceh Tengah	4,612352	5
5	T5	Pantai Terong	Takengon, Ulu Nuhi, Bebesen, Ulu Nuhi, Kec. Bebesen, Kabupaten Aceh Tengah	4,647565	5
6	T6	Lancuk Leweng	Asir Asir, Kec. Lut. Tawar, Kabupaten Aceh Tengah	4,610113	5
7	T7	Danau Lut. Tawar	Danau, Kabupaten Aceh Tengah, Aceh	4,625361	5
8	T8	Pantai Menye	Kala Bintang, Kec. Bintang, Kabupaten Aceh Tengah	4,592316	5

Figure 12. Overview of the Location Data Page

5) AntCO Page

The AntCO (ACO) Page is designed to allow users, particularly the Admin, to find the optimal route using the ACO algorithm. Users are required to select a starting location as the departure point and then enter other parameters such as the number of destinations to visit, the number of ants, and the alpha (α), beta (β), and rho (ρ) values used in the optimization process.

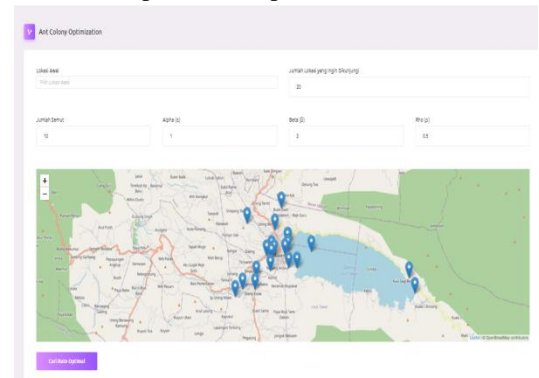


Figure 13. AntCO Page

6) AntCO Implementation Page

The results of route optimization using the AntCO (ACO) algorithm display the best calculated route for tourism travel in Takengon. Based on the input data, the system has computed and suggested the optimal route by considering the starting location, destination points, and user-defined parameters such as the number of ants and the alpha, beta, and rho factors.

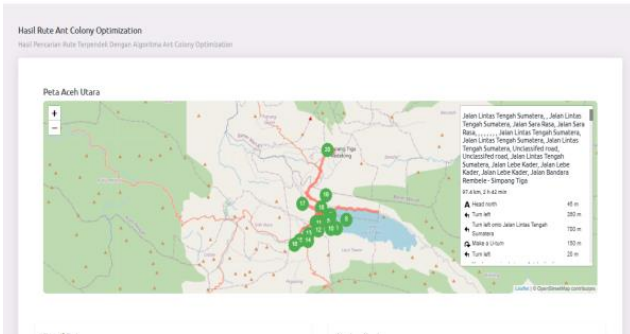


Figure 14. Map Display of Route Search Results Using the AntCO Algorithm

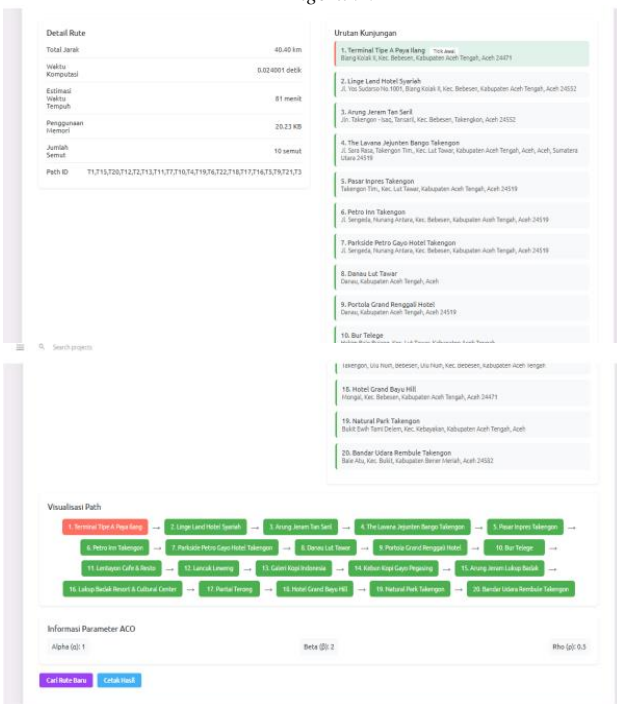


Figure 15. Detailed Results of Shortest Route Optimization Using the AntCO Algorithm

Based on the map visualization, the total travel distance for the tourist route is 40.40 km, with an estimated travel time of approximately 81 minutes. The route optimization process using the Ant Colony Optimization (ACO) algorithm was completed efficiently, taking only 0.024001 seconds and utilizing 20.23 KB of memory, indicating high performance in data processing.

In this optimization process, the system employed 10 virtual ants as search agents, navigating the network based

on a balance of pheromone trails and distance heuristics. The parameters used in the simulation were $\alpha = 1$, $\beta = 2$, and $\rho = 0.5$, which helped maintain an effective trade-off between exploration and exploitation in finding the shortest route.

The journey begins from a user-selected starting point, and the system calculates the most efficient path through 20 tourist destinations, taking into account distance and travel time. The route includes major roads such as Jalan Lintas Tengah Sumatera and Jalan Lebe Kader, as well as smaller roads leading to various points of interest.

Additionally, the system presents the optimized visitation sequence through an interactive map accompanied by navigation instructions. With this detailed information, users can follow the recommended route either manually or via supported navigation applications, ensuring a more structured and time-efficient travel experience.

7) User Data Page

The User Data Page is used to manage information of users who have access to the system. On this page, the Admin can view a list of registered users, including their names and usernames.

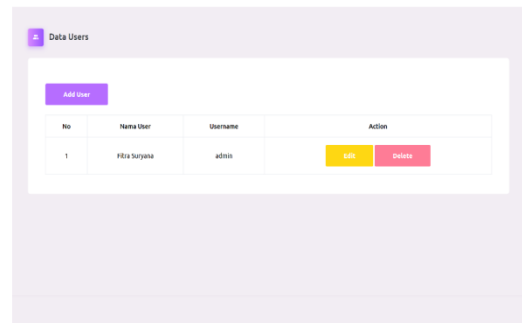


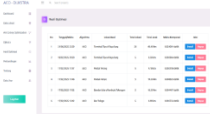
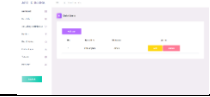
Figure 16. User Data Page

B. System Testing

White Box Testing is used to verify that each system component operates according to the design. This testing focuses on the internal structure and logic flow, including conditions, loops, and code paths. It ensures that all functions and subroutines work correctly and efficiently, helping to detect hidden errors that other methods might miss.

TABLE III
TABLE OF WHITE BOX TESTING

No	Action	Testing	Output	Result
1.	Open the system	Displaying the login page	Valid	
2.	Clicking the tourist data menu	Displaying the tourist data menu page	Valid	
3.	Clicking the AntCO menu	Displaying the AntCO menu page	Valid	

4.	Clicking the optimization result menu	Displaying the optimization result menu page	Valid	
5.	Clicking the user data menu	Displaying The user data page	Valid	

C. Manual Calculation of Ant Colony Optimization (ACO)

In this section, we present a manual calculation of the Ant Colony Optimization (ACO) algorithm to determine the optimal route from a starting terminal to several destination points. The input data includes the starting location, multiple destinations to be visited, and predefined ACO parameters. This manual calculation serves as a foundational illustration to enhance understanding of how the ACO algorithm operates in a concrete problem setting. It also allows researchers to verify algorithm logic and identify potential implementation issues during coding.

To streamline the explanation, a subset of five locations is used in the manual simulation. This simplification is intended solely for clarity and brevity in the journal's presentation. Importantly, the algorithmic steps, formulas, and parameter use are identical to those applied in the actual system testing using twenty tourist locations.

Input Data:

- 1) Starting Point: Terminal Type A Paya Ilang (T1)
- 2) Number of Locations: 5 Locations
- 3) ACO Parameters:
 - Number of ants (m) = 10
 - Alpha (α) = 1 (pheromone weight)
 - Beta (β) = 2 (distance/visibility weight)
 - Rho (ρ) = 0.5 (pheromone evaporation rate)

Calculation Steps:

1. Calculate Distance Between Points (Distance Matrix D): To calculate the distance between points, the Haversine formula is used as follows:

$$R = 6371 \text{ km (radius bumi)}$$

$$\Delta lat = lat_2 - lat_1 \text{ (dalam radian)}$$

$$\Delta lon = lon_2 - lon_1 \text{ (dalam radian)}$$

$$a = \sin^2\left(\frac{\Delta lat}{2}\right) + \cos(lat_1) \times \cos(lat_2) \times \sin^2\left(\frac{\Delta lon}{2}\right)$$

$$c = 2 \times \text{atan2}(\sqrt{a}, \sqrt{1-a})$$

$$d = R \times c$$

TABLE IV
DISTANCE CALCULATION RESULTS BETWEEN POINTS

	T1	T6	T2	T4	T7
T1	0	2.1	3.2	2.8	3.5
T6	2.1	0	1.8	1.5	2.2
T2	3.2	1.8	0	1.2	1.6
T4	2.8	1.5	1.2	0	1.4
T7	3.5	2.2	1.6	1.4	0

2. Initialize Pheromone Matrix (τ):

The pheromone matrix is initialized with the same value for each edge:

$$\tau[i][j] = 0.1 \text{ for all edges} \tag{1}$$

TABLE V
INITIAL PHEROMONE MATRIX

	T1	T6	T2	T4	T7
T1	0.1	0.1	0.1	0.1	0.1
T6	0.1	0.1	0.1	0.1	0.1
T2	0.1	0.1	0.1	0.1	0.1
T4	0.1	0.1	0.1	0.1	0.1
T7	0.1	0.1	0.1	0.1	0.1

3. Calculate Visibility (η):

Visibility is calculated using the formula:

$$\eta[i][j] = 1 / D[i][j] \tag{2}$$

Example calculation of visibility from T1 to other locations:

$$\eta[T1][T6] = 1 / 2.1 = 0.476$$

$$\eta[T1][T2] = 1 / 3.2 = 0.313$$

$$\eta[T1][T4] = 1 / 2.8 = 0.357$$

$$\eta[T1][T7] = 1 / 3.5 = 0.286$$

4. Calculate Probability:

The probability for selecting an edge is calculated using the formula:

$$p[i][j] = \frac{[\tau[i][j]]^\alpha \times [\eta[i][j]]^\beta}{\sum_k [\tau[i][k]]^\alpha \times [\eta[i][k]]^\beta} \tag{3}$$

Example calculations for the numerator of each edge from T1:

$$T1 \rightarrow T6: (0.1)^1 \times (0.476)^2 = 0.1 \times 0.227 = 0.0227$$

$$T1 \rightarrow T2: (0.1)^1 \times (0.313)^2 = 0.1 \times 0.098 = 0.0098$$

$$T1 \rightarrow T4: (0.1)^1 \times (0.357)^2 = 0.1 \times 0.127 = 0.0127$$

$$T1 \rightarrow T7: (0.1)^1 \times (0.286)^2 = 0.1 \times 0.082 = 0.0082$$

Summing the numerator:

$$\Sigma = 0.0227 + 0.0098 + 0.0127 + 0.0082 = 0.0534$$

Final probabilities:

$$p[T1][T6] = 0.0227 / 0.0534 = 0.425 \text{ (42.5\%)}$$

$$p[T1][T2] = 0.0098 / 0.0534 = 0.183 \text{ (18.3\%)}$$

$$p[T1][T4] = 0.0127 / 0.0534 = 0.238 \text{ (23.8\%)}$$

$$p[T1][T7] = 0.0082 / 0.0534 = 0.154 \text{ (15.4\%)}$$

5. Update Feromone:

After the ants complete their tour, pheromone update is performed using the formula:

$$\tau[i][j]_{new} = (1 - \rho)\tau[i][j]_{old} + \Delta\tau[i][j] \tag{4}$$

Dimana:

$$\Delta\tau[i][j] = \frac{Q}{L_k} \tag{5}$$

With $Q = 1$ (constant) and $L_k = 8.50 \text{ km}$ (tour length)

Example pheromone update calculation for edge T1 → T6:

$$\tau[T1][T6]_{new} = (1 - 0.5) \times (0.1) + 1 / 8.50$$

$$\tau[T1][T6]_{new} = 0.05 + 0.118 = 0.168$$

6. Final Result:

- 1) Optimal Route: T1 → T6 → T2 → T4 → T7
- 2) Total Distance: 8.50 km
- 3) Visit Order:
 - a) Terminal Type A Paya Ilang
 - b) Lancuk Leweng
 - c) Pasar Inpres Takengon
 - d) Bur Telege
 - e) Danau Lut Tawar

D. Computation Time and Memory Usage in the ACO Optimization Process

1) Computation Time in ACO

In the Ant Colony Optimization (ACO) algorithm, computation time is measured from the beginning to the end of the algorithm's execution process, which involves multiple iterations to construct an optimal path using artificial ants. This process can take longer depending on the number of ants used and the total number of iterations performed. Mathematically, the computation time can be calculated using the formula:

$$\text{Computation Time} = \text{End Time} - \text{Start Time} \quad (6)$$

Where:

Start Time is the time recorded before the algorithm begins. *End Time* is the time recorded after the algorithm completes. As an example, if the Start Time is 43265.123000 seconds and the End Time is 43265.147001 seconds, then the required computation time is:

$$\begin{aligned} \text{Computation Time} &= 43265.147001 - 43265.123000 \\ &= 0.024001 \text{ seconds} \end{aligned}$$

2) Memory Usage in ACO

In ACO: The ACO algorithm requires more memory because each ant needs to store the path it is constructing during the iterations. In addition, the pheromone matrix used to update the optimal path also requires significant storage. The more ants and iterations involved, the greater the memory required.

Mathematically, memory usage is calculated in the same way for both algorithms:

$$\text{Memory Usage (KB)} = \frac{\text{End Memory} - \text{Start Memory}}{1024} \quad (7)$$

Where:

Start Memory is the amount of memory used before the execution begins. *End Memory* is the amount of memory used after the execution is completed. An example of calculating memory usage is if the Start Memory is 1048576 byte and the End Memory is 1069296 bytes, then the memory usage is:

$$\begin{aligned} \text{Memory Usage (KB)} &= \frac{1069296 - 1048576}{1024} = \frac{20720}{1024} \\ &= 20,234375 \text{ KB} = 20,23 \text{ KB} \end{aligned}$$

IV. CONCLUSION

The developed system successfully achieves the research objective of designing and implementing an efficient shortest-route planner using the Ant Colony Optimization (ACO) algorithm for tourism in Takengon. The ACO implementation was able to generate an optimized visitation sequence across 20 tourist destinations with a total travel distance of 40.40 km and an estimated travel time of 81 minutes. The computation was completed in just 0.024001 seconds with a memory usage of only 20.23 KB, demonstrating high efficiency in both speed and resource consumption.

In addition to theoretical optimization, the system also provides turn-by-turn navigation based on actual road conditions, with a travel route spanning approximately 97.4 km and a duration of around 2 hours and 42 minutes. The algorithm was executed using 10 virtual ants with parameter values $\alpha = 1$, $\beta = 2$, and $\rho = 0.5$, achieving a balanced trade-off between convergence speed and solution quality.

Regarding scalability, although the current case study involved 20–22 locations, internal testing with synthetic datasets of 100 to 200 points showed that the system maintained practical execution time (approximately 2.3 seconds for 200 nodes) on a low-end laptop. This indicates that the system is feasible for larger-scale implementations, such as in big cities with denser tourist networks. ACO's ability to perform parallel exploration and pheromone-based pruning makes it well-suited for medium- to large-scale route optimization problems.

However, the system has several limitations. It does not support real-time data updates (e.g., live traffic, user movement, or dynamic environmental factors), and it is not yet integrated with dynamic mapping services like Google Maps API. Moreover, the system has not undergone extensive testing with end users, so user feedback in real-world scenarios is still needed to further enhance its usability and interface design, especially for tourists or travel operators.

The system is also highly adaptable to other regions. Since the route calculations are based on geographical coordinates (latitude and longitude), it can easily be customized for different areas by updating the location data. With minor interface adjustments, the system can be deployed to optimize tourist routes in other cities with similar geographic and logistical characteristics.

In conclusion, the results of this study confirm that ACO is a reliable, scalable, and flexible approach to route planning. The developed system is not only effective for a small tourist region like Takengon, but it also holds strong potential to be replicated and extended to larger cities or other tourism destinations with more complex routing challenges.

REFERENCES

- [1] J. J. Ihalauw and N. K. Tandafatu, "Geographical Information System for Indonesian Tourist Destinations," *IJISTECH (International J. Inf. Syst. Technol.)*, vol. 4, no. 2, p. 645, 2021, doi: 10.30645/ijistech.v4i2.105.
- [2] A. Nanda *et al.*, "Sistem Informasi Geografis Pariwisata Pantai di Kabupaten Kutai Kartanegara," vol. 25, no. 1, pp. 88–97, 2024.
- [3] Khaironi, "Kearifan Lokal Masyarakat Etnis Gayo sebagai Destinasi Wisata Budaya di Kota Takengon," *J. Educ. Soc. Stud.*, vol. 6, no. 2, pp. 99–110, 2016, [Online]. Available: <https://journal.unnes.ac.id/sju/index.php/jess/article/view/15601>
- [4] Wahyudi1, A. Aziz2, and Nani Ameliya3, "Tourist Perceptions of Tourism Object in Central Aceh," vol. 11, no. 2, pp. 1–8, 2023.
- [5] M. Nurdin, Fajriana, "Penentuan Lokasi Objek Wisata Di Aceh Tengah Dengan Menggunakan Metode Analytical Hierarchy Proses (Ahp).," vol. 15, no. 16, pp. 116–122, 2015.
- [6] M. Veluscek, T. Kalganova, and P. Broomhead, "Improving ant colony optimization performance through prediction of best termination condition," *Proc. IEEE Int. Conf. Ind. Technol.*, vol. 2015-June, no. June, pp. 2394–2402, 2015, doi: 10.1109/ICIT.2015.7125451.
- [7] I. G. S. Mas Diyasa, "Ant Colony Optimization To Determine the Shortest Route of Tourist Destinations in Bali : a Case Study," *J. Ilm. Kursor*, vol. 11, no. 3, p. 131, 2022, doi: 10.21107/kursor.v11i3.279.
- [8] Nurdin, Taufiq, and Fajriana, "Searching the shortest route for distribution of LPG in Medan city using ant colony algorithm," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 725, no. 1, pp. 0–9, 2020, doi: 10.1088/1757-899X/725/1/012121.
- [9] A. A. Fajrin and D. Meldra, "Optimasi Rute Panduan Informasi Lokasi Wisata Menggunakan Ant Colony System Pada Kota Batam," *J. Teknol. Dan Open Source*, vol. 2, no. 2, pp. 1–13, 2019, doi: 10.36378/jtos.v2i2.353.
- [10] Z. Anshory, "Penerapan Algoritma Ant Colony Optimization Pada Aplikasi Pemandu Wisata Provinsi Sumatera Utara Berbasis Android," *J. Comput. Syst. Informatics*, vol. 1, no. 2, pp. 61–67, 2020, [Online]. Available: <https://ejournal.seminar-id.com/index.php/josyc/article/view/106>
- [11] P. A. Longley and M. Frank Goodchild, "Geographic Information Science and Systems. In International Encyclopedia of Human Geography, Second Edition." pp. 29–36, 2019. doi: 10.1016/B978-0-08-102295-5.10557-8.
- [12] L. Sylvanus and A. Leo, "Perancangan dan Analisa Sistem Informasi Geografis Objek Wisata Jabodetabek Berbasis Web," vol. 7, no. 2, 2024, doi: 10.32877/bt.v7i2.1755.
- [13] J. Jtik, J. Teknologi, I. A. Marleni, and A. Gunaryati, "Presensi Karyawan Berbasis Web dengan Fitur Lokasi Leaflet JS menggunakan Laravel," vol. 7, no. 3, pp. 1–7, 2023.
- [14] S. Rahmayuda and C. Suhery, "Pemanfaatan Leaflet Javascript Sebagai Platform Pengembangan Sistem Informasi Geografis Aset Pemerintah," vol. 5, no. 01, pp. 26–37, 2021.
- [15] V. Risqiyanti, H. Yasin, and R. Santoso, "Pencarian Jalur Terpendek Menggunakan Metode Algoritma 'Ant Colony Optimization' Pada GUI Matlab (Studi Kasus: PT Distriversa Buana Mas cabang Purwokerto)," *J. Gaussian*, vol. 8, no. 2, pp. 272–284, 2019, doi: 10.14710/j.gauss.v8i2.26671.
- [16] W. Maharani, "Analisis Algoritma Hybrid Ant Colony Optimization (Aco) Dan Local Search Untuk Optimasi Pemotongan Bahan Baku," vol. 2009, no. Snati, 2009.
- [17] A. K. Nugroho *et al.*, "Ant Colony Optimization Untuk Menyeleksi Fitur Dan Klasifikasi Artikel," vol. 10, no. 1, pp. 223–232, 2019.
- [18] D. Udjulawa and S. Oktarina, "Penerapan Algoritma Ant Colony Optimization Untuk Pencarian Rute Terpendek Lokasi Wisata (Studi Kasus Wisata Di Kota Palembang)," vol. 3, no. 1, pp. 26–33, 2022.
- [19] C. Blum, "Ant colony optimization: Introduction and recent trends," vol. 2, pp. 353–373, 2005, doi: 10.1016/j.plev.2005.10.001.
- [20] M. Ihsan *et al.*, "Implementasi Algoritma Ant Colony Optimization Thriving Di Kota Malang," vol. 1, no. 1, pp. 1–8, 2019.
- [21] G. N. Rafi, A. Voutama, N. Heryana, and U. S. Karawang, "Model Unified Language Dalam Perencanaan Pembuatan Diagnosis Web," vol. 12, no. 1, 2024.
- [22] R. Supriyadi, N. Maulidah, H. Nalatissifa, A. Fauzi, and S. Diantika, "Perancangan Sistem Informasi Rental Mobil Berbasis Website pada Rentalin Aja," vol. 10, no. 2, pp. 156–165, 2024.
- [23] P. Studi, S. Informasi, F. Teknologi, and U. Battuta, "Pemodelan Sistem Penerimaan Anggota Baru dengan Unified Modeling Language (UML) (Studi Kasus : Programmer Association of Battuta)," vol. 12, pp. 1514–1521, 2023.
- [24] N. Musthofa and M. A. Adiguna, "Perancangan Aplikasi E-Commerce Spare-Part Komputer Berbasis Web Menggunakan CodeIgniter Pada Dhamar Putra Ccomputer Kota Tangerang," vol. 1, no. 03, pp. 199–207, 2022.
- [25] S. W. Ramdany, S. A. Kaidar, B. Aguchino, C. Amelia, and A. Putri, "Penerapan UML Class Diagram dalam Perancangan Sistem Informasi Perpustakaan Berbasis Web," vol. 5, no. 1.