Accredited SINTA 2 Ranking
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online at: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 9 No. 1 (2025) 153 - 165 e-ISSN: 2580-0760

Ant Colony Optimization for Jakarta Historical Tours: A Comparative Analysis of GPS and Map Image Approaches

Gabriel Fortino Bodhi[1]*, Charleen[2], Devi Fitrianah[3]
[1,2]Computer Science, Binus Graduate Program, Bina Nusantara University, Jakarta, Indonesia
[3]Bina Nusantara Binus Graduate Program, University, Jakarta, Indonesia
[1]gabriel.bodhi@binus.ac.id, [2]charleen@binus.ac.id, [3]devi.fitrianah@binus.ac.id

Abstract

The Traveling Salesman Problem (TSP) is a problem that represents a difficult combinatorial optimization problem starting from practical problems. The ant colony optimization (ACO) algorithm is implemented in several topics, particularly in solving combinatorial optimization problems. ACO is inspired by the behavior of ants in searching for the shortest path between a food source and their nest. In this research, ACO is used to find the best path or traveling salesman problem for museums and historical sites in Jakarta capital city of Indonesia. This research employs an approach based on the location's coordinates or latitude and longitude, while another method depends on coordinate data obtained from a supplied map image. After implementing both models, it can be concluded that the ACO model is not very good at solving TSP using actual coordinates. Meanwhile, the algorithm can quickly find near-optimal paths when using coordinates from a map image. The algorithm generates the optimal path in 11 seconds, reducing the initial distance from 17.938 to 4.430, using 4.731 ants and 75 trips with a distance power of 1. Statistical random variation was also performed, which proved that the algorithm is flexible and reliable when tested under various conditions.

Keywords: Ant Colony Optimization; Historical Sites; Intelligent System; Traveling Salesman Problem; Tour Planning

How to Cite: G. Bodhi, Charleen, and D. Fitrianah, "Ant Colony Optimization for Jakarta Historical Tours: A Comparative Analysis of GPS and Map Image Approaches", *J. RESTI (Rekayasa Sist. Teknol. Inf.)*, vol. 9, no. 1, pp. 153 - 165, Feb. 2025..
DOI: https://doi.org/10.29207/resti.v9i1.5968

1. Introduction

In metropolitan cities like Jakarta, where historical significance is preserved in numerous museums and monuments, tourists and history enthusiasts often face the challenge of planning efficient routes to visit multiple sites. This challenge becomes particularly complex when the aim is to minimize travel time while maximizing the number of locations visited. Given the large number of museums spread across the city, determining the optimal route is a combinatorial problem that can overwhelm manual planning methods. The problem of visiting all desired locations in the shortest possible time mirrors the well-known Traveling Salesman Problem (TSP).

Optimization problems involve solving problems by minimizing or maximizing the function of the problem [1]. Each function can yield many solutions. In optimization problems, the aim is to find the most optimal result while keeping certain constraints in mind. An optimization problem is defined as a set of parameters, and with the right parameters, we can determine the best solution [2]. The Traveling Salesman Problem is one of the problems that has drawn the attention of many mathematician' experts, with a method that is easy to explain but difficult to solve. TSP represents many difficult combinatorial optimization problems, ranging from practical problems that can be formulated as TSP problems. The most important and often occurring sub-problem in the traveling salesman problem is the routing problem, which requires finding the fastest sequence of points.

In an ideal situation, tourists would be able to visit all their chosen museums with minimal travel time, navigating through the city efficiently without needing to backtrack or waste time on unnecessary detours. However, achieving this efficiency manually is difficult due to the city's traffic patterns and the scattered nature of historical sites. Therefore, a system that can intelligently suggest the shortest route would greatly enhance the tourist experience.

Received: 05-08-2024 | Accepted: 14-02-2025 | Published Online: 20-02-2025

Intelligent systems can alleviate human tasks in various aspects. These intelligent systems have smart mechanisms with various functions, including information processing, decision-making, problem-solving, pattern recognition, data analysis, and many more [3]. Optimization problems can be easily addressed using various available methods. The optimization conducted can effectively solve the Traveling Salesman Problem (TSP) by selecting from a range of algorithms, such as genetic algorithms, ant-colony algorithms, simulated annealing algorithms, and many others.

TSP itself is a graph that has several attributes in problem-solving, including nodes, node edges, and different corner points depending on the case [4]. Each city must be passed through at least once. Generally, the length of the path represents the solution cost or can be said to be the total distance that the salesman must travel. The relevance of heuristic and bio-inspired solvers for such problems comes from their ability to find solutions that are acceptable in terms of time/performance balance [2]. As for the corner points, they represent the names of the places to be visited. Node edges can be used to calculate the distance travelled from point A to point B in the route.

The running time of the exact TSP algorithm increases exponentially with the number of cities. TSP researchers have proposed various methods and algorithms. Researchers initially focused on specific methods, such as integer linear programming, dynamic programming, and graph algorithms. TSP can be solved using exact, heuristic, or meta-heuristic methods [5]. Exact algorithms can find the best solution. However, they are prone to getting trapped in a search space explosion of combinations. As a result, many researchers use heuristic or approximation algorithms for TSP. Unlike exact methods, heuristic algorithms can find satisfactory or nearly optimal solutions within reasonable computational time [6]. The Ant Colony Optimization algorithm is one of the meta-heuristic methods for solving the TSP [5].

Ant colony optimization is a new metaphor for solving optimization problems and has been widely applied. ACO is a metaheuristic based on the behaviour of biological ants proposed by Marco Dorigo in 1991 [7]. The ACO algorithm was first inspired by pheromone laying and the selection of the shortest route using pheromones. Since its first discovery, many researchers have worked and published their findings in this field. Although the initial results were not promising, recent developments have elevated this metaheuristic to the status of a significant algorithm in Swarm Intelligence.

Indonesia, particularly Jakarta, is a place of great historical significance, which has led to the establishment of numerous museums and historical monuments. With the abundance of museums in Jakarta, a problem arises for history enthusiasts when they want to visit various museums. The Ant Colony Optimization method will be used to solve the traveling salesman problem for museums and historical sites in Jakarta, as it can select the most efficient route.

In several studies, it can be found that the ant colony optimization (ACO) algorithm has been implemented in various topics. ACO has been successfully used in several problems, especially in solving combinatorial optimization problems. ACO has high capabilities in optimization problems, but there are still some shortcomings, including stagnation behaviour, long computation time, and premature convergence problems in the basic ACO algorithm [8].

ACO itself is inspired by a living creature, ants, in the process of finding the shortest path between a food source and their ant nest without using visual communication [9]. Artificial ants in ACO are stochastic solution construction procedures that probabilistically build a solution by iteratively adding solution components to partial solutions while taking heuristic information about the problem instance being solved and (artificial) pheromone trails that change dynamically at run-time to reflect the ants' acquired search experience into account [10].

Ants communicate indirectly by using a chemical compound called pheromone. The process starts when ants are foraging for food, and during this process, ants drop pheromone compounds along the path they take from the nest. The pheromone compounds are dropped until the ants return to the nest after taking the food from the food source. The shortest distance will contain a stronger pheromone compound, which will make other ants choose that path.

Pheromones are a key component of the algorithm used to guide ants towards promising solutions in ACO. Pheromones are chemical substances stored on the ground by ants as they move through solution space. The pheromone trail is updated at each iteration of the algorithm based on the quality of the ant's solution. Pheromone updates are typically performed using local or global update rules, which determine how much pheromone is stored by ants along their path. Local update rules only update the pheromone trail on one ant's path, while global update rules update the pheromone trail on all ants' paths. The pheromone trail fades over time, preventing the algorithm from becoming trapped in local optima. This evaporation process allows the algorithm to explore new areas of solution space and eventually converge to a solution that is close to optimal. Overall, the use of pheromones in ACO allows the algorithm to efficiently search solution space and quickly converge to a solution that is close to optimal.

ACO is widely known has advantages in satisfactory robustness, distributed parallel processing, and seamless integration with other algorithms [11]. Although ACO can be utilized in a variety of areas, it was first introduced to TSP through a proof-of-concept application [12]. In this study, we propose an ACO

algorithm for solving TSP to determine the distance between museums and historical monuments in Jakarta, to facilitate tourists and history enthusiasts in determining the most effective destination route to minimize the time used. Each historical place will be used as a node or destination for the subject. In addition to making the time used more effective, the distance travelled to visit historical museums in Jakarta will be shorter. Nevertheless, conventional ant colony optimization (ACO) suffers from limitations such as slow convergence and low efficiency, necessitating the need for algorithmic modifications [13].

There are several ACO algorithms developed in various literature studies, especially in solving TSP. In each step of ACO, ants use the pseudorandom-proportional action choice rule in selecting the next destination (museum). When at location i, ant k will choose museum j based on the Equation 1.

$$J = arg\,max_{v \varepsilon allowed_k(i)}\{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}\}\ if\ q < q_0 \quad (1)$$

q is a uniformly distributed random variable in [0,1]. $q_0 (0 \leq q_0 \leq 1)$ is a predetermined pheromone trail parameter. η_ij= 1/d_ij $\eta_{ij} = 1/d_{ij}$ is heuristic information, where d_{ij} is the distance between museum i and museum j, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are two adjustable positive parameters that regulate the magnitude of pheromone trail and heuristic information, and $allowed_k(i)$ is a museum that has not yet been visited by Ant K when the ant is at Museum i. J is selected according to the transition probability selected by Equation 2.

$$P_y^k(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \epsilon J_k(i)}[\tau_{il}(t)]^{\alpha}[\eta_{il}]^{\beta}}\ if\ j \epsilon J_k(i)$$

$$, otherwise\ P_y^k(t) = 0 \quad (1)$$

To improve future equations, the pheromone trail needs to be updated to reflect the performance of ants and the quality of solutions found. There are two processes for updating pheromone trails, namely local and global. For local updates, each ant that has chosen a city will update the amount of pheromone on each edge according to Equation 3.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho.\tau_0 \quad (2)$$

$0 < \rho \leq 1$ is a parameter *decay*, $\tau_0 = 1/n.L_{nn}$"is the initial value of the pheromone trail, where n is the number of museums in TSP and L_{nn} is the cost incurred by the nearest heuristic neighbour. After the angle between museum i and j have been visited by all ants, the local update rule reduces the level of pheromone at the angle. Therefore, the effect of the local update rule is to make the angle less attractive to the next ants.

For global updates, after all ants have explored all museums, the amount of pheromone is only updated on the optimal path by Equation 4.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \quad (4)$$

In Equation 5,

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m}\Delta\tau_{ij}^k(t) \quad (5)$$

Intelligent systems have been widely used for various purposes, such as information processing, decision-making, problem-solving, and many more [3]. After being processed by intelligent systems, technology has been created that greatly assists human work in this modern era. There is an intelligent system-based technology for detecting the development and evaluating COVID-19 using deep neural networks [14]. With the technology introduced in research [14]. COVID-19 can be detected more quickly and easily by considering various parameters. There is also an intelligent system technology through a location-based approach that is used to enhance more efficient airport infrastructure management [15]

The Ant Colony Algorithm has been widely applied for route planning and produces good solutions with fewer parameters than other algorithms. This algorithm takes a long time to search and easily falls into local optima, stagnation, or deadlock conditions. Many experts have tried to solve this problem. Y. Wang. et al proposed the ACO algorithm to solve vehicle routing problems. The usefulness of its implementation is to reduce the number of vehicles and reduce the delivery distance between warehouses and customers [16].

The research [17] evaluates the effectiveness of ACO procedures in improving the reuse of information. State-of-the-art ACO for static optimization (MAX-MIN Ant System, MMAS) and the most relevant ACO algorithm proposed for dynamic optimization (P-ACO) are used. A variant of TSP with dynamic demands (DTSP) is used as the testing benchmark. This work is important for three reasons: it acknowledges that DCOP requires specially configured parameter settings, directly compares MMAS and P-ACO, isolates local search as an experimental factor, and conducts experimental investigation on the specific DCOP component proposed for ACO. Results show that the component contributes very little to performance when algorithms are allowed to use local search but is highly effective when there is none. Research [18] presents an improved Ant Colony Optimization (ACO) algorithm for solving the Travelling Salesman Problem. The algorithm uses three novel techniques to enhance performance, reduce execution time, and address issues associated with ACO-based methods. These techniques include clustering of nodes, adaptive pheromone evaporation, and a new termination condition. Experimental results show that the proposed algorithm outperforms other ACO-based methods in most cases. The impact of these techniques on the algorithm's behavior is thoroughly analyzed and discussed.

In research [19], an ACO-based method called ACO with Adaptive Visibility (ACOAV) is proposed, which intelligently adopts a general formula of heuristic visibility related to the final destination city. It uses a new distance metric that incorporates proximity and the final destination to select the next city, reducing the tour

cost. ACOAV is evaluated on a set of 35 benchmark TSP instances and compared rigorously with ACO. It was found to be the best for 29 TSP instances out of 35 instances, with optimal solutions achieved in 22 instances. Statistical tests comparing the performance revealed the significance of the proposed ACOAV compared to bio-inspired methods.

Although ACO is a natural TSP solving algorithm, it also has some drawbacks in its solving process such as slow convergence speed and prone to falling into local optima. Therefore, research [20] proposes an improved ant colony optimization based on graph convolutional network: Graph Convolutional Network Improved Ant Colony Optimization (GCNIACO). GCNIACO is introduced to produce better solutions, and better solutions are converted into pheromones on the ACO initial path. The algorithm's ability to jump out of local optima is improved, and the optimization results are compared with other classic algorithms.

According to research from P. Du, N. Liu, H. Zhang, and J. Lu in paper [21], the paper proposes an improved ACO algorithm called AHACO for solving the TSP. AHACO incorporates three enhancements including city classification using k-means, a modified 2-opt local optimizer, and a mechanism to escape local optima. Experimental results show that AHACO outperforms other algorithms, achieving an average solution quality improvement of 83.33% and superior performance for large-scale TSP instances.

Research [22] proposed a hybrid approach using the advanced sine-cosine algorithm (ASCA) and the advanced ant colony optimization (AACO) technique to achieve optimal path search and control for multiple mobile robots in both static and dynamic unknown environments. In research [23], an improved ACO algorithm based on particle swarm optimization developed to determine the optimal path for an AUV to reach its designated destination in a complex underwater environment for autonomous underwater vehicle. Then, R. Behmanesh and I. Rahimi address the multi-resource job shop scheduling problem (MRJSP) with resource flexibility aimed at minimizing makespan, using a mixed integer linear programming (MILP) model and an enhanced ant colony optimization (ACO) algorithm incorporating a pheromone update strategy inspired by selfish herd (SH) theory [24].

Research [25] proposed parallelization strategies for GPU-Based ACO solving the TSP. Parallelization of ACO is necessary for larger TSP instances due to the high number of calculations involved. Many parallel approaches have been proposed for ACO, particularly for modern hardware like GPUs. This paper compares and analyzes the performance of ACO implementations with different parallelization strategies for solving TSP instances of varying sizes. The results show that there is no overall best strategy and highlight the importance of factors like GPU occupancy and workload distribution among threads in reducing execution time. Unlike for research [26], this paper presents a new variant called Focused ACO (FACO) that improves performance by controlling differences between solutions and integrating with problem-specific local search. Computational studies show that FACO outperforms state-of-the-art ACOs for large Traveling Salesman Problem (TSP) instances, finding high-quality solutions in less than an hour on an 8-core CPU.

The ACO algorithm was one of the first nature-inspired algorithms and it mimicked actual ant colony behaviors [27]. According to research on ant behavior in the wild, ants can find the shortest path between their nest and a food source. The evaporation rate, chemical matter of pheromone that ants drop on the route that they have chosen, is the most important feature in seeking the shortest path. Ants in a colony will typically choose a path with a high pheromone rate [27]. The algorithm consists of a number of iterations. In each iteration, several ants build a complete solution using heuristic information and experience collected from previous ant populations [28]. Heuristic information is critical in the metaphase and telophase, guaranteeing that ACO converges with good performance [29]. The collected experiences are represented using pheromone trails, which are stored on solution components. Pheromones can be stored on components and/or connections in the solution depending on the problem being solved. The ACO algorithm simulates optimization in route search. The ACO algorithm procedure will be illustrated in Figure 1.

Research [30] developed improved heuristic mechanism ACO (IHMACO) overcomes the limitations of traditional ACO by integrating adaptive pheromone concentration, directional judgment, an enhanced pseudo-random transfer strategy, and dynamic pheromone evaporation adjustment, leading to more effective path planning and confirmed performance through experimental validation. In the wild, ants search for the shortest path between their nest and food source. They leave a chemical trail, known as a pheromone, along the path so that other ants can easily find the food source. Furthermore, ants reinforce this pheromone trail as they return to their nest, although the pheromone evaporates over time, reducing its influence on other ants. The density of pheromones along the tour is determined by the tour selection time, especially the last one. Other ants can find the shortest route by taking the path with the highest pheromone density [31].

By addressing these challenges, this study contributes a refined version of the ACO algorithm tailored specifically to the problem of tourist route optimization in Jakarta. The goal is to minimize travel distances between museums, thereby saving time and improving the overall tourist experience. The proposed approach will build on the foundational work in ACO-based optimization while addressing its limitations to ensure robust performance in real-world applications. Through this work, we aim to offer a practical solution that helps

visitors explore Jakarta's rich historical offerings with ease and efficiency.

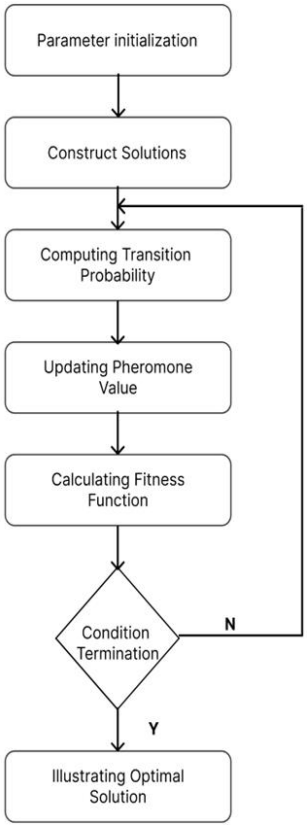

Figure 1. ACO Algorithm Flowchart

2. Research Methods

The stages of this research are as follows: 1) preliminary study; 2) data gathering and pre-processing, 3) model construction, 4) evaluation, and 5) result analysis.

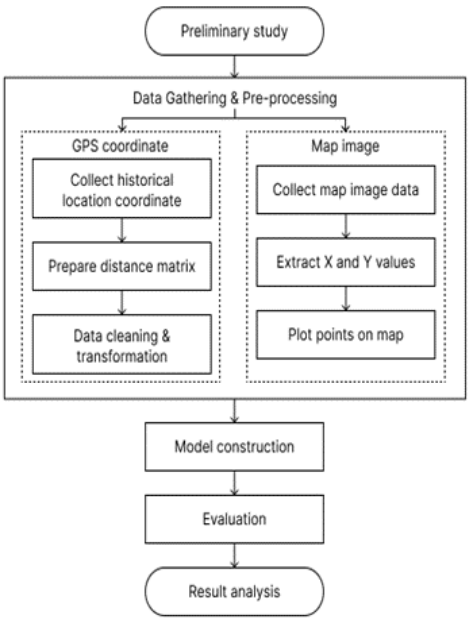

Figure 2. Proposed Research Methodology

Figure 2 represents the proposed research methodology flowchart. The preliminary study aims to gain a full understanding of the use of ACO to solve TSP, which is the problem that has been selected for this research. In the second d stage, data will be collected and processed so the data is suitable for a machine learning model. The two ACO models that were used to solve TSP will then be compared to evaluate their algorithm reliability in finding the best route available.

2.1 Preliminary Study

The first step of this research is to first investigate a research including the literature review. The purpose of a literature review is to identify the previous research, clarify foundational theories, and explain the methodology used in this research.

Journal articles related to the use of ant colony optimization to solve the traveling salesman problem will be searched and reviewed as part of the literature study.

2.2 Data Gathering and Pre-Processing

This paper used two types of ACO algorithms to solve the traveling salesman problem. The first one uses the original coordinates of historical locations, while the second one uses the coordinates of points on a map. The first and final point used is the Soekarno-Hatta Airport. For the first algorithm, the dataset with .csv extension consist of 53 drop-point with all the data belonging to cluster 0, and each data contains information about longitude and latitude. The tracking_id is a unique id that is used for each row of data. Table 1 is a list of locations used and their coordinates.

Table 1. List of Museum and Historical Locations in Jakarta

No	Name	Address
1	Museum Nasional / Museum Gajah	Jl. Medan Merdeka Barat No.12, Gambir, Kota Jakarta Pusat
2	Museum Bank Indonesia	Jalan Jembatan Batu No.3, Pinangsia, Tamansari, RT.3/RW.6, Pinangsia, Tamansari, Kota Jakarta Barat
3	Museum Fatahillah/Museum Sejarah jakarta	Jalan Taman Fatahillah No.1, Pinangsia, Tamansari, Kota Jakarta Barat
4	Museum Bank Mandiri	Jl. Lapangan Stasiun No. 1, RT. 3 / RW. 6, Pinangsia, Tamansari, RT.3/RW.6, Pinangsia, Tamansari, Kota Jakarta Barat
5	Museum Wayang	Jl. Lada No.23, RT.7/RW.7, Pinangsia, Tamansari, Kota Jakarta Barat

No	Name	Address
6	Museum Keramik	Jl. Pos Kota, RT.9/RW.2, Daerah Khusus Ibukota Jakarta, Jakarta Barat
7	Museum Polri	Jl. Trunojoyo No.3, RT.5/RW.2, Selong, Kby. Baru, Kota Jakarta Selatan
8	Museum Layang - Layang	Jalan Haji Kamang No.19, RT.2/RW.1, Pondok Labu, Cilandak, RT.2/RW.10, Pd. Labu, Cilandak, Kota Jakarta Selatan
9	Museum Taman Prasasti	Jl. Tanah Abang I No.1, Kelurahan Petojo Selatan, Kecamatan Gambir, RT.11/RW.8, Petojo Selatan, Gambir, RT.11/RW.8, Petojo Sel., Gambir, Kota Jakarta Pusat
10	Museum Tekstil	Jl. Aipda KS Tubun No.2-4 Jakarta Pusat
11	Museum Kebangkitan Nasional	Jl. Abdul Rachman Saleh No.26, RT.4/RW.5, Senen, Kota Jakarta Pusat
12	Monumen Nasional	Jl. Tugu Monas, Gambir, Jakarta Pusat, DKI Jakarta
13	Museum katedral	Jl. Katedral No.7B, Ps. Baru, Sawah Besar, Kota Jakarta Pusat
14	Museum Perumusan Naskah Proklamasi	Jl. Imam Bonjol No.15, RT.9/RW.4, Menteng, Kota Jakarta Pusat
15	Museum Sumpah Pemuda	Jalan Kramat Raya No.106, Kwitang, Senen, RT.2/RW.9, Kwitang, Senen, Kota Jakarta Pusat
16	Museum Satria Mandala	Jl. Jend. Gatot Subroto No.14, RT.6/RW.1, Kuningan Bar., Mampang Prpt., Kota Jakarta Selatan
17	Museum Bahari	Jl. Pasar Ikan No.1, RT.11/RW.4, Penjaringan, Kota Jakarta Utara
18	Museum Joang 45	Jl. Menteng Raya No.31, RT.1/RW.10, Kb. Sirih, Menteng, Kota Jakarta Pusat
19	Taman Mini Indonesia Indah	Jakarta Timur
20	Museum Antara	Antara 61, Kelurahan Pasar Baru, Kecamatan Sawah Besar, Jakarta Pusat, DKI Jakarta
21	Museum Jendaral Besar Dr. A.H Nasution	Jl Teuku Umar No. 40, Jakarta Pusat
22	Museum Mohammad Hoesni Thamrin	jl. Kenari II No. 15
23	Sasmita loka Pahlawan Revolusi Jendral Ahmad Yani	Jl. Lembang No.58 dan jl Laruharhari No. 65, Jakarta Pusat
24	Galeri Nasional Indonesia	Jl. Medan Merdeka Timur No. 14, Jakarta Pusat
25	Museum Santa Maria Juanda	Jl. Ir. H. Juanda No.29, Kebon Kelapa, Gambir, Jakarta Pusat
26	Bible Museum & Library	Lantai 2-3 Gedung Pusat Alkitab Jalan Salemba Raya No.12 2, RT.2/RW.6, 6, Paseban, Senen, Central Jakarta City, Jakarta 10430
27	Museum DPR RI	Jl. Jendral Gatot Subroto, Senayan, Jakarta Pusat
28	Museum Kehutanan “Ir. Djamaludin Suryohadikusumo”	Kompleks Manggala Wanabakti, RT.01/03, Gelora, Tanahabang, Jakarta Pusat
29	Museum Istiqlal	Jl. Raya Taman Mini Indonesia Indah Pintu 1 Jakarta Timur
30	Monumen Pancasila Sakti	Jl. Raya Pondok Gede RT.4/RW.12, Lubang Buaya, Cipayung, Jakarta Timur
31	Museum Anatomi Universitas Katolik (Unika) Atmajaya	Gedung Klara Asisi Lantai 2, Fakultas Kedokteran Universitas Atma Jaya, Jl. Pluit Raya No. 2, Jakarta Utara

No	Name	Address
32	Museum Basoeki Abdullah	Jl. Keuangan Raya No. 19 Cilandak Barat, Jakarta Selatan
33	Museum Benda-Benda Alkitab Yerushalayim	Jl. Raya Rawa Sengon No. 35 RT.001 / RW.022, Kelapa Gading Barat, Kelapa Gading, Kota Jakarta Utara, Daerah Khusus Ibukota Jakarta
34	Museum BNI 1946	Jl. Lada No.1, Kota Tua, Pinangsia, Tamansari, Jakarta Barat
35	Museum Ciputra Artpreneur	Jl. Prof. DR. Satrio Kav. 3-5, Ciputra World 1, Retail Podium Level 11-13, Kuningan, Jakarta Selatan
36	Museum Purna Bhakti Pertiwi	Jl. Taman Mini Indonesia Indah, RW.3, Pinang Ranti, Kec. Makasar, Kota Jakarta Timur
37	Museum di Tengah Kebun	Jl. Kemang Timur Raya No.66, RT.7/RW.3, Bangka, Mampang Prapatan, Jakarta Selatan
38	Museum Komodo dan Taman Reptil	Taman Mini Indonesia Indah, Jl. Raya Taman Mini, Jakarta Timur
39	Museum FKUI-Indonesia	Jl. Salemba Raya No. 6 Kenari, Senen, Jakarta Pusat
40	Museum Graha Widya Patra	Taman Mini Indonesia Indah, Jl. Raya Taman Mini, Jakarta Timur
41	Museum Hakka Indonesia	Taman Mini Indonesia Indah, Jl. Raya Taman Mini, Jakarta Timur
42	Museum Harry Darsono	Jl. Cilandak Tengah No.71, RT.2/RW.13, Cilandak Barat, Cilandak, Jakarta Selatan
43	Museum Indonesia	Taman Mini Indonesia Indah, Jl. Raya Taman Mini, Jakarta Timur
44	Museum Keprajuritan	Taman Mini Indonesia Indah, Jl. Raya Taman Mini, Jakarta Timur
45	Museum Listrik dan Energi Baru	Taman Mini Indonesia Indah, Jl. Raya Taman Mini, Jakarta Timur
46	Museum MACAN (Modern and Contemporary Art in Nusantara)	AKR Tower Level M, Jl. Panjang No. 5 Kebon Jeruk, Jakarta Barat
47	Museum Olahraga Nasional	Taman Mini Indonesia Indah, Jl. Raya Taman Mini, Jakarta Timur
48	Museum Penerangan	Taman Mini Indonesia Indah, Jl. Raya Taman Mini, Jakarta Timur
49	Museum Perangko Indonesia	Taman Mini Indonesia Indah, Jl. Raya Taman Mini, Jakarta Timur
50	Museum Pusaka	Taman Mini Indonesia Indah, Jl. Raya Taman Mini, Jakarta Timur
51	Museum Serangga	Taman Mini Indonesia Indah, Jl. Raya Taman Mini, Jakarta Timur
52	Museum Tragedi 12 Mei '98 Universitas Trisakti	Jl. Kyai Tapa No.1, RT.6/RW.16, Tomang, Grogol Petamburan, Kota Jakarta Barat, DKI Jakarta
53	Museum Transportasi	Taman Mini Indonesia Indah, Jl. Raya Taman Mini, Jakarta Timur

For the second algorithm, the location points of each historical location are collected, and then the x and y values based on the image are collected into one array. Next, the plotting process will be executed for the TSP points on the map of Jakarta that has been imported as shown in Figure 3.

Data pre-processing in this research was carried out in several stages. The first stage, data cleaning, involves removing missing or inconsistent data that may cause the Ant Colony Optimization (ACO) algorithm to malfunction. This process includes tasks such as data imputation, standardization, or normalization. Following data cleaning, data transformation is conducted, which entails converting the input data into a format that can be utilized by the ACO algorithm. Specifically, for the Traveling Salesman Problem (TSP), the input data, consisting of a collection of cities

and their coordinates, is transformed into a structure that the ACO algorithm can effectively use.

Overall, data pre-processing is an important step in any optimization algorithm, including ACO, because it can have a major impact on the performance of the algorithm and the quality of the solutions. It is possible to improve the efficiency and effectiveness of the ACO algorithm and generate better solutions in a shorter amount of time by carefully performing data pre-processing.

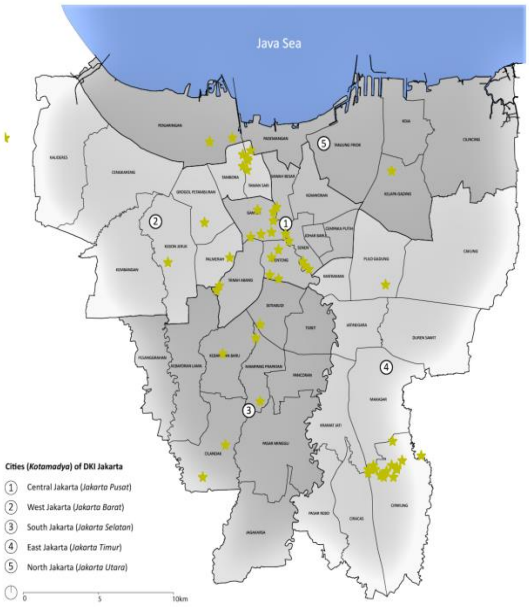

Figure 3. Map of Jakarta with Plotting of Location Points

2.3 Model Construction

This paper applies the ACO algorithm through two approaches. The first approach involves experimenting with ACO using GPS coordinates, and the second approach involves experimenting with ACO using map images. To increase efficiency for the first approach, this algorithm is used to optimize last-mile distribution routes by reducing the driver's travel distance. This study tackles the TSP problem by combining population-based ACO methods with a new hierarchical pheromone update to improve solution quality.

The algorithm will first read the data and load it into the pandas dataframe, after which the data will be grouped by cluster and the amount of data in each cluster will be calculated. Then, the algorithm will prepare a distance matrix based on the haversine of the latitude and longitude data. We use haversine to get the distance between geo-locations. Then the results will be displayed in kilometres.

The haversine formula is used to calculate the distance between two points on a sphere, such as the Earth. The formula takes into account the radius of the sphere and the latitude and longitude coordinates of the two points. The law of haversines states that the sides of a spherical triangle are proportional to the sine of the opposite angles. The Haversine formula is widely used in applications such as navigation systems, geographic information systems (GIS), and location-based services that involve calculating distances between points on the earth's surface.

The core process of this algorithm is ACO which tries to get the best solution. First of all, the parameters nk, maxIteration, beta, zeta, rho, q0, plot, and verbose are assigned values. After that, the optimizer object is created from the ACO class, and the fit method is used to get the best result and path. Finally, the convergence matrix of the optimization process is plotted.

A convergence matrix is a table or plot showing the progress of an optimization algorithm over time, usually in relation to the objective function. The convergence matrix usually contains information such as the iteration number, the objective function value, and the termination criteria used to stop the algorithm. The convergence matrix can be used to evaluate the performance of the algorithm and determine whether it has reached a satisfactory solution. A good optimization algorithm should, in general, converge to a solution quickly and efficiently, with objective function values that are close to the optimal value.

Convergence matrices can also be used to diagnose algorithm problems such as slow convergence or oscillations in the objective function. This data can be used to improve the algorithm or fine-tune its parameters to improve performance. Overall, the convergence matrix is an effective tool for analyzing and evaluating the performance of optimization algorithms and is frequently used in research papers and engineering applications.

For the second approach, ACO can be applied to TSP with the city location shown on the map. In such cases, the map or image can be converted into a graphical structure that can be used as input to the ACO algorithm. The basic idea is to represent each city or location on the map as a node in a graph and connect the nodes with edges that represent the distance between them. The ACO algorithm will then use this graph structure as input to find the shortest or most optimal path through the graph.

The logic of the algorithm used is that we first send ants out on random walks to complete the Traveling Salesman task for themselves. The ant_count parameter in the numpy array determines how many ants there are, and the ant_speed parameter determines how many steps each ant takes per epoch. Individual ants will keep an eye on a list of cities they have visited along their "path", a list of "remaining" locations they still need to see, the "distance" to the next location, the number of "round_trips" they have made to the colony, and the "path_cost" of how many steps they have taken. There is a "distance" vector countdown as the ant moves.

When an ant reaches a node, it randomly selects a new location from a list of choices left by other ants. This

decision is weighed by the amount of pheromone left behind by other ants. Pheromone_power determines how strongly the ant is affected by small differences in the pheromone. Distance_power determines whether ants prioritize visiting closer nodes first. Reward_power determines how the best/current path is used when leaving a new pheromone. This parameter encourages the ants to explore longer paths around the path of the strongest pheromone. Decay_power determines how quickly old pheromone trails decay; after an ant has visited all the nodes on its list, it returns home.

As soon as an ant returns home, it updates its "self. pheromones" map for its completed journey, letting the others know that the path traveled is a popular one. Pheromone_reward grows with "self.round_trips" which has the effect of gradually eroding the previous pheromone pathway. Ants will update the "self.round_trips" count and the "self.ants_used" count, then reset itself and start solving the traveling salesman problem again.

If an ant finds a new best route, it informs the queen, which tracks the best route. The queen will then double the pheromone along this new optimal path, increasing the likelihood that the ants will follow it. The multiplier that the queen uses when a new best path is found is controlled by the "best_path_smell" variable.

The first ants quickly find a usable route, and the colony then rapidly assembles on a route that is close to ideal. Ants use the 1/distance metric to decide which city to go to next, in addition to following and laying a pheromone trail. The ants will keep doubling their efforts until they run out of options.

Ant multiplication effort is a technique used in ACO to increase the number of ants or iterate the algorithm until the optimal path is found. The goal is to improve algorithm performance and find the best path in less time. The first step in implementing ant multiplication is to start with a small number of ants or iteration in the ACO algorithm. These initial values can be selected based on prior knowledge of the problem or by experimenting with various values to find the best starting point.

The next step is to run the algorithm and record the best path found so far. This path represents the best solution of the algorithm in the current iteration. The number of ant or algorithm iterations is then doubled. Increasing the number of ants or iterations allows the algorithm to explore more paths in the search space, thereby increasing the probability of finding the optimal path. The ACO algorithm ran again in the fourth step, starting with the pheromone trail from the previous run. This is done to store previous run information and use it as a starting point for a new run.

If the best path found in this process is not better than the previous best path, the algorithm is stopped, and the previous best path is used as the final solution. It is important to note that the optimal number of ants or iterations will vary depending on the problem at hand, and trying to multiply ants does not always produce better results. As a result, it is very important to test and validate the results of the algorithms using appropriate statistical techniques, as well as to evaluate the performance of the algorithms carefully for the specific problem being solved.

The ACO algorithm that we use is also tested using statistical random variation to assess the robustness and reliability of the solution. The natural variability or uncertainty present in any data set or measurement due to chance or random factors is referred to as statistical random variation. Random variation is often quantified in statistical analysis using measures such as the standard deviation or variance. Random variations in experimental results can occur due to factors such as measurement error, differences in experimental conditions, or natural variations in the response of the system being studied. Random variation is an important consideration in statistical analysis because it can influence data interpretation and conclusions. Understanding and accounting for random variation is an important aspect of statistical analysis and is necessary to draw valid and reliable conclusions from data.

Using statistical random variation, researchers can assess how the ACO algorithm performs under different conditions and identify its strengths and weaknesses by introducing randomness into the test environment. This can help improve the algorithm and its effectiveness in real-world applications. In addition, testing with random variation helps in determining whether the solution obtained is not simply local optimal and checks the convergence of the algorithm.

2.4 Evaluation

In this section, we explain the specific evaluation criteria and methodologies for evaluating the ACO algorithm's performance and reliability when applied to the TSP. The evaluation is divided into two experiments, one using GPS coordinates as input data and the other using a map image.

In the evaluation of the ACO Algorithm using GPS Coordinates, our primary focus lies in evaluating the quality of the paths generated. To achieve this, we calculate the total distance or cost of the routes created. This metric serves as a crucial indicator of route efficiency and provides insights into the algorithm's capability to improve route efficiency when compared to the initial path length. Additionally, we conduct a convergence analysis to gain a deeper understanding of how the algorithm evolves over iterations. This analysis monitors the solution's progress and assists us in determining whether the algorithm exhibits convergent optimization or approaches near-optimal solutions. We use statistical measures such as standard deviation to ensure the stability and consistency of the algorithm's performance. A lower standard deviation indicates that the algorithm produces consistent reliable results.

Finally, we evaluate the algorithm's computational efficiency, including execution time. This evaluation assists us in understanding the algorithm's speed in producing results as well as its computational resource requirements.

In the evaluation of the ACO Algorithm using Map Image, we once again prioritize the assessment of path quality. Similar to the GPS coordinates-based experiment, we calculate the total distance or cost of the routes generated. Lower path costs are indicative of better solutions. Additionally, we conduct an effort-doubling analysis to understand the impact of increasing computational effort on solution quality. This analysis tracks changes in path cost, ant count, epochs, round trips, and time with each doubling of effort, providing valuable insights into the algorithm's behavior. We explore the algorithm's sensitivity to parameter changes, particularly examining how it responds to variations in parameters like pheromone power, distance power, and reward power, all of which can significantly affect its performance. Furthermore, we assess computational effectiveness, including execution time, to gauge the algorithm's speed in producing results. This evaluation also involves a comparative analysis with the GPS coordinate experiment to determine which input method yields superior results under varying conditions.

2.5 Result Analysis

The evaluation of experimental results is a critical step in determining the dependability of the proposed algorithms. The evaluation process for this study is centered on quantifying the performance of the ACO algorithms in addressing the Traveling Salesman Problem. A variety of metrics are carefully considered to comprehensively assess the performance of the algorithms.

In metrics evaluation, two critical criteria are used to evaluate the effectiveness of the ACO algorithms in dealing with the TSP. The first, path length, is a direct indicator of the algorithms' ability to generate optimized routes. A shorter path length indicates a more efficient and optimal solution. The evaluation uses the difference between the initial path length and the path length obtained after optimization to assess the algorithms' ability to improve route efficiency. Convergence analysis is also useful in understanding the behavior of the algorithms during the iterative optimization process. We can determine how quickly the algorithms stabilize and whether they improve steadily over iterations by studying convergence matrices and graphs. Convergence matrices and graphical representations are critical tools for this analytical task.

Other than direct metrics, statistical analysis plays a critical role in interpreting the results. This analysis is comprised of two major components: standard deviation and random variation tests. The standard deviation, an essential statistical measure, plays the role of revealing the variability and consistency of the algorithms' performance. Lower standard deviation values indicate a higher degree of consistency and reliability in the results, highlighting the dependability of the algorithms. Similarly, statistical random variation tests are used to investigate how algorithms respond to fluctuations, uncertainties, and variations. We can determine the resilience of the algorithms to different scenarios and the degree of reliability achieved by injecting randomness into the test environment.

3. Results and Discussions

3.1 Results of ACO Algorithm Implementation with GPS Coordinate

This algorithm utilizes GPS coordinate data from historical locations to solve the TSP. Using the haversine formula, the initial data of latitude and longitude coordinates is processed and transformed into a distance matrix. The ACO algorithm then uses this distance matrix as an input. The ACO algorithm iteratively explores the solution space, employing pheromone trails and heuristic principles to find the best solution that optimizes the travel route.

Both algorithms use Soekarno-Hatta Airport in Jakarta as the starting and ending points of the traveling salesman's journey.

[0, 20, 13, 25, 24, 12, 11, 18, 15, 21, 22, 39, 9, 1, 14, 10, 23, 5, 33, 26, 6, 34, 3, 2, 4, 17, 31, 52, 28, 46, 27, 7, 16, 35, 48, 43, 47, 29, 19, 50, 53, 38, 44, 37, 42, 32, 8, 36, 30, 40, 49, 41, 45, 51], the sequence number represents the path results from the first algorithm

The sequence is based on Table 1. This algorithm requires about 3.02 seconds for 100 iterations and 54 data points. The best path and distance obtained are 254 km. The convergence matrix of the results is shown in Figure 4.

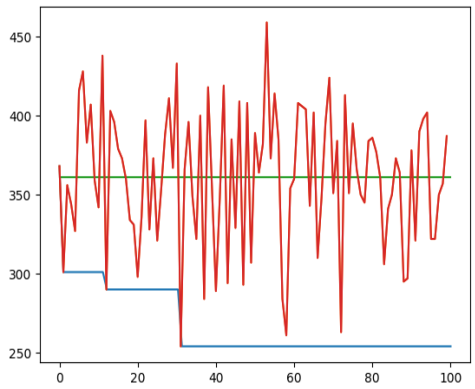

Figure 4. The Convergence Matrix of The First Algorithm

The X-axis represents the number of iterations, while the Y-axis represents the total distance value or the best score found. The blue line represents the progression of the best solution from iteration to iteration. In the context of ACO, this is the smallest value of the total distance found by the ants in the population. The red line depicts the ants' best solution at a given iteration. This means that at that iteration, the best ant in the

population discovered the best solution. The horizontal green line represents the average distance traveled by all of the ants in the population at a given iteration. This indicates how well the population performs in an iteration.

Convergence occurs when the algorithm stabilizes, and further iterations do not improve the solution significantly. The convergence criterion has been met in this case, and the best distance found remains relatively constant after a certain number of iterations. The algorithm converged before reaching the 40th iteration because it found a satisfactory solution before reaching the 40th iteration.

3.2 Results of ACO Algorithm Implementation with Map Image

The input for this algorithm is a map image depicting historical locations as shown in Figure 5. Each city or location on the map is represented as a node in a graph in this algorithm, and these nodes are connected by edges that represent the distances between the cities. This graph structure is then used as input by the ACO algorithm to find the shortest or most optimal path through the graph.

Table 2. Effort Doubling Process

Path cost	Ant used	Epoch	Round trips	Time
8256	1	7444	1	0
7221	65	15252	2	0
6808	130	23119	3	0
6580	143	24680	3	0
6125	148	24933	3	0
5997	198	30063	4	0
5755	200	30345	4	0
5490	230	32363	4	0
5467	259	35337	5	0
5454	260	35359	5	0
5374	265	35833	5	0
5349	282	36928	5	0
5203	287	37156	5	0
5156	303	38226	5	0
5111	332	41183	6	1
5016	367	43112	7	1
4952	373	43606	6	1
4878	394	45796	7	1
4843	420	47515	8	1
4783	452	49712	8	1
4781	458	50463	8	1
4655	460	50635	8	1
4529	527	55517	9	1
4432	549	56726	9	1
4418	599	60335	10	1
4410	1341	110039	22	3
4408	2206	168210	35	4
4430	4731	336530	75	11

As explained in the methodology, the first ant will find the best path and will double its efforts until it runs out of options. The results of using the ACO algorithm with a map image as input are shown in Table 2. The table displays the changes in path cost, ant count, epochs, round trips, and time during the effort doubling process.

The "Path cost" column in the table indicates the best path cost found at each iteration. A lower path cost value denotes a better outcome. The "Ant used" column shows how many ants were used in each iteration. The more ants used, the more likely it is that an optimal solution will be found. The number of iterations performed during the effort doubling process is represented in the "Epoch" column. More iterations allow for the exploration of more solution options. The "Round Trips" column shows how many round trips each ant took. More round trips mean more variations explored in search of better solutions. Finally, the "Time" column displays the amount of time spent during the effort-doubling process.

Table 2 shows that the path cost tends to decrease with each doubling of effort. This indicates that the ACO algorithm with effort doubling successfully improves on the previous iteration's solution. With doubling the effort, the number of ants used and epochs increase, indicating a broader exploration of the solution space. It should be noted, however, that after several effort doublings, the reduction in path cost becomes insignificant. This suggests that the algorithm has reached a point where further improvements are difficult to achieve. As a result, doubling effort is critical for improving solutions in the early stages, but it should be noted that increased computational cost and time are associated with it.

The second algorithm takes 11 seconds to generate the optimal path. The initial distance before ACO is 17938, and this algorithm reduces it to 4430. The number of ants needed is 4731, with 75 trips using distance_power 1.

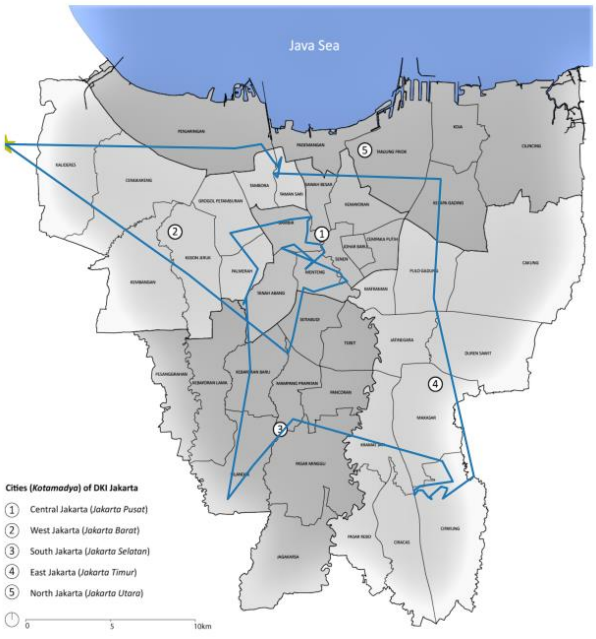

Figure 5. Result of The Second Algorithm Experiment (Based on Longitude and Latitude & Map Image)

Although the algorithm cannot guarantee finding the optimal path, it can quickly converge on a path that is close to optimal, which may be acceptable in most cases. Furthermore, statistical random variation was carried out on this algorithm, with the results shown in Table 3.

Table 3. Statistical Random Variation Result

	Count	Mean	Std	Min	Max
Results_converged	10.0	4171.1	147.5	3952.0	4348.0
Results_timed	10.0	4155.7	152.0	3991.0	4488.0
difference	0	15.4	-4.5	-39.0	-140.0

Statistical analysis reveals that for convergent results, the standard deviation is 147.5, which means about 0.04% of the average path distance (4171.1). The difference between the maximum and minimum path spacing is 396 (0.1%). The average time for merging is 23.7 seconds. When run on a fixed 10-second timer, the standard deviation decreases slightly to about 15.4 and the average path distance to about 4155.7, approximately (<1%) slightly higher than the convergent results. This shows that the algorithm is fairly consistent and can still produce reasonably good results even if it is terminated early. When running the algorithm for a long time, the "law of diminishing" results apply.

Both types of ACO take different approaches to representing input data, but they both use pheromone and heuristic principles to find the best solution in the TSP. Finally, these two algorithms are evaluated and compared in order to determine their dependability in determining the best available route.

The primary novelty of this study lies in the innovative combination of population-based ACO methods with a hierarchical pheromone update mechanism. This unique approach enhances solution quality by improving pheromone trail accuracy, which encourages better path selections among the ants and leads to a faster convergence toward optimal solutions. Additionally, introducing hierarchical pheromone updates adds a robust layer of solution consistency and enhances stability in path distance results across different runs.

Moreover, this study expands on the practical application of ACO in last-mile distribution, offering both a GPS-based and a map-image-based approach to route optimization. The dual approach demonstrates versatility in adapting to different input types, a novel direction in ACO applications that strengthens its relevance across various transportation and logistics scenarios. These findings validate that the ACO algorithm—especially when equipped with hierarchical updates—can produce stable, high-quality results suitable for real-world implementation in last-mile distribution challenges.

3.3 Implications and Further Study

This study shows that the ACO algorithm effectively addresses the TSP in route planning, potentially optimizing travel routes and improving travel experiences for both travel agencies and tourists. Travel agencies can create efficient travel itineraries by applying the ACO algorithm to tour planning, reducing travel distances between destinations and allowing tourists to spend more time enjoying their chosen destinations. This results in lower operational costs and higher customer satisfaction.

The ACO algorithm helps travelers to visit more places in less time, avoid inefficient routes or traffic-congested roads, and enjoy more time savings, cost efficiency, and improved travel experiences. This research also paves the way for the creation of advanced technological solutions for the tourism industry, such as mobile applications that offer optimized route suggestions, estimate travel times, and provide information about tourist destinations.

The research focuses on optimizing the distance between points in order to find the shortest path for the TSP. However, it has not explored the optimization of travel time between different locations. The spatial dimension of the TSP has been addressed, but the temporal dimension remains unaddressed. The incorporation of travel time optimization into the existing framework may improve the applicability of proposed solutions in real-world scenarios. Future research could focus on developing an algorithm that considers both distance and travel time at the same time, with the goal of finding routes that minimize both factors collectively. A hybrid approach could be investigated, combining ACO with other time-related optimization techniques or training machine learning models to predict travel times based on historical traffic patterns.

4. Conclusions

In this study, we conducted a search for the best route or traveling salesman problem (TSP) for museums or historical sites in Jakarta using ACO. ACO for TSP has been proven effective in finding nearly optimal solutions for small to medium-sized TSP instances. For TSP with large sizes, it may be better to add a clustering algorithm, such as using KNN. We used two algorithms, the first algorithm used coordinates or longitude and latitude points of the historical places, and the second algorithm used coordinate data from a provided map image. After implementing both models, it can be concluded that the ACO model is not very effective in solving TSP using actual coordinates. However, when using coordinates from a map image, the algorithm can quickly find a nearly optimal path. After conducting statistical random variation, the algorithm was also found to be flexible and reliable when tested under different conditions. Testing with statistical random variation can verify that the results

obtained are not only local optima and check the convergence of the algorithm.

References

[1] G. Dhiman, "ESA: a hybrid bio-inspired metaheuristic optimization approach for engineering problems," *Eng Comput*, vol. 37, no. 1, 2021, doi: 10.1007/s00366-019-00826-w.

[2] N. Rokbani *et al.*, "Bi-heuristic ant colony optimization-based approaches for traveling salesman problem," *Soft comput*, vol. 25, no. 5, 2021, doi: 10.1007/s00500-020-05406-5.

[3] A. A. Hopgood, *Intelligent Systems for Engineers and Scientists : A Practical Guide to Artificial Intelligence*, vol. 4. 2021.

[4] P. C. Pop, O. Cosma, C. Sabo, and C. P. Sitar, "A comprehensive survey on the generalized traveling salesman problem," 2024. doi: 10.1016/j.ejor.2023.07.022.

[5] B. P. Silalahi, N. Fathiah, and P. T. Supriyo, "Use of Ant Colony Optimization Algorithm for Determining Traveling Salesman Problem Routes," *Jurnal Matematika "MANTIK,"* vol. 5, no. 2, 2019, doi: 10.15642/mantik.2019.5.2.100-111.

[6] Y. Wang and Z. Han, "Ant colony optimization for traveling salesman problem based on parameters optimization," *Appl Soft Comput*, vol. 107, 2021, doi: 10.1016/j.asoc.2021.107439.

[7] A. Akhtar, "Evolution of Ant Colony Optimization Algorithm," *Computer Science - Cornell University* , 2019.

[8] S. Li, Y. Wei, X. Liu, H. Zhu, and Z. Yu, "A New Fast Ant Colony Optimization Algorithm: The Saltatory Evolution Ant Colony Optimization Algorithm," *Mathematics*, vol. 10, no. 6, 2022, doi: 10.3390/math10060925.

[9] A. L. Anindya, K. B. Y. Bintoro, and S. D. H. Permana, "Modification of Ant Colony Optimization Algorithm to Solve the Traveling Salesman Problem," *JISA(Jurnal Informatika dan Sains)*, vol. 3, no. 2, 2020, doi: 10.31326/jisa.v3i2.704.

[10] M. Dorigo and T. Stützle, "Ant colony optimization: Overview and recent advances," in *International Series in Operations Research and Management Science*, vol. 272, 2019. doi: 10.1007/978-3-319-91086-4_10.

[11] Z. Zhang, Z. Xu, S. Luan, X. Li, and Y. Sun, "Opposition-based ant colony optimization algorithm for the traveling salesman problem," *Mathematics*, vol. 8, no. 10, 2020, doi: 10.3390/MATH8101650.

[12] W. Gao, "Modified ant colony optimization with improved tour construction and pheromone updating strategies for traveling salesman problem," *Soft comput*, vol. 25, no. 4, 2021, doi: 10.1007/s00500-020-05376-8.

[13] W. Gao, "New ant colony optimization algorithm for the traveling salesman problem," *International Journal of Computational Intelligence Systems*, vol. 13, no. 1, 2020, doi: 10.2991/ijcis.d.200117.001.

[14] C. Jin *et al.*, "Development and evaluation of an artificial intelligence system for COVID-19 diagnosis," *Nat Commun*, vol. 11, no. 1, 2020, doi: 10.1038/s41467-020-18685-1.

[15] M. Ponjavic and A. Karabegovic, "Location intelligence systems and data integration for airport capacities planning," *Computers*, vol. 8, no. 1, 2019, doi: 10.3390/computers8010013.

[16] Y. Wang, L. Wang, G. Chen, Z. Cai, Y. Zhou, and L. Xing, "An Improved Ant Colony Optimization algorithm to the Periodic Vehicle Routing Problem with Time Window and Service Choice," *Swarm Evol Comput*, vol. 55, 2020, doi: 10.1016/j.swevo.2020.100675.

[17] S. M. de Oliveira, L. C. T. Bezerra, T. Stützle, M. Dorigo, E. F. Wanner, and S. R. de Souza, "A computational study on ant colony optimization for the traveling salesman problem with dynamic demands," *Comput Oper Res*, vol. 135, 2021, doi: 10.1016/j.cor.2021.105359.

[18] P. Stodola, P. Otřísal, and K. Hasilová, "Adaptive Ant Colony Optimization with node clustering applied to the Travelling Salesman Problem," *Swarm Evol Comput*, vol. 70, 2022, doi: 10.1016/j.swevo.2022.101056.

[19] A. S. Bin Shahadat, M. A. H. Akhand, and M. A. S. Kamal, "Visibility Adaptation in Ant Colony Optimization for Solving Traveling Salesman Problem," *Mathematics*, vol. 10, no. 14, 2022, doi: 10.3390/math10142448.

[20] T. Fei, X. Wu, L. Zhang, Y. Zhang, and L. Chen, "Research on improved ant colony optimization for traveling salesman problem," *Mathematical Biosciences and Engineering*, vol. 19, no. 8, 2022, doi: 10.3934/mbe.2022381.

[21] P. Du, N. Liu, H. Zhang, and J. Lu, "An Improved Ant Colony Optimization Based on an Adaptive Heuristic Factor for the Traveling Salesman Problem," *J Adv Transp*, vol. 2021, 2021, doi: 10.1155/2021/6642009.

[22] S. Kumar, D. R. Parhi, M. K. Muni, and K. K. Pandey, "Optimal path search and control of mobile robot using hybridized sine-cosine algorithm and ant colony optimization technique," *Industrial Robot*, vol. 47, no. 4, 2020, doi: 10.1108/IR-12-2019-0248.

[23] G. Che, L. Liu, and Z. Yu, "An improved ant colony optimization algorithm based on particle swarm optimization algorithm for path planning of autonomous underwater vehicle," *J Ambient Intell Humaniz Comput*, vol. 11, no. 8, 2020, doi: 10.1007/s12652-019-01531-8.

[24] R. Behmanesh and I. Rahimi, "Improved ant colony optimization for multi-resource job shop scheduling: A special case of transportation," *Econ Comput Econ Cybern Stud Res*, vol. 55, no. 4, 2021, doi: 10.24818/18423264/55.4.21.18.

[25] B. A. M. Menezes, H. Kuchen, H. A. Amorim Neto, and F. B. De Lima Neto, "Parallelization Strategies for GPU-Based Ant Colony Optimization Solving the Traveling Salesman Problem," in *2019 IEEE Congress on Evolutionary Computation, CEC 2019 - Proceedings*, 2019. doi: 10.1109/CEC.2019.8790073.

[26] R. Skinderowicz, "Improving Ant Colony Optimization efficiency for solving large TSP instances," *Appl Soft Comput*, vol. 120, 2022, doi: 10.1016/j.asoc.2022.108653.

[27] L. Eskandari, A. Jafarian, P. Rahimloo, and D. Baleanu, "A Modified and Enhanced Ant Colony Optimization Algorithm for Traveling Salesman Problem," 2019. doi: 10.1007/978-3-319-91065-9_13.

[28] W. Deng, J. Xu, and H. Zhao, "An Improved Ant Colony Optimization Algorithm Based on Hybrid Strategies for Scheduling Problem," *IEEE Access*, vol. 7, 2019, doi: 10.1109/ACCESS.2019.2897580.

[29] K. Yang, X. You, S. Liu, and H. Pan, "A novel ant colony optimization based on game for traveling salesman problem," *Applied Intelligence*, vol. 50, no. 12, 2020, doi: 10.1007/s10489-020-01799-w.

[30] C. Liu *et al.*, "An improved heuristic mechanism ant colony optimization algorithm for solving path planning," *Knowl Based Syst*, vol. 271, 2023, doi: 10.1016/j.knosys.2023.110540.

[31] A. Thammano and Y. Oonsrikaw, "Improved Ant Colony Optimization with Local Search for Traveling Salesman Problem," in *Proceedings - 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2019*, 2019. doi: 10.1109/SNPD.2019.8935817.

