Received: June 7, 2024. Revised: July 24, 2024.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Improved Whale Optimization Algorithm with Variable Neighbourhood Search Strategy (WOA-VNS) in Solving Vehicle Routing Problem (VRP) for Recommending Multi-days Tourist Routes in Yogyakarta

Saskia Putri Ananda[1] Z.K.A. Baizal[1]* Gia Septiana Wulandari[1]

[1]*School of Computing, Telkom University, Bandung, Indonesia*
* Corresponding author's Email: baizal@telkomuniversity.ac.id

Abstract: Traveling has become an essential need for people to fulfill their psychological needs. Generally, tourists want to visit a new destination for several days. To get route guidance (visiting schedule), tourists usually use the services of a travel agent, but this service cannot be tailored to the tourist's wishes. In previous research, many have concluded that one-day and multi-day tourist routes are analogous to the Traveling Salesman Problem (TSP). However, this study has yet to emphasize daily optimization for multi-day routes because daily routes are only cut based on time constraints. One possible approach to optimize tourist routes per day is the analogy of solving Vehicle Routing Problem (VRP). Therefore, in this research, we propose a new model that combines the Whale Optimization Algorithm (WOA) with a Variable Neighborhood Search (VNS) strategy known as WOA-VNS to recommend multi-day tourist routes, which is analogous to the VRP to overcome deficiencies with the TSP analogy. The number of vehicles corresponds to the number of days tourists visit, thus ensuring optimal daily routes. The system considers user preferences for popularity, ratings, and time using the concept of Multi-Attribute Utility Theory (MAUT). The MAUT value are used as WOA-VNS fitness values. Five metrics (fitness value, number of Point of Interest (POI)s, trip duration, cost, and rating attributes) were tested on five random POIs. Results show the VRP analog is more suitable for multi-day routes, with WOA-VNS-VRP outperforming WOA-VNS-TSP and conventional algorithms such as Ant Colony Optimization (ACO), Genetic Algorithm (GA), and Bat Algorithm (BA), achieving value average fitness of 0.8570, on the tourist location dataset in Yogyakarta.

Keywords: Recommender system, Multi-day tourist routes, Vehicle routing problem, Whale optimization algorithm, Variable neighborhood search strategy.

1. Introduction

The survey analysis site (money.co.uk) states Indonesia as the country with the most beautiful natural panorama in the world. This attracts tourists to visit Indonesia, especially people in cities who make travelling part of fulfilling their psychological needs. Generally, tourists want the experience of exploring a new destination during a several-day visit. However, they often need help planning their trips because the route guidance travel agents provide sometimes needs to match individual preferences. Every tourist has a unique destination, so we need a model to help them plan multi-day tourist routes according to their wishes. In this research, we chose Yogyakarta as a tourist location in the spotlight because of its cultural richness, traditional culinary delights, natural beauty, and friendliness. With these various attractions, Yogyakarta has become a world-class tourist destination that attracts the attention of many tourists.

The problem of planning tourist routes has been carried out in many previous studies. Such as research conducted by Mangini et al. [1]. and Mao [2], but this research only completed a one-day itinerary. After the development of technology, researchers began to create multi-day tourist routes. The problem of planning multi-day tourist routes is a problem that can be solved in a recommender system, as has been studied in previous studies. In previous studies, this

problem has been solved using various methods such as Ant Colony Optimization (ACO) [3], Simulated Annealing (SA) [4], and Tabu Search [5]. Previous studies determined the optimal tourist route, which is analogous to the Traveling Salesman Problem (TSP), which will be visited for several days during tourist visits. Daily deductions are made based on time constraints. In this case, the resulting route is not optimal because the emphasis on optimizing daily trips could be more assertive. Therefore, in this research, we aim to optimize multi-day tourist route planning so that each recommended daily route is optimal. One possible approach for optimizing tourist routes per day is the analogy of solving VRP. This problem will be analogous to the Vehicle Routing Problem (VRP) because TSP is more suitable for solving tourist route problems during a one-day visit. VRP is the problem for determining vehicle routes to visit several locations to reduce travel time or distance that meets given constraints. This route starts at the depot and visits several locations with one or more vehicles before returning to the depot [6]. Although the VRP problem is generally used in the context of shipping goods [7-9], in this study, we adapt it to respond to specific problems in multi-day tourist route planning, as was done in previous research using Ant Colony System (ACS) and Brainstorm Optimization Algorithm (BSO) [10]. With the number of vehicles adjusted to the number of visiting days, one vehicle will search for a route for one visiting day.

The tourist route recommender system that is built is expected to be able to provide optimal routes for several days of visits, as well as optimize daily tourist routes according to individual tourist preferences. There are several route criteria (multi-criteria based) considered in this research, based on various aspects such as the popularity or rating of tourist attractions, cost, and time of visit (where the longer the time available, the more places can be visited in a day). To address this issue involving multiple criteria, we utilize the concept of Multi-Attribute Utility Theory (MAUT). MAUT is a decision-making technique that helps assess and select various available options [11], especially when decision-makers face complex situations with many attributes that need to be considered, each with different weights [12]. Apart from considering user preferences, the model must also consider other factors, such as the opening hours of tourist attractions and the tourist's desired time. In mathematical terms, a multiple-vehicle routing problem with time windows refers to the challenge of planning routes that encompass multiple destinations while also accounting for specific timeframes for each visit. Time windows constraints exist for tourists and tourist attractions. Tourists' time window refers to the time they determine when visiting, while the time window of tourist attractions refers to the opening and closing hours of the place.

The algorithm commonly used to handle optimization problems is the Swarm Intelligent algorithm, one of which is the Whale Optimization Algorithm (WOA). The choice of the WOA for route optimization can be attributed to its unique characteristics and capabilities. The WOA imitates how whales interact socially and takes inspiration from the bubble-net hunting strategy employed by humpback whales [13, 14]. Because of its ability to search for global optimization, this method can be used to solve a variety of challenging optimization issues [15]. In order to overcome the WOA's drawbacks, which include sluggish convergence speed and local optima, numerous techniques have been added to it, including simulated annealing, adaptive weighting, Gaussian mutation, and differential evolution [16, 17]. These modifications have strengthened the algorithm's capacity to synchronize global search and local development, increasing its efficiency for optimization tasks [18]. Furthermore, the WOA has been successfully applied in various domains, such as wireless sensor networks, weather prediction, and image denoising, demonstrating its versatility and effectiveness in solving diverse optimization problems [19-21]. The algorithm's adaptability and ability to handle multi-objective optimization tasks make it suitable for complex routing optimization problems, where multiple objectives need to be considered simultaneously [18]. In summary, the whale optimization algorithm is chosen for route optimization due to its global optimization-seeking capability, adaptability, versatility, and successful application in various domains. Its unique inspiration from the social behavior of whales, along with continuous enhancements and integrations with other algorithms, makes it a promising choice for addressing complex routing optimization problems.

Therefore, in this research, we propose a new model by analogizing the multi-day tourist routes problem with VRP to overcome the shortcomings with the TSP analogy, using WOA optimized with Variable Neighborhood Search Strategy (VNS), known as WOA-VNS. We chose VNS because of its ability to expand the solution search space. It will produce a solution that can avoid local optimal traps and provide recommendations for more optimal and efficient multi-day tourist routes.

The rest of this document is structured as follows. Section 2 presents multiple studies concerning travel

route suggestions, addressing VRP in different scenarios, and implementing the hybridized WOA algorithm to tackle diverse challenges. Following that, Section 3 outlines the research methodology. In Section 4, we analyze the experimental outcomes of our proposed algorithm, juxtaposed with several alternative algorithms across five metrics. Lastly, Section 5 concludes and offers recommendations for future research.

2. Related work

The problem of multi-day tourist routes has been discussed in recent years in recommendation systems [3-5]. In these studies, tourist route planning has been analogous to the Traveling Salesman Problem (TSP), where a travel route is determined for visits over several days, then the system makes daily deductions based on time constraints. Thus, the resulting route is not optimal per day because TSP is more suitable for solving tourist route problems in one day of visit.

One approach that optimizes tourist routes per day in Another approach that can optimize tourist routes per day in multi-day tourist routes is the analogy of solving the Vehicle Routing Problem (VRP). In previous research, VRP was often associated with solving goods delivery problems [7], [22] and flow shop scheduling [23]. Solving these two problems involves different VRP variants tailored to the specific constraints of each problem. For example, Sbai et al. [24] formulated the postal distribution problem as a variant of the well-known VRP: Capacitated Vehicle Routing Problem (CVRP). In addition, Erderić et al. [25] solve the problem of shipping goods by considering full recharge and partial recharge. By considering these two things, researchers analogized the problem of shipping goods as Electric Vehicle Routing Problem with Time Windows and Full Recharge at CSs (EVRPTW-FR) and Electric Vehicle Routing Problem with Time Windows and Partial Recharging (EVRPTW-PR). On the other hand, Yang et al. [26] expanded the Split Delivery Vehicle Routing Problem (SDVRP) variant called SDVRP with Goods Consumed during Transit (SDVRP-GCT). This research considers conditions where the food or goods transported by a vehicle decrease gradually during the journey due to consumption or use. Another study was conducted by Azad et al. [23], who tried to integrate permutation flow shop scheduling with VRP. The VRP analogy optimizes distribution routes to suit predetermined production schedules.

Although VRP is typically used in freight forwarding and flow shop scheduling, it can also be adapted to solve multi-day tourist routing problems. The success of this approach in the context of multi-day tourist routes has been proven by research conducted by Hendrawan et al. [10]. Regarding our literature study, this research is the first to analogize the problem of multi-day tourist routes to VRP. This research considers trip duration per day and time windows based on opening and closing hours between POIs, so it is assumed to be CVRP with Time Windows (CVRPTW). The algorithm used is the Ant Colony System (ACS), which is optimized with the Brainstorm Optimization Algorithm (BSO) and is called hybrid ACS-BSO. The results of this study show that the multi-day tourist route problem, which is analogous to VRP, outperforms the multi-day tourist route problem, which is analogous to TSP, on four of the five metrics tested. Therefore, this research will also solve the multi-day tourist routes problem by analogizing it with the VRP problem using other optimization algorithms.

One algorithm that provides promising solutions and is often applied in solving optimization problems, such as the Vehicle Routing Problem (VRP), is the Whale Optimization Algorithm (WOA). The main advantage of WOA lies in its ability to explore the solution space efficiently. This algorithm, inspired by the hunting behavior of whales, combines exploration and exploitation in a balanced manner [27]. This balance allows WOA to effectively search for optimal solutions in large solutions, which is essential in dealing with combinatorial optimization problems such as VRP, where the search space is vast and complex [28]. By leveraging these exploration capabilities, WOA can navigate complex routes and multiple constraints within the VRP to find near-optimal solutions. WOA is also known for its ability to achieve convergence quickly and its ease of implementation, making it a practical choice for optimizations such as VRP [27]. The simple structure of the algorithm and the small number of parameters to adjust increase its efficiency in finding solutions in a reasonable time, which is very important for real-world applications such as route planning in VRP scenarios [29]. WOA's convergence speed is particularly advantageous for optimization problems that require fast solutions, where time is crucial.

In addition, the adaptability and flexibility of WOA make it very effective in facing various optimization challenges, including VRP [27]. This algorithm can handle multiple optimization problems and be modified to improve its performance, making WOA a powerful tool for dealing with the complexity of VRP [29]. By adapting to the specific needs of each VRP instance, WOA can tailor its search process to meet the unique demands of route

optimization and vehicle scheduling. WOA also shows competitive performance compared to other metaheuristic algorithms such as Particle Swarm Optimization (PSO) [30], Gray Wolf Optimizer (GWO) [31], Simulated Annealing (SA) [32], and Differential Evolution (DE) [33] in various optimization problems [34]. These advantages make WOA a viable option for overcoming complex routing challenges in VRP.

However, although the Whale Optimization Algorithm (WOA) has several advantages in solving optimization problems, WOA also has weaknesses that must be considered, namely the tendency to enter local optima [14] and premature convergence [35]. One way to overcome this deficiency is to hybridize it with other approaches. In previous studies, WOA has been combined with various approaches, such as those carried out by Bassett et al. [36], which optimizes WOA with insert-reversed block, Nawaz-Enscore-Ham (NEH), Local Search Strategy, and swap mutation in solving flow shop scheduling problems. In addition, Demiral combines WOA with the Nearest Neighbor (NN) algorithm to solve TSP because classical WOA provides lower results in solving TSP. Jiang et al. [37] also designed Green Open VRP (GOVRP) to minimize fuel consumption and developed a Hybrid WOA (HWOA) to overcome this problem. This research uses four neighborhood structure variables (exchange operation, swap operation, insertion operation, and inverse operation) to build a Variable Neighborhood Search (VNS). Testing was carried out on 18 cases, which were tested 20 times. The test results show that HWOA outperforms other comparison algorithms (SA and HEDA) both in terms of best value (BST) and average value (AVG). Zhang et al. [38] also combine WOA with Variable Neighborhood Search Strategy (VNS), Gaussian interference, and adaptive weighting to overcome the slow convergence problem in conventional WOA when solving TSP. The resulting algorithm is called the Discrete Whale Optimization Algorithm with Variable Neighborhood Search (VDWOA). The results show that the optimal value obtained from WOA combined with VNS is superior to other comparison algorithms.

From the various hybridizations carried out on WOA, WOA gives optimal results when combined with VNS. This is because WOA is known for its high exploration capabilities and efficiency in exploring the solution space, which significantly helps deal with the complexity of VRP. Meanwhile, VNS provides varied search strategies through dynamic environmental changes, producing better solutions [39], [40]. This hybridization creates a balanced approach, combining the global exploration capabilities of WOA with the structured local search strategy of VNS, providing a comprehensive optimization framework for addressing VRP challenges [31]. Thus, integrating WOA with VNS can overcome problems such as premature convergence and trapping in local optima that WOA may experience.

Based on our literature study, previous research on the problem of multi-day tourist routes analogous to TSP is less than optimal because it does not emphasize optimizing daily routes. In the TSP analogy, the most optimal route is determined first, then cut based on time constraints for each day. Therefore, this research aims to optimize multi-day tourist route planning with the VRP analogy. Using the VRP analogy, the number of vehicles is adjusted to the number of visiting days to make the resulting route more optimal every day. We use enhanced WOA with VNS for route discovery. The WOA was selected for its superior global search capabilities, high adaptability, flexibility and successful application in various fields. We improve WOA with VNS because of its advantages, which include expanding the search space, helping WOA avoid local optimum, and solving combinatorial optimization problems on a large scale. To prove the effectiveness of the proposed algorithm, we will carry out a comparison with pure WOA, another conventional algorithm that is often applied and provides quite optimal results in solving various variants of VRP (Ant Colony Optimization (ACO) [41, 42], Genetic Algorithm (GA) [42, 43], and Bat Algorithm (BA) [44, 45]), as well as a comparison between the VRP and TSP based.

3. Methodology

This study focuses to find optimal multi-day travel itinerary based on user's preferences. User can input their desired Point of Interest (POI) and the Degree of Interest (DOI) for each attribute. The attributes consist of travel time, cost, and rating. The system will arrange inputted POI to each day in the itinerary using greedy strategy based on user's time windows and POI's time windows. The itinerary starts at 08.00 AM and ends at 09.00 PM each day. Notations that are used in this study are follows,

V	Length of agent
N	Number of days
D	POI set $\{d_1, d_2, d_3, \ldots, d_V\}$
$\vec{X}$	Agent
$\vec{A}$	Coefficient vector
$\vec{C}$	Coefficient vector

A	Array (a_1, a_2, a_3,…, a_V) containing indices that sort S
R	Array (r_1, r_2, r_3,…, r_V) containing sorted D by indices in A
T	MAUT value
T'	Normalized MAUT value
H	Set of MAUT attributes
h	MAUT attribute
DOI_h	Degree of interest of h attribute
s_h	Value of attribute
$s_{h,norm}$	Normalized value of h attribute
$s_{h,min}$	Minimum value of an attribute
$s_{h,max}$	Maximum value of an attribute
T_{min}	Minimum value of MAUT
T_{max}	Maximum value of MAUT
t	Index of current iteration
$\overrightarrow{X^*}$	Best agent
b	Constant that determines the form of the logarithmic spiral
l	Random number in range [-1,1]
p	Random number in range [0,1]
z	Random number in range [0,1]
$\overrightarrow{X_{rand}}$	Selected random agent
$\vec{D}$	Distance between an agent and best agent
$\vec{a}$	Vector that the value decreases from 2 to 0 throughout the iteration process
$\vec{r}$	Random vector in range [0,1]

3.1 Dataset

In this research, we use tourist locations in Yogyakarta. We get the dataset via Google Maps API (time matrix) and Serp API (hotel and tourist attraction data). The dataset comprises 88 hotels and 99 POI options. Tables 1 and 2 show examples of hotel and POI data respectively. Tables 3 to 5 show sample of travel duration between POIs, from POI to hotel, and from hotel to POI respectively.

3.2 Problem identification and fitness value

The problem of multi-day tourist routes is defined as the Vehicle Routing Problem (VRP). VRP provides more flexibility in setting boundaries. Meanwhile, the Traveling Salesman Problem (TSP) determines the most efficient route by visiting a particular set of POIs exactly once and then returning to the hotel. The focus is on selecting the best order to visit POIs, aiming to minimize overall travel time throughout the day in the itinerary. By VRP, we can expand itinerary optimization boundaries by optimizing the number of POIs included and the cost and ranking. Fig. 1 shows how the VRP divides the itinerary. We analogize each day's itinerary as a vehicle. Each vehicle will find its own POI.Fig. 2 shows how TSP breaks down travel plans. TSP will find the shortest path to visit all POIs.

Table 1. The example of hotel data

No	Title	Rating	Lattitude	Longitude
1	Aveta Hotel Malioboro	4.4	-7.793524490450224	110.36568239941055
2	Kj Hotel Yogyakarta	4.5	-7.8221015220655925	110.36748019261607
3	Grand Rohan Jogja	4.7	-7.7980359294743735	110.40502653679434
4	Indonesia Hotel	4.2	-7.791483247630746	110.3651444367943
5	Kalya Hotel	4.3	-7.813308578385204	110.40138240980863

Table 2. The example of POI data

No	Place name	Rating	Opening hour	Closing hour	Latitude	Longitude	Time spent (second)	Cost (IDR)
1	Museum sandi	4.7	00.00	23.59	-7.7845549	110.3711548	5400	3,000
2	Tugu	4.8	00.00	23.59	-7.7829218	110.3670757	3600	0
3	Omah UGM Kotagede Yogyakarta	4.5	16.00	18.00	-7.8137278	110.3629074	1800	35,000

Table 3. The example of travel duration data (between POIs)

Origin	Destination	Travel duration (second)
Embun Ketingan	Taman Kehati	1706
Merapi Park Yogyakarta	Grojogan Watu Purbo Gate	1629
Pasar Beringharjo	House of Chocolate Monggo & Gelato Tirtodipuran	565

Table 4. The example of travel duration data (from POI to Hotel)

POI	Hotel	Travel duration (second)
Pasar Kotagede	Hotel Tentrem Yogyakarta	1616
Bentara Budaya Yogyakarta (BBY)	Novotel Suites Yogyakarta Malioboro	366
Panggung Krapyak	Yogyakarta Marriott Hotel	2019

Table 5. The example of travel duration data (from Hotel to POI)

Hotel	POI	Travel duration (second)
Grand Tjokro Yogyakarta	Desa Wisata Pentingsari	2444
Grage Bussines Hotel Yogyakarta	Golden Bioskop Virtual Reality	197
Hotel Laxston Jl. Magelang	Bukit Pangol	2575

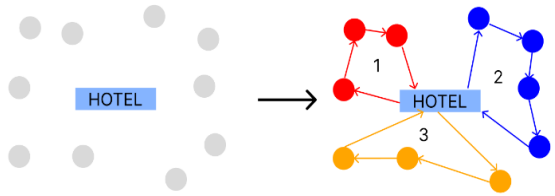

Figure. 1 VRP flow to split itinerary

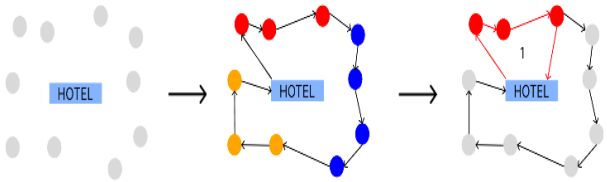

Figure. 2 TSP flow to split itinerary

Then, the itinerary is divided based on the user's selected time windows. This mechanism can produce optimal local solutions, which may have a short path one day and a long path another [19].

Multi-day itineraries are represented as an array called an agent. The length of the agent is equal to the number of POIs selected by the user. For example, if the user selects 15 POI, the length of the agent is 15. The dimension of an agent denoted by V. An agent consists of V real values in the range [0,10]. Algorithm 1 shows the agent decoding method to obtain a multi-day tourist route using a greedy algorithm. First the algorithm produce array $A\ (a_1, a_2, \dots, a_V)$ containing indices that sort inputted agent $\vec{X}$ (performed in line 1). Array A produced by Eq. (1) where function $argsort()$ is a function that return the indices that would sort an array. Once array A produced, the algorithm produce array $R\ (r_1, r_2, \dots, r_V)$ that containing array D that has sorted by indices in A (performed in line 2). Array D contain selected POI. Elements of array R produced by Eq. (2).

$$A = argsort(\vec{X}) \quad (1)$$

$$r_v = D[a_v] \quad (2)$$

The algorithm arranges POIs in R to N different array. Where N denote number of days in itinerary. Thus, each array will store route for corresponding day. All N different array stored in variable array $routes$ initialized in line 4. The algorithm arrange POI using greedy strategy. In line 3 of the algorithm, array $R\_assigned$ is initialized to store each POI that has included to the itinerary. The strategy keeps running while any unassigned POI (POI that not included in $R\_assigned$) is possible to be included to one of the days. This condition assessed in line 5. If the condition fulfilled, the greedy strategy select a day that have fewest POI included in it. The index of selected day is stored to variable $selected\_day$ as shown in line 7.

Once the algorithm selects an index of a day, the algorithm starts to find POI that possible to be included to selected day route. In line 8 of the algorithm, variable $is\_poi\_has\_assigned$ is initialized to indicate is there any POI has assigned to selected day. If the value of $is\_poi\_has\_assigned$ is false (assessed in line 9), the algorithm iterates over POIs in R to check if there any POI is possible to be included to selected day route (as assessed in line 12). A POI is possible to be included to the selected day route if its time windows match with the route and the POI not included in $R\_assigned$. If the any POI is possible to be included to selected day route, the POI will be included to corresponding day (the POI is appended to $routes[selected\_day]$) as performed in line 13. The algorithm also change value of $is\_poi\_has\_assigned$ to true to indicate a POI has included to itinerary (performed in line 15). Thus, when the algorithm returns to line 9, the condition is not met and then the algorithm return to line 5.

If no POI is possible to be assigned to selected day, the algorithm append value of $selected\_day$ to array $days\_included$ as performed in line 20.

Algorithm 1 $DecodeAgentToItinerary\ (S, D, N) \rightarrow routes$ (N days route)
// Generate N days itinerary by values in S

Input: S (agent to be converted to routes), D (Set of POIs chosen by user), N (number of days chosen by user)

1: $A \leftarrow$ array calculated by (1)
2: $R \leftarrow$ array which the element calculated by (2)
3: $R\_included \leftarrow$ empty array, used to store POIs that has assigned to route
4: $routes \leftarrow$ Array containing N empty array. Each empty array would store route for each day
5: **while** any POI possible to assign to one of the days **do**
6: $days\_included \leftarrow$ empty array, used to store indices of day that has selected but no POI possible to assign to it
7: $selected\_day \leftarrow$ index of day with fewest POI included
8: $is\_poi\_has\_assigned \leftarrow false$
9: **while not** $is\_poi\_has\_assigned$ **do**
10: $i \leftarrow 0$
11: **while** $i <$ length (R) **and not** $is\_poi\_has\_assigned$ **do**
12: **if** $R[i]$ possible to assign to $selected\_day$ **and** $R[i]$ not in $R\_included$ **then**
13: Append $R[i]$ to $routes[selected\_day]$
14: Append $R[i]$ to $R\_included$
15: $is\_poi\_has\_assigned \leftarrow true$
16: **end if**
17: $i \leftarrow i + 1$
18: **end while**
19: **If not** $is\_poi\_has\_assigned$ **then**
20: Append $selected\_day$ to $days\_included$
21: $selected\_day \leftarrow$ index of a day outside $days\_included$ with fewest POI included
22: **end if**
23: **end while**
24: **end while**
25: **return** $routes$

Where $days\_included$ is an array that contain indices of day that has selected but no POI is possible to be assigned to it. Once value of selected day is appended to $days\_included$, the algorithm looks for another index of day outside $days\_included$ with fewest POI included in that day. The selected index will replace value of $selected\_day$ as performed in line 21. After value of $selected\_day$ is replaced, the algorithm returns to line 9 with new value of $selected\_day$.

If no POI left or every unassigned POI is not possible to be included to itinerary, the condition in line 5 not fulfilled. So, the greedy strategy is stopped, and the algorithm return decoded routes as shown in line 25.

Fitness function is used to evaluate and compare and agent to another agent. We used MAUT value as fitness value to evaluate agent. Agent is evaluated by four attributes, e.g., time, cost, popularity, and POI included in itinerary. User can input Degree of Interest (DOI) to attributes time, cost, and popularity. We set DOI value from 0 to 1. The higher the attribute's DOI value, the higher user preferences to its attribute. We set DOI to attribute POI included to 1. MAUT value is calculated by Eq. (3). Where T is MAUT value, H is set of attributes, and $s_{h,norm}$ is normalized attribute value. $s_{h,norm}$ value is calculated by Eq. (4). Where s_h is attribute value, $s_{h,min}$ is minimum value of the attribute, and $s_{h,max}$ is maximum value of the attribute. To compare WOA with other algorithm, we normalized fitness value in range [0, 1] by applying Eq. (5) [20]. Where T' is normalized fitness value, T_{min} and T_{max} is minimum and maximum value of fitness value respectively.

$$T = \sum_{h \in H} DOI_h \cdot s_{h,norm} \quad (3)$$

$$s_{h,norm} = \frac{s_h - s_{h,min}}{s_{h,max} - s_{h,min}} \quad (4)$$

$$T' = \frac{T - T_{min}}{T_{max} - T_{min}} \quad (5)$$

3.3 Whale optimization algorithm (WOA)

The WOA is a computational algorithm inspired by the hunting behavior of humpback whales, particularly their distinctive method known as bubble-net feeding. In this technique, humpback whales dive approximately 12 meters underwater, encircle their prey, create spiraling bubbles, and

ascend toward the surface. This behavior can be modelling in mathematics.

$$\vec{X}(t+1) = \overrightarrow{X^*}(t) - \vec{A} \cdot \vec{D} \quad (6)$$

$$\vec{D} = \left|\vec{C} \cdot \overrightarrow{X^*}(t) - \vec{X}(t)\right| \quad (7)$$

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a} \quad (8)$$

$$\vec{C} = 2 \cdot \vec{r} \quad (9)$$

3.3.1. Encircling prey

The swarm of Humpback whale can recognize their prey's location. A Humpback whale (agent) will approaching best whale. Best agent is the whale closest to prey. Agent updating position is calculated by Eq. (6). Where t indicates index of iteration, $\vec{A}$ and $\vec{C}$ are coefficient vector, X^* is the best agent, X is current agent. $\vec{D}$ is distance of current agent to the best agent. $\vec{D}$ is calculated by Eq. (7). Vector $\vec{A}$ and $\vec{C}$ is calculated by Eq. (8) and Eq. (9) respectively. In both the exploration and exploitation phases, the vector $\vec{a}$ linearly decreases from 2 to 0 throughout the iteration process. Additionally, $\vec{r}$ represents a random vector ranging from 0 to 1.

3.3.2. Bubble net attacking (exploitation phase)

There are two methods to implement bubble net attacking:

- The shrinking encircling mechanism is attained by reducing the value in Eq. (3). Moreover, the fluctuation range of $\vec{A}$ is narrowed down by $\vec{a}$.
- Spiral updating position. Current behavior updates the agent by updating position of the agent with spiral helix movement. This updating position is calculated by Eq. (10). Where $\overrightarrow{D'}$ is the distance of current agent to best agent, b is a constant used to determine the form of the logarithmic spiral, while l is a random number ranging from -1 to 1.

$$\vec{X}(t+1) = \overrightarrow{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \overrightarrow{X^*}(t) \quad (10)$$

The humpback whale swarm simultaneously employs the shrinking encircling mechanism and updates its position along the spiral. We assume that both the shrinking encircling mechanism and the spiral position update will be selected with a 50% probability. This selection process is represented in Eq. (11), where p is a random number ranging from 0 to 1.

$$\vec{X}(t+1) = \begin{cases} \overrightarrow{X^*}(t) - \vec{A} \cdot \vec{D} & if\ p < 0.5 \\ \overrightarrow{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \overrightarrow{X^*}(t) & if\ p \geq 0.5 \end{cases} \quad (11)$$

3.3.3. Search for prey (exploration phase)

In this phase, agent will be searching prey according to random selected agent. The updating position is calculated by Eq. (12). Where $\overrightarrow{X_{rand}}$ is random agent and $\vec{D}$ is distance between agent and best agent. $\vec{D}$ value calculated by Eq. (13).

$$\vec{X}(t+1) = \overrightarrow{X_{rand}}(t) - \vec{A} \cdot \vec{D} \quad (12)$$

$$\vec{D} = \left|\vec{C} \cdot \overrightarrow{X_{rand}}(t) - \vec{X}(t)\right| \quad (13)$$

WOA algorithm can be seen in Algorithm 2. First WOA evaluate each agent by its fitness value (performed in line 1) and then pick the best agent (performed in line 2). Iteration will continue running if current iteration index (denoted by t) lower than maximum number of iteration (examine in line 3). In each iteration, each agent is updated using either encircling prey, bubble net attacking, or search for prey based on value of $\vec{a}$, A, C, l, and p. Those value are updated in line 6. If value of p lower than 0.5 and value of $|A|$ lower than 1, agent updated using encircling prey as performed in line 9. If value of p lower than 0.5 and value of $|A|$ equal or higher than 1, agent updated using search for prey as performed in line 12. If value of p higher or equal than 0.5, agent updated using bubble net attacking as performed in line 15. In the end of each iteration, WOA update fitness value of each agent (performed in line 19) and then update $\overrightarrow{X^*}$ if any agent better than $\overrightarrow{X^*}$ (performed in line 20).

Algorithm 2 $WOA(population, max\_iter) \rightarrow$ best agent
// Find best agent using WOA

Input: $population$ X_i (i = 1, 2, 3, …, n), max_iter

1: Calculate the fitness value of each agent
2: $\overrightarrow{X^*} \leftarrow$ the best agent
3: t = 0
4: **while** $t <$ max_iter **do**
5: **for** each $agent$ in $population$ **do**
6: update $\vec{a}$, A, C, l, and p
7: **if** p < 0.5 **then**

```
        if |A| < 1 then
          Update agent using Eq. (6)
        else if |A| ≥ 1 then
          X_rand ← random search agent
          Update agent using Eq. (12)
        end if
      else if p ≥ 0.5 then
        Update agent using Eq. (10)
      end if
    end for
    Amend agent where goes beyond search space
    Calculate the fitness of each search agent
    Update X* if any better solution
    t ← t + 1
  return X*
```

3.4 Variable neighborhood search strategy (VNS)

Variable Neighborhood Search (VNS) stands as a metaheuristic optimization algorithm employed to address combinatorial and global optimization problems. The fundamental concept of Variable Neighborhood Search involves navigating the solution space through the utilization of diverse neighborhood structures. Each neighborhood structure delineates a series of maneuvers capable of transitioning one solution into another. The VNS systematically explores multiple neighborhoods in an iterative fashion, aiming to break free from local optima and discover improved global solutions. VNS algorithm can be seen in Algorithm 3 [21]

VNS will keep running if the duration (stored in variable $duration$) not exceed maximum determined duration (denoted by $max\_elapsed$). The condition assessed in line 5 of algorithm. First VNS initialize variable k in line 6 and assign its value with 1. Once k initialized, VNS run function $n1()$, $n2()$, or $n3()$ to obtain new agent while value of k is equal or lower than 3 (value of k assessed in line 7).

VNS run function $n1()$ (implemented in line 9) if the value of k is 1. Function $n1()$ produce new agent by selects two random values (called as minimum index and maximum index) in current variable $agent$ and then reverse the elements between them.

VNS run function $n2()$ (implemented in line 11) if the value of k is 2. Function $n2()$ select three random values in $agent$ e.g., a, b, and c. This function produces new agent by running $n1()$ on agent with minimum index and maximum index is a and b. Then run $n1()$ again with minimum index and maximum index is a and c. And then run $n1()$ again with minimum index and maximum index is b and c.

VNS run function $n3()$ (implemented in line 13) if the value of k is 3. Function $n3()$ select two random values in $agent$ and swap them [36] to produce new agent.

New produced agent by $n1()$, $n2()$, or $n3()$ is stored to variable $agent\_$. VNS compare fitness value of $agent\_$ and the $agent$ as shown in line 17. If $agent\_$ better than $agent$, $agent$ is replaced by $agent\_$ as implemented in line 18. Then value of k is set to 1 as implemented in line 19. If $agent\_$ worse than $agent$, VNS done the increment for k as implemented

Algorithm 3 $VNS(agent, max\_duration) \rightarrow$ agent
// Optimize an agent using VNS

Input: $agent, max\_duration$

```
 1: start_timestamp ← current timestamp
 2: agent_ ← Null
 3: curr_ts ← current timestamp
 4: duration ← curr_ts - start_timestamp
 5: while duration < max_duration do
 6:   k ← 1
 7:   while k <= 3 do
 8:     if k = 1 then
 9:       agent_ ← n1(agent)
10:     else if k = 2 then
11:       agent_ ← n2(agent)
12:     else
13:       agent_ ← n3(agent)
14:     end if
15:     fitness_agent ← fitness_value(agent)
16:     fitness_agent_ ← fitness_value(agent_)
17:     if fitness_agent_ > fitness_agent then
18:       agent ← agent_
19:       k ← 1
20:     else
21:       k ← k + 1
22:   end while
23:   curr_ts ← current timestamp
24:   elapsed ← curr_ts - start_timestamp
25: end while
26: return agent
```

in line 21. After value of k is updated, VNS return to run line 7. If value of k greater than 3, VNS updated current value of $duration$ (updated in line 24) and then return to line 5. If then duration reach maximum duration, VNS stop the iteration and then return the agent.

3.5 WOA-VNS

The WOA is vulnerable to local optimal. So, we used VNS to address this problem. The WOA-VNS algorithm is little bit different with WOA. There is one additional variable z. In each agent iteration, z is assigned with random value between [0,1]. The WOA-VNS will check z value before running encircling prey or bubble net attacking, as performed in line 11 and 18. If z value lower than 0.5, the WOA-VNS will run VNS instead of running encircling prey or bubble net attacking. Meanwhile, the WOA-VNS algorithm can be seen in Algorithm 4.

4. Experiment result

This research analyzes various travel plans generated by the WOA-VNS-VRP algorithm and evaluates their reliability through experiments. First, experiments are carried out comparing WOA-VNS and pure WOA in the context of VRP to show the superiority of the hybrid algorithm over the standalone algorithm, with variations in the number of iterations and agents. Second, WOA-VNS-VRP is compared with WOA-VNS-TSP to prove the superiority of the VRP approach over TSP. Other experiments compare WOA-VNS with other conventional optimization algorithms such as ACO, GA, and BA in the context of VRP to evaluate the general performance and demonstrate the effectiveness of the proposed algorithm compared with conventional algorithms. Additionally, in-depth analysis is performed to evaluate the effectiveness of each algorithm based on individual attributes. The metrics used include fitness value as the primary metric, the number of POIs entered, trip duration, total cost, and average rating as secondary metrics. All experiments were implemented using Python on an 8th Generation Intel Core i5-8250U ~1.60 GHz processor with 16GB RAM. The testing result can be seen in Table 8. We used the best hyperparameter based on previous hyperparameter testing.

4.1 Comparison of hybrid and standalone algorithms

The research will examine the performance of WOA-VNS compared to pure WOA in the test scenario. We will assess the impact of the number of agents and iterations on the routes developed through VRP. The testing will involve a range of agents and iterations, specifically between 50 and 125. The criterion employed to assess the quality of a route is the fitness value assigned to each itinerary. The outcomes of WOA and WOA-VNS are displayed in Tables 6 and 7 correspondingly.

Table 6. The result of WOA testing

	The Number of Iterations			
The Number of Agent	50	75	100	125
50	0.8767	0.8804	0.8499	0.8573
75	0.8473	0.8153	0.8354	0.8469
100	0.8463	0.8631	0.8539	0.8658
125	0.8565	0.8753	0.8672	0.8735

Table 7. The result of WOA-VNS testing

	The Number of Iterations			
The Number of Agent	50	75	100	125
50	0.8043	0.8691	0.8705	0.8803
75	0.8736	0.8796	0.8818	0.9024
100	0.8668	0.8741	0.9207	0.9456
125	0.8757	0.8828	0.9363	0.9463

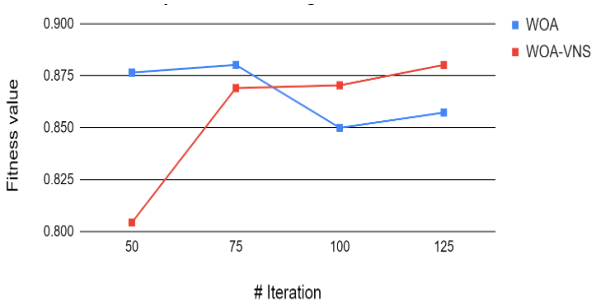

Figure. 1 Fitness value comparison when #agent=50

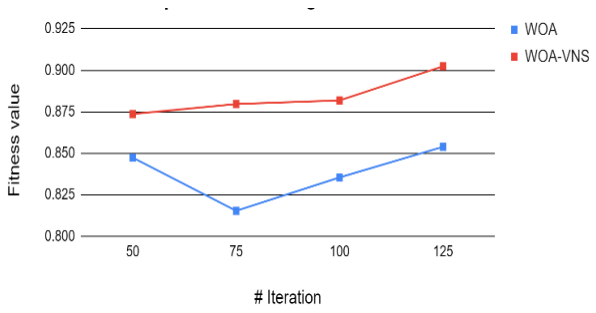

Figure. 2 Fitness value comparison when #agent=75

4.1.1. Parameter setup

The study fine-tuned the parameters for each algorithm to strike a balance between solution quality and computational time. The hybrid WOA-VNS incorporates parameters from both the WOA and VNS algorithms. The hybrid WOA-VNS has parameter $vns\_duration$ with the value is 1 second.

4.1.2. Testing result

The results of WOA and WOA-VNS can be seen in Tables 6 and 7, respectively. Figure. 1 compares the fitness values of the WOA and WOA-VNS

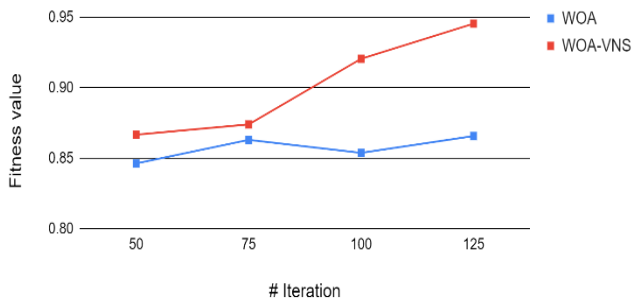

Figure. 3 Fitness value comparison when #agent=100

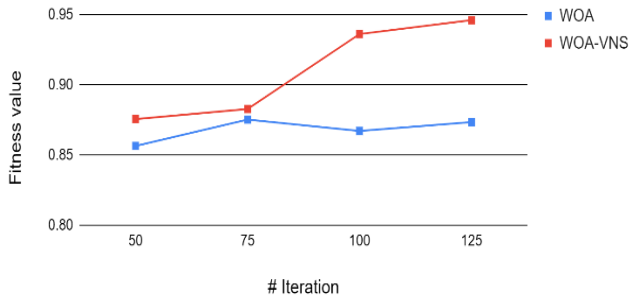

Figure. 4 Fitness value comparison when #agent=125

methods from testing with 50 agents ($A-50$). The WOA method has a higher fitness value when the number of iterations is 50 ($I-50$) and 75 ($I-75$), but the WOA-VNS method has a higher fitness value when the number of iterations is 100 ($I-100$) and 125 ($I-125$). Figure. 2-6 show the fitness value of the WOA and WOA-VNS methods, with several agents being 75 ($A-75$), 100 ($A-100$), and 125 ($A-125$), respectively. The WOA-VNS has better fitness value in all combinations of several iterations and agents. Furthermore, in $A-100$ and $A-125$, WOA-VNS WOA-VNS gain higher fitness value as the number of iterations increases. So, WOA-VNS is very stable when the number of agents is 100 or more. When $I-125$, methods WOA and WOA-VNS gain the best fitness value compared to other iterations. So, a higher number of iterations and agents led those algorithms to gain a higher fitness value. Therefore, the VRP method is superior in producing multi-day tourist route plans compared to the TSP method.

Algorithm 4 $WOA\_VNS\,(\,population\,,\ m\_i\,) \rightarrow$ best agent
// Find best agent using WOA and VNS

Input: $population\ X_i\ (i = 1, 2, 3, \ldots, n)$, m_i

Calculate the fitness value of each agent
$X^* \leftarrow$ the best agent
$t \leftarrow 0$
while $t <\ m\_i$ **do**
  for each $agent$ in $population$ **do**
    update $\vec{a}$, A, C, l, z and p
    if $p < 0.5$ **then**
      if $|A| < 1$ **then**
        Update $agent$ using Eq. (4)
      else if $|A| \geq 1$ **then**
        if $z \geq 0.5$ **then**
          Update $agent$ using VNS
        Else
          $X_{rand} \leftarrow$ random search agent
          Update $agent$ using Eq. (10)
      end if
    else if $p \geq 0.5$ **then**
      if $z \geq 0.5$ **then**
        Update $agent$ using VNS
      Else
        Update $agent$ using Eq. (8)
    end if
  end for
  Amend agent where goes beyond search space
  Calculate the fitness of each search agent
  Update X^* if any better solution
  $t \leftarrow t + 1$
return X^*

4.2 Hyperparameter testing

We tested the hyperparameter to find the best hyperparameter combination. Fig. 7 shows the result of the testing. We test 125 combinations of the number of iterations, number of agents, and VNS duration. The number of iterations value is assessed with 20, 40, 60, 80, and 100 values. The number of agents value is evaluated with values of 20, 40, 60, 80, and 100. The VNS duration value is assessed with values of 0.2 second, 0.4 second, 0.6 second, 0.8 second, and 1 second. From the testing, we got the best fitness value of 0.79785 in the following combination $number\_agent$ = 60, max_iter = 100, and $vns\_duration$ = 0.8 second. This combination will used in next testing.

4.3 Comparison of VRP and TSP based

We also compared multi-day tourist routes, analogous to TSP, and multi-day tourist routes, analogous to VRP. The test was repeated five times with 11 different random POIs, one hotel chosen at random, two days travel duration and all DOI set to 1. We compared WOA-VNS-VRP with WOA-VNS-TSP based on fitness value, number of POIs included, average trip duration, average total cost, and average rating. From the tests carried out, it is known that WOA-VNS-VRP, with an average fitness value of 0.8570, outperforms WOA-VNS-TSP, with an average fitness value of 0.7382.

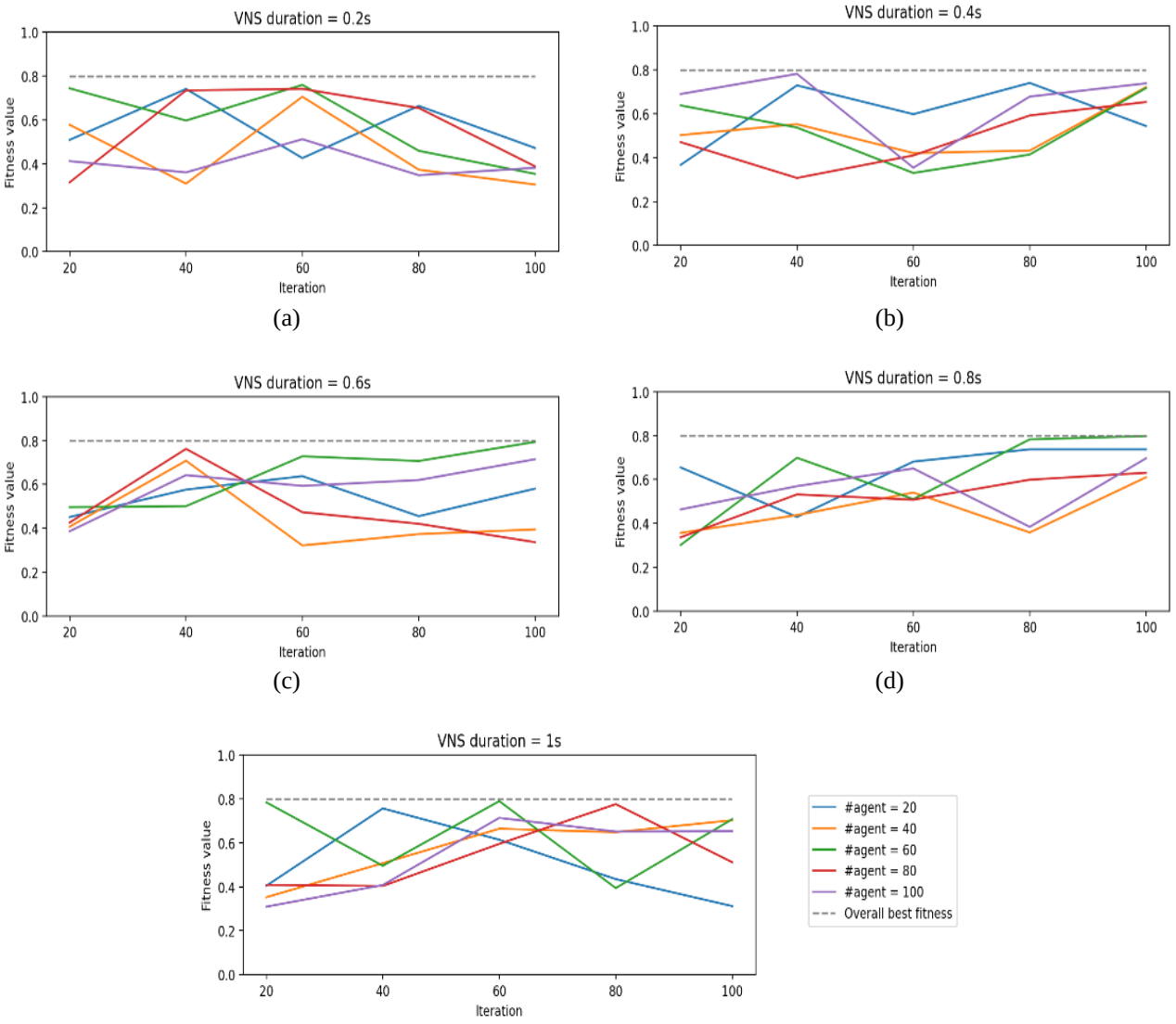

Figure. 5 Hyperparameter testing result: (a) VNS duration = 0.2s, (b) VNS duration = 0.4s, (c) VNS duration = 0.6s, (d) VNS duration = 0.8s, and (e) VNS duration = 1s,

According to Table 8, The average POI visited by WOA-VNS-VRP was 10 POIs, outperforming WOA-VNS-TSP, which only visited 9 POIs. In terms of the average travel duration and average total cost metric, WOA-VNS-VRP can minimize the average travel duration 4.83383 hours, and the average total cost of 106,000 IDR, compared to WOA-VNS-TSP, with an average travel duration of 5.9925 hours and an average total cost of 116,600 IDR. However, regarding the average rating matrix, each algorithm gives good results. Therefore, the VRP method is superior in producing multi-day tourist route plans compared to the TSP method.

4.4 Fitness value analysis

Experiments are conducted to test that the proposed WOA-VNS outperforms pure WOA and other conventional algorithms regarding fitness value as the primary metric. In this test, all DOIs in set 1 are used to prove that all attributes are considered. The test was repeated five times with 11 different random POIs; one hotel was randomly selected, and the trip duration was two days. Figure. 6 shows the average WOA, WOA-VNS, ACO, GA and BA fitness to solve VRP.

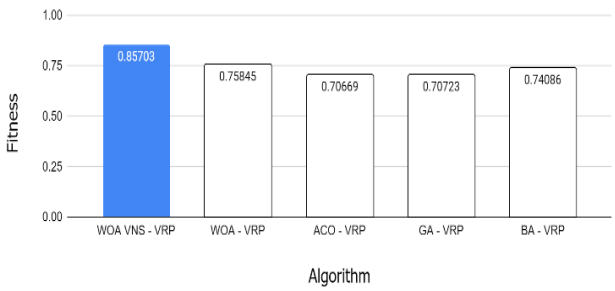

Figure. 6 Average fitness value

Table 8. WOA, ACO, and GA, and BA testing result

Metrics	Algorithm	Set of random POI					Average
		1st	2nd	3rd	4th	5th	
Fitness Value	WOA - VRP	0.7218	0.7903	0.7233	0.7472	0.8094	0.7584
	WOA VNS - TSP	0.6834	0.6908	0.8032	0.8093	0.7042	0.7382
	WOA VNS - VRP	0.7461	0.9832	0.8239	0.8023	0.9294	**0.8570**
	ACO - VRP	0.7023	0.8902	0.4392	0.6923	0.8092	0.7066
	GA - VRP	0.7645	0.7288	0.6934	0.6346	0.7146	0.7072
	BA - VRP	0.7093	0.7092	0.6923	0.7912	0.8021	0.7408
The number of POI included	WOA - VRP	9	9	9	9	8	8.8
	WOA VNS - TSP	8	9	9	9	10	9
	WOA VNS - VRP	9	12	10	9	10	**10**
	ACO - VRP	9	9	9	8	10	9
	GA - VRP	9	9	9	8	10	9
	BA - VRP	9	10	10	9	10	9.6
Travel duration (hours)	WOA - VRP	6.16833	3.18083	5.56917	6.19111	4.51472	5.12483
	WOA VNS - TSP	6.06917	4.37667	5.85417	9.03056	4.63194	5.9925
	WOA VNS - VRP	4.89944	3.64694	5.15583	6.38667	4.08028	**4.83383**
	ACO - VRP	5.39889	3.77028	5.50444	6.63861	4.36083	5.13461
	GA - VRP	5.085	4.465	7.92194	7.66667	4.28778	5.88528
	BA - VRP	5.38111	3.69056	5.4975	7.54528	4.44194	5.31128
Total cost (IDR)	WOA - VRP	155000	102500	128000	74500	143000	120600
	WOA VNS - TSP	162000	102500	98000	74500	146000	116600
	WOA VNS - VRP	155000	82500	128000	44500	120000	**106000**
	ACO - VRP	162000	82500	128000	44500	128000	109000
	GA - VRP	162000	102500	128000	74500	146000	122600
	BA - VRP	142000	100000	120000	74500	146000	116500
Average rating	WOA - VRP	4.5884	4.5863	4.5826	4.5796	4.5913	4.585707
	WOA VNS - TSP	4.5862	4.5934	4.5927	4.5793	4.5937	4.589075
	WOA VNS - VRP	4.5972	4.5837	4.5839	4.5837	4.5968	**4.589086**
	ACO - VRP	4.5815	4.5163	4.5136	4.5927	4.5983	4.560522
	GA - VRP	4.5222	4.5222	4.4888	4.5375	4.5599	4.526166
	BA - VRP	4.5835	4.5838	4.5812	4.5504	4.5835	4.576503

Based on fitness value as the primary matrix, WOA-VNS gets an average fitness value of 0.8570, outperforming pure WOA with an average fitness value of 0.7584 and also ACO, GA, and BA with average fitness values of 0.7066, 0.7072 and 0.7408. So, method WOA-VNS perform better than pure WOA, ACO, GA and BA.

4.5 Attribute analysis

The generated itinerary is influenced by several attributes, such as the number of POIs included, travel duration, total cost, and average rating. This study conducted experiments to evaluate the algorithm's effectiveness in optimizing those attributes.

4.5.1. Number of POI included attribute

The metrics used in this experiment are the number of POI-included attributes for two travel days. The proposed algorithm to optimize the number of POI-included attributes uses five random POI samples. All DOI (travel duration, cost and rating attribute) is set to 1. Fig. 9 shows the average number of POI included in the itinerary. Method WOA-VNS -VRP performs best with an average of 10 POI included. Methods BA performed with an average of 9.6 POI included, better than ACO and GA with an average of 9 POI included. The WOA method visited the fewest POIs with an average of 8.8 POI included, but after being optimized with VNS, it outperformed other algorithms.

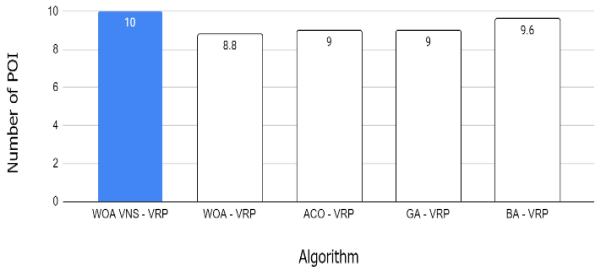

Figure. 9 Average number of POI included

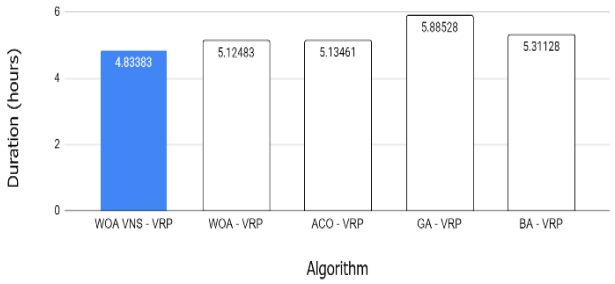

Figure. 10 Average travel duration

4.6.2. Travel duration attribute

The metrics used in this experiment are the total travel duration required to visit all POIs for two days of travel. The proposed algorithm for optimizing trip duration attributes uses five random samples of POI, with DOI for the trip duration attribute set to 1. In contrast, the other attributes (rating and cost) are set to 0.5. Fig. 10 shows the average itinerary duration from pure WOA, WOA-VNS, ACO, GA and BA methods to solve VRP. WOA-VNS produces the lowest average trip duration to visit all POIs, with an average travel duration of 4.83383 hours, outperforming pure WOA with an average total duration of 5.12483 hours, and also ACO, GA, and BA with an average total duration of 5.13461 hours, 5.88528 hours and 5.31228 hours respectively. Based on Fig. 9, WOA-VNS-VRP succeeded in visiting the most POIs compared to other comparison algorithms. In addition, WOA-VNS-VRP has the least trip duration, which proves that WOA-VNS-VRP is better at minimizing trip duration than other comparison algorithms.

4.5.3. Cost attribute

The metrics used in this experiment are the total cost needed to visit all POIs for two days of travel. The proposed algorithm for optimizing cost attributes uses five random samples of POI. The DOI for the cost attribute is set to 1, while the other attributes (rating and duration) are set to 0.5. Fig. 12 shows the average total cost from WOA, WOA-VNS, ACO, GA and BA methods. Method WOA-VNS-VRP performs best with an average total cost of 106,000 IDR, outperforms pure WOA, ACO, GA, and BA with average total costs of 120,600 IDR, 109,000 IDR, 122,600 IDR, and 116,500 IDR respectively. These results indicate that WOA-VNS-VRP is more effective in optimizing cost attributes than other comparison algorithms.

4.5.4. Rating attribute

The metrics used in this experiment are the average rating of all POI for two days of travel.

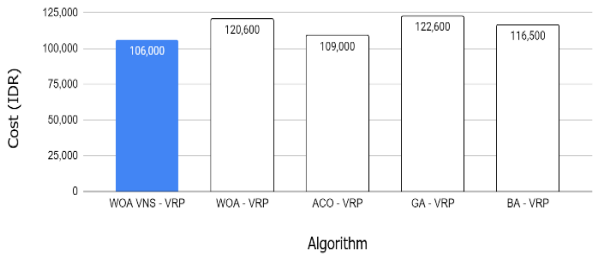

Figure. 11 Average total cost

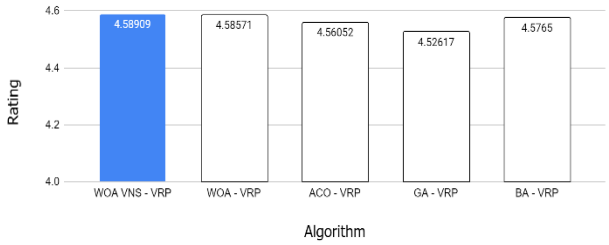

Figure. 12 Average rating

The proposed algorithm for optimizing rating attributes uses five random samples of POI. The DOI for the rating attribute is set to 1, while the other attributes (cost and duration) are set to 0.5. Fig. 12 shows average WOA, WOA-VNS, ACO, GA and BA algorithm ratings. Method WOA-VNS performs best, with an average rating of 4.58909. However, each algorithm has no significant difference; all algorithms give a good rating result. These results indicate that WOA-VNS-VRP is more effective in optimizing rating attributes than other comparison algorithms.

4.6 Running time comparison

Figure. 7 show running time comparison for WOA-TSP, WOA-VRP, WOA-VNS-TSP, and WOA-VNS-VRP. We used same parameter obtained from hyperparameter testing (except for *number_agent* hyperparameter). The WOA-VNS give longest running time in all value of number of POI. But it is reasonable because VNS take longer time to find better agent.

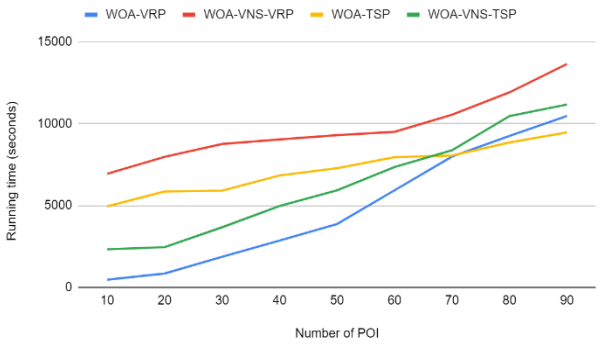

Figure. 7 Running time comparison

This shows that WOA-VNS-VRP still needs to improve in terms of time complexity. Despite this, WOA-VNS-VRP remains effective, considering that WOA-VNS-VRP excels in all five tested metrics.

5. Experiment result

In this research, we propose using the WOA-VNS algorithm to solve the Vehicle Routing Problem (VRP) in the context of multi-day tourist routes to overcome the limitations of TSP-based tourist routes. The VNS algorithm optimizes WOA by rearranging agents so that it can search for the best solution in a wider search space. Test results show that WOA-VNS produces consistently higher fitness values during iterations, while pure WOA produces fluctuating fitness values. We also understand the algorithm using five metrics: fitness value, number of included POIs, trip duration, total cost, and average rating. Based on test results, VRP is more suitable for multi-day itineraries because several vehicles can represent one day of travel. This is proven by WOA-VNS-VRP, which outperforms WOA-VNS-TSP on the five metrics tested, with a fitness value reaching 0.8570. In addition, the WOA-VNS algorithm also outperforms pure WOA and other conventional algorithms (ACO, GA, BA) in all five metrics. The average POI visited by WOA-VNS-VRP in two days was 10 POIs, with an average trip duration of 4.83383. Meanwhile, for WOA-VNS-TSP, pure WOA, ACO, GA, and BA, respectively, there were 9 POIs with a trip duration of 5.9925, 8.8 POIs with a trip duration of 5.12483, 9 POIs with a trip duration of 5.13461, 9 POIs with a trip duration of 5.88528, and 9.6 POIs with a trip duration of 5.31128. In terms of average total costs, WOA-VNS-VRP also has the lowest average total costs, namely 106000 IDR, compared to other algorithms: WOA-VNS-TSP of 116600 IDR, pure WOA of 120600 IDR, ACO of 109000 IDR, GA is IDR 122600, and BA is IDR 116500. Regarding average rating, WOA-VNS-VRP still gets the highest average rating, 4.589086. Despite this, all algorithms provide good ranking results without significant differences. This model's advantage is that it can provide a more optimal route for each day of several tourist visits. Calculating fitness values with MAUT also makes the resulting routes according to user preferences. However, regarding running time, WOA-VNS takes the longest to find the best solution because VNS takes longer to find a better agent. Future research can improve the running time and apply this model to mobile devices to make it easier for users. In addition, future research can add new obstacles, such as culinary tourism, to produce more complete tourist routes. For example, special culinary tourism visits are made during lunchtime between 12.00 and 14.00 and dinnertime between 18.00 and 20.00.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, Saskia Putri Ananda; Z. K. A. Baizal; methodology, Saskia Putri Ananda, Z. K. A. Baizal, Gia Septiana Wulandari and; software, Saskia Putri Ananda, Z. K. A. Baizal; validation, Saskia Putri Ananda, Z. K. A. Baizal, Gia Septiana Wulandari; formal analysis, Saskia Putri Ananda; investigation, Saskia Putri Ananda; data curation, Saskia Putri Ananda, Z. K. A. Baizal; writing—original draft preparation, Saskia Putri Ananda; visualization, Saskia Putri Ananda; writing—review and editing, Saskia Putri Ananda; Z. K. A. Baizal, Gia Septiana Wulandari; supervision, Z.K.A. Baizal.

References

[1] A. M. Mangini, M. Roccotelli, and A. Rinaldi, "A novel application based on a heuristic approach for planning itineraries of one-day tourist", *Appl. Sci.*, Vol. 11, No. 19, 2021, doi: 10.3390/app11198989.

[2] X. Mao, "Study on ant colony optimization algorithm for 'one-day tour' traffic line", *Cluster Comput.*, Vol. 22, pp. 3673-3680, 2019, doi: 10.1007/s10586-018-2217-9.

[3] Z. K. A. Baizal, A. A. Rahmawati, K. M. Lhaksmana, M. Z. Mubarok, and M. Qadrian, "Generating travel itinerary using ant collony optimization", *Telkomnika (Telecommunication Comput. Electron. Control.*, Vol. 16, No. 3, pp. 1208-1216, 2018, doi: 10.12928/TELKOMNIKA.v16i3.7268.

[4] Z. K. A. Baizal, K. M. Lhaksmana, A. A. Rahmawati, M. Kirom, and Z. Mubarok, "Travel route scheduling based on user's preferences using simulated annealing", *Int. J. Electr. Comput. Eng.*, Vol. 9, No. 2, pp. 1275-1287, 2019, doi: 10.11591/ijpeds.v9i2.pp1275-1287.

[5] M. Anranur Uwaisy, Z. K. A. Baizal, and M. Yusza Reditya, "Recommendation of scheduling tourism routes using tabu search method (case study bandung)", *Procedia Computer Science*, 2019, Vol. 157, pp. 150-159, doi: 10.1016/j.procs.2019.08.152.

[6] B. Eksioglu, A. V. Vural, and A. Reisman, "The vehicle routing problem: A taxonomic review",

Computers and Industrial Engineering, Vol. 57, No. 4. pp. 1472-1483, 2009, doi: 10.1016/j.cie.2009.05.009.

[7] A. Chandra and B. Setiawan, "Optimasi Jalur Distribusi dengan Metode Vehicle Routing Problem (VRP) Optimizing the Distribution Routes Using Vehicle Routing Problem (VRP) Method", *Jurnal Manajemen Transportasi & Logistik*, Vol. 5, No. 2, 2018, doi: 10.54324/j.mtl.v5i2.233.

[8] B. Pan, Z. Zhang, and A. Lim, "Multi-trip time-dependent vehicle routing problem with time windows", *Eur. J. Oper. Res.*, Vol. 291, No. 1, pp. 218-231, 2021, doi: 10.1016/j.ejor.2020.09.022.

[9] C. Chen, E. Demir, and Y. Huang, "An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and delivery robots", *European journal of operational research*, Vol. 294, No. 3, pp. 1164-1180, 2021, doi: 10.1016/j.ejor.2021.02.027.

[10] R. Hendrawan, Z. K. A. Baizal, and G. S. Wulandari, "Generating a Multi-Day Travel Itinerary Recommendation Using the Hybrid Ant Colony System and Brainstorm Optimization Algorithm", *Int. J. Intell. Eng. Syst.*, Vol. 17, No. 2, pp. 223-234, 2024, doi: 10.22266/ijies2024.0430.20.

[11] R. Schäfer, "Rules for Using Multi-Attribute Utility Theory for Estimating a User's Interests", In: *Proc. of Ninth Workshop Adaptivität und Benutzermodellierung in Interaktiven Softwaresystemen*, pp. 8-10, 2001.

[12] J. Figueira, S. Greco, and M. Ehrgott, *Multiple Criteria Decision Analysis: State of the Art Surveys*, Springer Science & Business Media, 2005.

[13] M. Alemi-Rostami and G. Rezazadeh, "Selective harmonic elimination of a multilevel voltage source inverter using whale optimization algorithm", *Int. J. Eng. Trans. B Appl.*, Vol. 34, No. 8, 2021, doi: 10.5829/ije.2021.34.08b.11.

[14] L. Liu and R. Zhang, "Multistrategy Improved Whale Optimization Algorithm and Its Application", *Comput. Intell. Neurosci.*, Vol. 2022, 2022, doi: 10.1155/2022/3418269.

[15] Y. Gao, H. You, and J. Xu, *Adaptive Whale Optimization Algorithm with simulated annealing strategy and Its Application in Magnetic Target Location*. 2021.

[16] "Whale optimization algorithm based on Gaussian mutation and differential evolution", *Acad. J. Comput. Inf. Sci.*, Vol. 5, No. 6, 2022, doi: 10.25236/ajcis.2022.050602.

[17] Y. Li, X. Zhang, J. Zhao, X. Yang, and M. Xi, "Position Deployment Optimization of Maneuvering Conventional Missile Based on Improved Whale Optimization Algorithm", *Int. J. Aerosp. Eng.*, Vol. 2022, 2022, doi: 10.1155/2022/4373879.

[18] H. Bali *et al.*, "Multi-Objective Energy Efficient Adaptive Whale Optimization Based Routing for Wireless Sensor Network", *Energies*, Vol. 15, No. 14, 2022, doi: 10.3390/en15145237.

[19] R. S. Moorthy and P. Parameshwaran, "An Optimal K-Nearest Neighbor for Weather Prediction Using Whale Optimization Algorithm", *Int. J. Appl. Metaheuristic Comput.*, Vol. 13, No. 1, 2021, doi: 10.4018/ijamc.290538.

[20] C. Wang, M. Li, R. Wang, H. Yu, and S. Wang, "An image denoising method based on BP neural network optimized by improved whale optimization algorithm", *EURASIP J. Wirel. Commun. Netw.*, Vol. 2021, No. 1, 2021, doi: 10.1186/s13638-021-02013-2.

[21] V. Neenavath and B. T. Krishna, "An energy efficient multipath routing protocol for manet", *J. Eng. Res.*, 2022.

[22] A. Adhi, B. Santosa, and N. Siswanto, "Hybrid Metaheuristics for Solving Vehicle Routing Problem in Multi Bulk Product Shipments with Limited Undedicated Compartments", *Int. J. Intell. Eng. Syst.*, Vol. 14, No. 5, 2021, doi: 10.22266/ijies2021.1031.29.

[23] T. Azad, H. F. Rahman, R. K. Chakrabortty, and M. J. Ryan, "Optimization of integrated production scheduling and vehicle routing problem with batch delivery to multiple customers in supply chain", *Memetic Comput.*, Vol. 14, No. 3, 2022, doi: 10.1007/s12293-022-00372-x.

[24] I. Sbai, S. Krichen, and O. Limam, "Two meta-heuristics for solving the capacitated vehicle routing problem: the case of the Tunisian Post Office", *Oper. Res.*, Vol. 22, No. 1, pp. 507-549, 2022, doi: 10.1007/s12351-019-00543-8.

[25] T. Erdelić and T. Carić, "Goods Delivery with Electric Vehicles: Electric Vehicle Routing Optimization with Time Windows and Partial or Full Recharge", *Energies*, Vol. 15, No. 1, 2022, doi: 10.3390/en15010285.

[26] L. do C. Martins, E. M. Gonzalez-Neira, S. Hatami, A. A. Juan, and J. R. Montoya-Torres, "Combining production and distribution in supply chains: The hybrid flow-shop vehicle routing problem", *Comput. Ind. Eng.*, Vol. 159, p. 107486, 2021, doi: 10.1016/j.cie.2021.107486.

[27] Q. Zhang and L. Liu, "Whale optimization

algorithm based on lamarckian learning for global optimization problems", *IEEE Access*, Vol. 7, 2019, doi: 10.1109/ACCESS.2019.2905009.

[28] A. H. Ismail, N. Hartono, S. Zeybek, M. Caterino, and K. Jiang, "Combinatorial Bees Algorithm for Vehicle Routing Problem", *Macromol. Symp.*, Vol. 396, No. 1, 2021, doi: 10.1002/masy.202000284.

[29] A. E. Hegazy, M. A. Makhlouf, and G. S. El-Tawel, "Dimensionality Reduction Using an Improved Whale Optimization Algorithm for Data Classification", *Int. J. Mod. Educ. Comput. Sci.*, Vol. 10, No. 7, pp. 37-49, 2018, doi: 10.5815/ijmecs.2018.07.04.

[30] J. M. Abdullah and T. Ahmed, "Fitness Dependent Optimizer: Inspired by the Bee Swarming Reproductive Process", *IEEE Access*, Vol. 7, 2019, doi: 10.1109/ACCESS.2019.2907012.

[31] H. M. Mohammed, S. U. Umar, and T. A. Rashid, "A Systematic and Meta-Analysis Survey of Whale Optimization Algorithm", *Computational Intelligence and Neuroscience*, Vol. 2019. 2019, doi: 10.1155/2019/8718571.

[32] Y. Li, T. Han, H. Zhao, and H. Gao, "An adaptive whale optimization algorithm using gaussian distribution strategies and its application in heterogeneous ucavs task allocation", *IEEE Access*, Vol. 7, 2019, doi: 10.1109/ACCESS.2019.2933661.

[33] A. Abudayor and Ö. U. Nalbantoğlu, "A NOVEL HYBRID ALGORITHM BASED ON CROW SEARCH ALGORITHM AND WHALE OPTIMIZATION ALGORITHM FOR HIGH-DIMENSIONAL OPTIMIZATION AND FEATURE SELECTION", *Indian J. Comput. Sci. Eng.*, Vol. 14, No. 2, 2023, doi: 10.21817/indjcse/2023/v14i2/231402050.

[34] A. N. Mat, O. Inan, and M. Karakoyun, "An application of the whale optimization algorithm with Levy flight strategy for clustering of medical datasets", *Int. J. Optim. Control Theor. Appl.*, Vol. 11, No. 2, 2021, doi: 10.11121/IJOCTA.01.2021.001091.

[35] M. Zhong and W. Long, "Whale optimization algorithm with nonlinear control parameter", in *MATEC Web of Conferences*, 2017, Vol. 139, doi: 10.1051/matecconf/201713900157.

[36] M. Abdel-Basset, G. Manogaran, D. El-Shahat, and S. Mirjalili, "A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem", *Future Generation Computer Systems*, Vol. 85. Elsevier B.V., pp. 129-145, 2018, doi: 10.1016/j.future.2018.03.020.

[37] W. Jiang, R. Hu, B. Qian, N. K. Yu, and B. Liu, "Hybrid Whale Optimization Algorithm for Solving Green Open Vehicle Routing Problem with Time Windows", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2021, Vol. 12836 LNCS, pp. 673-683, doi: 10.1007/978-3-030-84522-3_55.

[38] J. Zhang, L. Hong, and Q. Liu, "An improved whale optimization algorithm for the traveling salesman problem", *Symmetry (Basel).*, Vol. 13, No. 1, pp. 1-13, 2021, doi: 10.3390/sym13010048.

[39] V. C. Hemmelmayr, K. F. Doerner, and R. F. Hartl, "A variable neighborhood search heuristic for periodic routing problems", *Eur. J. Oper. Res.*, Vol. 195, No. 3, 2009, doi: 10.1016/j.ejor.2007.08.048.

[40] D. Satyananda and S. Wahyuningsih, "Sequential order vs random order in operators of variable neighborhood descent method", *Telkomnika (Telecommunication Comput. Electron. Control.*, Vol. 17, No. 2, 2019, doi: 10.12928/TELKOMNIKA.V17I2.11789.

[41] J. Zheng, "A Vehicle Routing Problem Model with Multiple Fuzzy Windows Based on Time-Varying Traffic Flow", *IEEE Access*, Vol. 8, 2020, doi: 10.1109/ACCESS.2020.2974774.

[42] V. Lesch, M. König, S. Kounev, A. Stein, and C. Krupitzer, "Tackling the rich vehicle routing problem with nature-inspired algorithms", *Appl. Intell.*, Vol. 52, No. 8, 2022, doi: 10.1007/s10489-021-03035-5.

[43] H. Xu, F. Duan, and P. Pu, "Solving dynamic vehicle routing problem using enhanced genetic algorithm with penalty factors", *Int. J. Performability Eng.*, Vol. 14, No. 4, 2018, doi: 10.23940/ijpe.18.04.p3.611620.

[44] T. A. S. Masutti and L. N. De Castro, "Bee-Inspired Algorithms Applied to Vehicle Routing Problems: A Survey and a Proposal", *Mathematical Problems in Engineering*, vol. 2017. 2017, doi: 10.1155/2017/3046830.

[45] Y. Li, Q. Guo, and J. Liu, "Improved bat algorithm for Vehicle routing problem", *Int. J. Performability Eng.*, Vol. 15, No. 1, 2019, doi: 10.23940/ijpe.19.01.p32.317325.