

Multi-objective ride-sharing optimization method based on the narwhal optimizer and ant colony optimization

Mohammad Tubishat · Ahmed Shuhaiber · Malik Braik ·
Mohammed Azmi Al-Betar

Received: date / Accepted: date

Abstract Meta-heuristic optimization algorithms have gained popularity due to their exceptional features such as gradient-free mechanisms, high flexibility, and significant potential to avoid local optimum solutions. The emergence of on-demand taxi services has led to a heavy reliance on taxis for public transportation in urban areas. To reduce traffic congestion in urban areas and increase the efficiency of taxi transportation, ride-sharing is a crucial method that reduces operating costs and conserves road resources. This paper explores the use of a combination of Discrete Narwhal Optimizer (DNO) and Ant Colony Optimization (ACO), referred to as DNOACO, to optimize ride-sharing taxi routes while taking into consideration the interests of both drivers and passengers. This presents a completely distributed ride sharing method that uses asynchronously localized communication between taxis and passengers to solve the dynamics of taxi topology. On-the-spot ride scheduling is difficult, though, and becomes even more difficult as the number of passengers increases. In in-

tricate urban systems, passengers must utilize a variety of modalities and vehicles to get to their destination. For these passengers, selecting the best route is a challenging optimization problem. This work considers a distributed ride-sharing network that has a taxi routing problem, which is a transportation network problem. Optimization of operational costs and customer satisfaction are presented as the objective criterion, and waiting time, travel distance, and additional riding time as a result of ride-sharing are utilized to measure them respectively. After establishing a routing optimization model for ride-sharing taxis, the proper DNO algorithm that satisfies the model's requirements is created. The model is finally verified by a computational experiment, and the outcome indicates that this approach has promise. The evaluation results of DNOACO are contrasted with those of other algorithms using the same approach of dynamic shared taxi. In a lesser number of iterations and a significantly shorter amount of time, the DNOACO algorithm can reach an optimum solution due to its rapid rate of convergence. Using extensive single-user taxi trip data, empirical study demonstrates that the proposed method guarantees a maximum of 90% ride-sharing success and a 97.5% taxi occupancy rate throughout peak hours.

Keywords Optimization · Meta-heuristics · Narwhal optimizer · Ant Colony Optimization · Ride sharing

Mohammad Tubishat
College of Technological Innovation
Zayed University, Abu Dhabi, UAE
E-mail: Mohammad.Tubishat@zu.ac.ae

Ahmed Shuhaiber
College of Technological Innovation
Zayed University, Abu Dhabi, UAE
E-mail: ahmed.shuhaiber@zu.ac.ae

Malik Braik
Al-Balqa Applied University
Department of Computer Science
E-mail: mbraik@bau.edu.jo

Mohammed Azmi Al-Betar (Corresponding Author)
Artificial Intelligence Research Center (AIRC)
College of Engineering and Information Technology
Ajman University, Ajman, UAE
E-mail: m.albetar@ajman.ac.ae

1 Introduction

Optimization problems are commonplace in our daily lives and work. The need for effective and efficient solutions to tackle optimization problems has increased significantly in the scientific literature [1]. For a particular optimization problem, optimization must sat-

isfy some requirements to uncover the optimal solution or a suitable approximation from among several alternatives. The rapid development of new technologies is increasing the prevalence and complexity of optimization problems in a wide range of engineering fields, such as industrial modeling [2], production design [3], vehicle routing [4], pattern recognition applications [5], and many more. Optimization can save computing resources and costs, significantly improve problem-solving efficiency, and lessen the related computational load. Meta-heuristic methods can be classed as a wide range of optimization strategies used to address optimization problems [6]. In a variety of application areas and computational intelligence, meta-heuristics have become well-known and reliable techniques. These methods can successfully handle complex optimization problems, which can be classified as NP-hard. These adaptable methods have been successfully used in a variety of domains, including medical diagnostics [7], image processing [8], and others. Because they can be very effective at addressing certain problems but may not be able to give satisfying solutions to others, meta-heuristics are very reliant on the specific problem [9]. This can be partly explained by the fact that these techniques often end up stumbling upon local or suboptimal solutions [10].

As per the above discussion points, meta-heuristic optimization has become increasingly popular among researchers [6, 1]. Techniques are taught theoretically and made accessible to anyone, which speeds up research in this area. These strategies are meant to make current solutions more effective [1]. The primary meta-heuristics are single-objective and multi-objective optimization strategies. Recent single meta-heuristic algorithms include Stochastic Paint Optimizer (SPO) [11], Capuchin Search Algorithm (CSA) [12], Ant Colony Optimization (ACO) [13], Tornado Optimizer (TO) [10], and Genetic Algorithm (GA) [14]. Several academic disciplines employ the meta-heuristic algorithms [15, 9]. Furthermore, some multi-objective meta-heuristic algorithms include the Multi-objective Bonobo Optimizer (MOBO) [16], Advanced Neural Network Algorithm (ANNA) [17], Multi-objective material generation algorithm (MOMGA) [18], and Multi-Objective Crystal Structure Algorithm (MOCryStAl) [19].

The urban social economy is developing rapidly, which means that the transport needs of residents are growing and that the cost of a taxi is increasing. One of the main drivers of urban traffic is the taxi yet demand and capacity for transportation may not always coincide during peak and off-peak hours. In the off-peak hours, it is frequently empty loaded because of the high running costs; in the busy hours, it is frequently challenging to hail a taxi because many of them only hold one pas-

senger. Passengers' travel expenses and time may increase due to the excessive detour phenomena caused by traditional taxi ride-sharing behavior. The benefits of ride-sharing taxis include intelligent scheduling and route decision making, which can provide more affordable rates and lower overall operating expenses. This can increase the enthusiasm of drivers and passengers, make efficient use of traffic resources, lower tail gas emissions, safeguard the environment, and encourage the sustainable growth of urban traffic. For ride-hailing, the optimization model and method of the ride-sharing route must be studied to determine the shortest ride-sharing route. Ride-sharing often has no health problems. However, in the meantime, precautions must be taken throughout the epidemic time, such as providing masks for both drivers and passengers and isolating the interior of the cabs with plastic sheeting. Despite the absence of ride-sharing behavior, precautions are still necessary in the event of an epidemic because of the possible safety risks [20].

One of the common issues is routing, which Hamilton addressed for the first time in Traveling Salesman [21]. To facilitate transit inside cities, urban transportation is now carried out through a multimodal transportation network. As a result, urban tourists now find it challenging to navigate. Subway, bus, taxis, and other modes can all be included in multi-modal transportation networks [22]. Every traveler in the urban multi-modal network may have a different preferred path. When choosing a route, travelers consider a number of factors, including traffic, comfort, and safety, in addition to the quickest route. There may be a mismatch of goals when more convenient routes are less safe. According to [23], multi-modal routing is a multi-objective optimization problem because of the misalignment of the objectives and diverse involved objectives. Even in the mono-objective situation, the multi-objective routing problem has many potential solutions, making it a complex optimization problem of the non-deterministic polynomial-hard order [24]. Classical techniques are unable to solve this type of problem in an acceptable amount of time, since there are many potential solutions, which makes the search space in this problem extremely vast [25]. Due to the necessity of solving complex problems with a large search space, several studies have been conducted on optimization methods [26]. To tackle complicated optimization problems, meta-heuristic algorithms based on swarm and population intelligence have been created, drawing inspiration from nature [27].

To address the routing problem and determine the best path, a lot of study has been done [28]. When the network is operating under mono-modal and mono-objective conditions, the routing issue is at its most

straightforward. Exact and stochastic methods are employed to solve routing problems.

In stochastic techniques, a pool of solutions is offered for the desired problem, and near-optimal (not quite optimum) solutions will be created, whereas just one exactly right solution is determined at the conclusion of exact methods. Dijkstra is a well-known and decisive approach to routing problems, and it has been the subject of several investigations [29]. To solve a multi-modal routing problem using the Dijkstra algorithm, Kheirikharzar et al. [30] took into account three objective functions: cost, time, and the number of mode changes. The method was executed once for every objective using the Pareto solution set. It is not possible to solve routing problems with a large transportation network in an acceptable amount of time using the traditional Dijkstra method. There are other precise techniques that have been extensively studied, including Bellman-Ford, Breadth-first Search (BFS), Depth-first Search (DFS), and A*. The weighted graph network is the foundation of all these techniques, and in the final form, one path will be returned [31]. It has been unsatisfactory to solve routing problems using traditional algorithms in many situations, particularly when the problem involves a big network with bulky data [32]. Many studies have applied fuzzy logic to multi-modal routing problems [33]. The accuracy of the solutions obtained while using fuzzy logic to solve problems is dependent on the amount of modeling knowledge, fuzzy logic, and experience of the individual [26]. Furthermore, the problem's solution would be more difficult in big multi-modal networks because of the network's size [34]. As a result, several field investigations on routing difficulties now use meta-heuristic algorithms considering the problems mentioned above. Numerous studies have been conducted to use the particle swarm optimization (PSO) method to address the routing problem, including [35]. In these studies, the routing problem is intended to be mono-modal and mono-objective. Additionally, the PSO method is continuous, but the routing problem is inherently discrete. Eberhart used a binary statement to subtly introduce the fundamental PSO algorithm [36].

Although the urban multi-modal network routing problem involves a complex space, the use of binary coding to describe it significantly increases the computational complexity [37]. Accordingly, routing in big networks is not a good fit for the PSO binary [38].

1.1 Meta-heuristic Algorithms

Meta-heuristic algorithms (MAs) are frequently used to solve complicated problems in contemporary society,

including dynamic optimization of ride-sharing taxis, with the goal of achieving acceptable performance levels. MAs have drawn the attention of several academic researchers working to optimize ride-sharing taxi systems [39]. By employing precisely calibrated learning variables to explore and use the search space of ride-sharing systems, these algorithms provide an optimization framework that can efficiently search for optimum solutions [39]. MAs have several benefits and may be applied successfully to a variety of optimization problems across multiple fields. Among the most well-known optimizations of multi-objective algorithms is the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [40] algorithm. It differs from the genetic algorithm in that it has several objectives. Deb et al. introduced NSGA-II, an elite evolutionary algorithm [40]. The crowding distance notion in the algorithm has made NSGA-II superior to genetic multi-objective algorithms [41], and it is not restricted in the number of optimization goals. The PSO algorithm prioritizes flexibility and its accessible operators above other algorithms, such as genetic and ant algorithms [42]. Coello et al. [43] were the first to create multi-objective particle swarm optimization (MOPSO), the multi-objective instance of the PSO method. The MOPSO method produces Pareto-fronts of optimum solutions, from which the user may select the optimal one based on his preferences. The MOPSO method was used by Masoumi et al. [44] to solve the land-use allocation problem in 2012. They used the continuous MOPSO method to solve the discrete problem by mapping. 34 land uses, numbered 1 through 34, are included in this study.

1.2 Motives of this Study

Existing studies recognize that no single, universal optimization technique guarantees the best possible solution or solves all optimization problems with superior performance. Therefore, advances can improve robustness in finding more accurate results for related optimization problems. Additionally, many previous studies published in literature on the optimization of ride-sharing taxi systems have achieved good results in many cases. However, in other cases, they have yielded sub-standard performance. Based on the shortcomings of existing methods, this study presents a reliable meta-heuristic algorithm specifically designed for dynamic ride-sharing taxi optimization. This meta-heuristic algorithm seeks to outperform existing optimization methods described in literature. The system optimization is the main focus of a large body of work on the ride-sharing optimization problem. Drivers' and passengers' interests are directly influenced by ride-sharing costs,

but because they are playing a game with one another, the straightforward cost-sharing approach is typically used, which raises questions regarding fairness. Establishing a reasonable ride-sharing route optimization structure for ride hailing that improves ride-sharing's rationality while accounting for user fairness and system optimization is difficult, though. As a result, while some research has been done on optimizing routes for taxi ride-sharing, most of it has been on the method itself. To tackle this problem effectively, many operators or algorithms are developed. In modeling, the concepts of user fairness and system optimization are rarely considered. Additionally, drivers and passengers who use ride-sharing services are not subject to any constraints.

1.3 Outlines of this Study

The multi-objective routing problem in the current study is in an urban multi-modal transport system that is as vast as a multi-modal transportation network. The vehicle routing problem for ride-sharing taxis—an on-demand transportation service problem—is addressed in this work. The queue approach, which sends requests to the taxi with the fewest passengers, is the current trend. The ride-sharing system assists in connecting passengers' requests for rides with available taxis. The taxis can carry several passengers on comparable routes, which may or may not overlap in terms of origin and destination. It falls under a subclass of "dial-a-ride" problems, which can be either dynamic or static [45]. The source and destination locations, the available cars, and the trip requests are all known ahead of time in static ride sharing [46]. The system contains all the data needed to process the requests using either a heuristic-based (approximation) method or a non-heuristic approach. In contrast, dynamic ride-sharing systems pair drivers and passengers in real-time as requests come in and are unaware of any such information [47].

The dynamic system is challenged by time-bound constraints and the lack of prior knowledge about incoming requests and taxis. There are three possible models for dynamic ride-sharing systems: central, distributed, and hybrid. Following an actual analysis of the consequences of the taxi network topology, we present in this study a distributed taxi ride sharing algorithm that can handle its sheer dynamic nature. This study analyzes the outcomes for various passenger counts per trip by simulating the current queue system and the novel approach for taxi assignments. DNOACO, a multi-objective Discrete Narwhal Optimizer (NO) [48] paired with Ant Colony Optimization (ACO) technique, was developed to handle this discrete problem by utilizing

the discrete search motion of narwhals. Thus, the multi-objective NO method has been advanced to the discrete state by this study. The outcomes from the promoted methods are assessed and contrasted with Genetic Algorithm-Ant Colony Optimization (GAACO), Multi-Objective Particle Swarm Optimization (MOPSO), Multi-Objective Artificial Bee Colony (MOABC) [49], Multi-Objective Grasshopper Optimization Algorithm (MOGOA), Multi-Objective Gray Wolf Optimization (MOGWO) [50], as well as the Multi-Objective Artificial Bee Colony (MOABCO) algorithm [51].

The primary justification for selecting these algorithms is that many of them are intrinsically discrete, making them appropriate for solving this problem. Additionally, other studies indicate that they work well for routing problems.

1.4 Contributions of this Study

Given the aforementioned deficiencies, this work adds four new insights to the body of knowledge on the vehicle routing problem for ride-sharing taxis:

- We have created an asynchronous distributed method to effectively manage the highly volatile dynamics in a distributed ride-sharing environment.
- The scalability of the proposed method has been demonstrated by testing it on a random sample of passengers and taxis. It operates through localized communication between passengers and taxis.
- The optimization model bridges the gap in the current optimization paradigm for a ride-sharing route by considering the advantages of both drivers and passengers to guarantee the excitement of ride-sharing.
- Using a very large dataset of individual taxi ride records, the proposed algorithm has been extensively tested to assure a maximum success rate of 76% in ride sharing.

The rest of this paper is structured as follows. A review of popular ride-sharing systems is described in Section 2. The problem description and model assumptions for the ride-sharing route optimization model are provided in Section 3. The optimization model is addressed using the narwhal optimizer and ant colony optimization, which are briefly described in Sections 4 and 5, respectively. The optimization model is built in Section 6. After that, a case study is carried out in Section 7 to confirm the optimization impact and applicability of the ride-sharing route optimization model. Future outlooks and conclusions are presented in the next Section 8.

2 Related Works

The use of numerical optimization methods for shared taxi systems has become easier due to the increasing availability of robust and reasonably priced computer resources. Existing ride-sharing systems may be divided into “static” and “dynamic” categories [45].

2.1 Static Ride Sharing

Static ride sharing involves communicating each rider’s origin and destination in advance, after which a reasonable final route is determined. Framed as integer programming, these problems are solved with a variety of non-heuristic (branch-and-bound algorithm [46]) and meta-heuristic algorithms, including the simulated annealing algorithm [52], the local search heuristic algorithm [53], and dynamic programming for route generation [54]. Although the static solutions behaved effectively for some ride-sharing models (carpooling), they are not very flexible when information is unavailable before processing or reaction times are shortened.

2.2 Dynamic Ride Sharing: Centralized Method

Since the information may be readily given to the central system, the centralized models incorporated actual road networks and city maps. Real taxi traffic data was contributed by Mustafa and Hua [55] and Chen et al. [56]. Additionally, Stiglic et al. [57] integrated ride-sharing platforms with public transportation to make the changeover easier for passengers. A centralized solution involves the central server managing all incoming requests and quickly matching them with available taxis. Since the system must respond rapidly, Cao et al. [58] proposed the SHAREK system, a scalable and effective ride-sharing service, which features early pruning techniques that use Euclidian and Semi-Euclidean distances for shortest path computations in stages rather than actual distances. Few strategies prioritized speedier ride matching above delivering the best possible match [59].

2.3 Dynamic Ride Sharing: Distributed Method

A distributed ride-sharing system removes the central processing bottleneck by allowing passengers and taxis to communicate via message passing rather than depending on a single entity to establish the connection. Pedro et al. [60] presented a new distributed and dynamic ride-sharing system and said that by dividing the

intricate calculations between the vehicle and passenger levels, processing times are reduced. To send and receive requests and responses, wireless technology is utilized. A similar strategy was put out by Bathla et al. [61], which makes advantage of localized communication between passengers and neighboring taxis. It manages the incoming requests with no serious delay and the passengers confirmed that the taxi is already serving. Large-scale GPS traces of 4,000 taxis in Shanghai, China, were used for testing. Both strategies effectively raised the taxis’ earnings and the passengers’ cost savings. Using vehicles with high-quality sensor nodes, Sharma, Seo, and Jong [62] and Zao et al. [63] presented a vehicular network for communications (peer-to-peer communication). Relay vehicles were another feature of the system that allowed messages to be sent to the appropriate vehicles. However, more sensors, processing power, and compatibility with many transmission protocols are necessary for a distributed system (802.11p).

2.4 Dynamic Ride Sharing: Hybrid Approach

In order to overcome the limitations of the previously created systems, which were either limited to a central server or a small number of mobile clients, hybrid models were proposed. Dimitrijević et al. [64] proposed developing a system that uses cloud distribution, parallel processing, and a message queue to store requests and answers. The data is kept in the distributed cloud environment across several database instances. Using cloud computing services like Microsoft Azure and Hadoop’s MapReduce, Ma et al. [65] and Masayo et al. [66] developed mobile cloud architecture.

2.5 Existing Models

As the world’s population continues to rise and major cities expand at an accelerated rate, urban residents’ travel preferences and frequency have changed significantly. This is particularly true given the growing need for mobility and privacy when driving [67]. Car ownership is on the rise in major cities worldwide, and the ensuing issues with environmental pollution and urban traffic congestion have become major roadblocks to the growth of these cities. There are four or more seats in a car, yet most private cars drive alone. Without a doubt, empty seats are a waste of the dwindling transportation resources, and sharing a car with several passengers is a keyway to make efficient use of those resources. Online ride splitting and private vehicle ride sharing are only two of the shared transportation services provided by transportation network businesses like Uber, Lyft,

Dida, and Didi. The conventional taxi paradigm, in which drivers work for transportation network corporations and aim to turn a profit, is the basis for online ride splitting.

In contrast, private car ride sharing, also referred to as peer-to-peer ride sharing, involves both drivers and passengers using transportation network firms. Drivers in private car ride-sharing are more concerned with reducing travel expenses than they are with making a profit. Because drivers in private car ride-sharing do not have to give up their travel time to pick up riders, this feature makes private car ride-sharing more effective than online ride splitting in terms of lowering vehicle trip distance.

With the rise in commutes, strategies to make travel easier have been devised. These strategies include increasing the number of vehicles, creating a variety of vehicle types, and creating efficient travel methods, such as dynamic ride sharing. With the advent of Uber Pool, Ola Share, and Lyft Line, taxi services such as Uber, Ola, and Lyft have made travel more affordable than hiring a private taxi. Traditional carpooling is not the same as the shared taxi model of today. A passenger may request a shared taxi at any time, and there is no set time limit or need that you know the driver or person escorting you.

This approach lowers the cost of transportation. Simulations were conducted to establish dynamic ride sharing. Pick-up and drop-off may be coupled on an existing route without altering the ongoing journey owing to the Uber Pool and Lyft Line schemes. Several criteria are taken into consideration when assigning a taxi to a passenger. For example, a taxi with the fewest passengers might be sent to the new passenger without taking the distance or the route into account. Assigning the taxi to someone inside a hail able radius is an alternative strategy. The taxis will be distributed throughout a map, and the system will look for the taxi within a certain radius upon receiving a request. In Figure 1, the prospective for dynamic ride-sharing is quantified.

In Figure 1, the blue circle indicates the pick-up location for passenger A, the orange circle indicates the pick-up location for passenger B, and the yellow squares in the circle represent taxis. According to this, the idea of a hail able region for pick-up places the taxi closest to the passenger's pickup location inside that area. The simulation runs and looks for the taxi in passenger B's location when a simultaneous request from passenger A appears in a nearby area. There is only one car accessible within B's radius, which is also the closest one to A, as can be seen in the figure. In a scenario like this, this car gets assigned to B, and since there are other cars in A's area that are closer, A will be assigned to

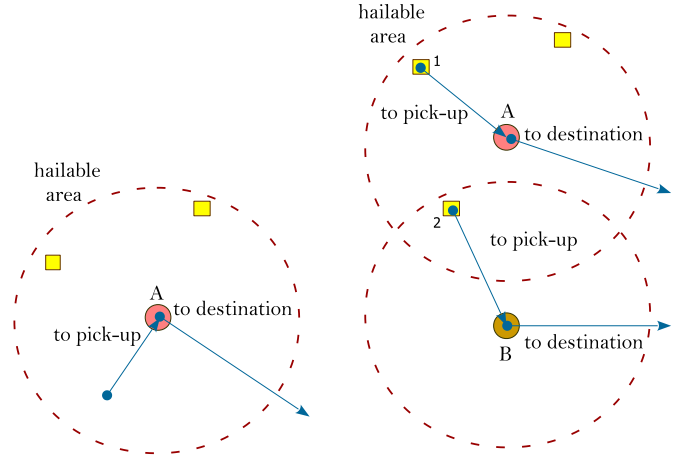


Fig. 1: Measuring the possibility of dynamic ride-sharing.

A. This study employs the strategy of allocating taxis in the closest region. There is less chance of a car being available in the pickup area because there aren't as many cars as there would be with regular taxi services. Therefore, the idea of hail regions will not be a feasible means to optimize the ride's path. To allocate a taxi that is closest to the pickup location, a method has been created.

3 Problem Description and Model Development

Examine the straightforward illustration in Fig. 2, where d denotes a driver and r a rider. The path is displayed as the bold solid line when driver d_1 joins a ride with riders r_1 and r_2 , respectively. This results in total distance savings of 4 units. However, the path is displayed as the dotted line when the driver shares a trip with both riders r_1 and r_2 simultaneously, and the three participants save a total of -4 units of distance. It is evident from the aforementioned example that driver d_1 is prepared to ride with each of the two passengers separately but not with both at the same time [68].

3.1 Problem Description

The problem of vehicle routing with time windows extends to the problem of ride-sharing ride-hailing optimization. The optimization takes into account the following criteria [20].

(1) Combinatorial optimization problem

Within a specific time frame, known as the system refresh time, the road network has r pairs of travel

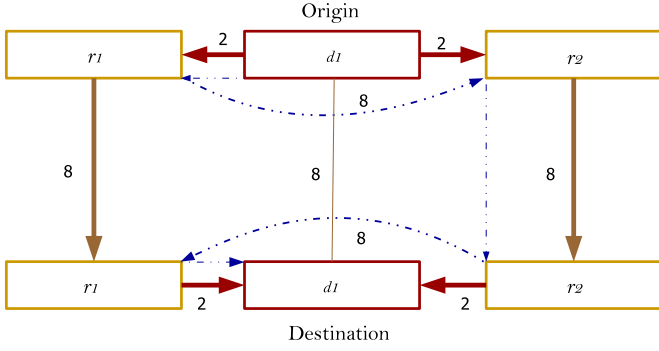


Fig. 2: An example of preference traits in ride-sharing.

demand. To minimize the overall operating mileage of the ride-hailing service system, the optimization looks for the optimal pairing of passengers with a vehicle.

(2) **Taxi assignment problem**

During the aforementioned time frame, the road network has k taxis. The algorithm determines the best ride-sharing plan for the system by allocating taxis to the aforementioned passengers.

(3) **Shortest path problem**

Each taxi ride-sharing demand's node order is provided once the ride-sharing route's scheme has been decided. Next, it is necessary to determine the shortest path between neighboring nodes. Since this is a classic problem that can be addressed by a classic algorithm, this work does not address it.

A mathematical explanation of the dynamic vehicle routing described in this work is as follows: There are r pairs of demand for travel $(1, 2, \dots, r)$ in the road network. Every one of them is associated with a distinct set of beginning and finishing locations. For ride-sharing, there are additionally k taxis $(1, 2, \dots, k)$. Q is the taxi's rated passenger capacity. The number of passengers in taxi s at position i after boarding and disembarking is q_i^s in the road network. $[T_{j1}, T_{j2}]$ is the time frame that passengers have provided. When $q_i^s \leq Q$ and $ArrT_j^s \leq T_{j2}$ are met, the best ride-sharing plan with the shortest system operating mileage is found to reduce the system's overall mileage.

3.2 Cost Function

This study uses the shortest ride-sharing route of the ride-sharing system to build the objective function identified in Eq. 1.

$$\min Z = \sum_{i \in N} \sum_{j \in N} \sum_{s \in K} x_{ij}^s \cdot d_{ij} \quad (1)$$

where N is the set of all nodes, K is the set of all taxi numbers, i and j are the node indexes, s is the taxi number index, d_{ij} is the actual distance of the shortest path between node i and node j , and $x_{ij}^s = 1$ if s taxi crosses the arc (i, j) , $x_{ij}^s = 0$; if not.

3.3 Taxi Route Constraints

The fundamental principles for ride-sharing ride hailing on the road network are represented by the vehicle path constraints in Eqs. 2, 3, 4, and 5 [69].

$$\sum_{i \in N(i \neq j)} \sum_{s \in K} x_{ij}^s = 1 \quad \forall j \in N \quad (2)$$

$$\sum_{j \in N(i \neq j)} \sum_{s \in K} x_{ij}^s = 1 \quad \forall i \in N \quad (3)$$

$$\sum_{p \in N} - \sum_{m \in N} x_{jm}^s = 0 \quad \forall j \in N, \forall s \in k \quad (4)$$

where the indexes of all nodes are p and m .

$$y_{ab}^r = 1 \Rightarrow \sum_{i \in N} x_{ib}^s - \sum_{j \in N} x_{aj}^s = 0 \quad (5)$$

$$\forall a \in O, \forall b \in D, \forall s \in K, \forall r \in R$$

where $y_{ab}^r = 1$ if group r arrives at location a and departs at position b , $y_{ab}^r = 0$; otherwise, the set of all demand beginning points and ending points is O , the set of all demand ending points is D , the set of passenger demand pairs is R , the index of demand starting number is a , and the index of demand ending number is b , and r stands for passenger demand pairs index.

Equations 2 and 3 guarantee that one car is present at each beginning and finishing location. It is guaranteed by Equation 4 that the vehicle coming to any node is the same vehicle leaving the node. To make sure it works, the distance between each passenger demand location and the taxi departure point is considered to be zero. This means that the forward distance still equals the real distance, and the reverse distance is zero. This makes it possible to think of the taxi as going back to the starting place of departure when it arrives at the final demand-ending location. The same cab must service any pair of demands, according to Equation 5. The nodes will have distinct numbers if there are multiple sets of demands with the same beginning or ending point.

3.4 Constraints on Rated Capacity

As can be shown from Eqs. 6 and 7, the number of passengers in the car at any one moment throughout the taxi ride-sharing service process should be fewer than or equal to its rated passengers.

$$z_{ij}^r = 1 \Rightarrow \sum_{s \in K} x_{ij}^s = 1 \quad \forall i, j \in N, \forall r \in R \quad (6)$$

where if the passenger of group r travels via the arc (i, j) , $z_{ij}^r = 1$; if not, $z_{ij}^r = 0$.

$$\sum_{j \in N} \sum_{r \in R} z_{ij}^r \cdot p^r \leq Q \quad \forall i \in N \quad (7)$$

where Q is the taxi's rated capacity and p^r is the number of passengers in group r .

The link between decision variables is represented by Equation 6, which states that there can only be one taxi traveling on any arc (i, j) with passengers. To make certain that the number of passengers in the taxi does not exceed the maximum loading capacity Q , Equation 7 limits the capacity of each arc.

3.5 Constrains of the Time Window

Passengers always anticipate that the taxi will come within the predetermined time range when they accept a taxi ride-sharing service. As a result, passengers must provide the time window condition, which is the predicted arrival time interval $[T_{j1}, T_{j2}]$. As seen in Figure 3, the study uses a soft time window restriction, which differs from the hard time window constraints of other research, and T_{j0} is considered as the key point of the passenger's tolerance decline.

$$ArrT_j^s = x_{ij}^s (DepT_i^s + t_{ij}^s) \quad \forall i, j \in N, \forall s \in K \quad (8)$$

where t_{ij}^s denotes the time from i to j , $ArrT_j^s$ denotes the real arrival time of taxi s at point j , and $DepT_i^s$ denotes the actual departure time of taxi s at point i .

$$ArrT_j^s - ArrT_i^s - B \cdot x_{ij}^s \geq t_{ij}^s - B \quad (9)$$

$$\forall i \in N, \forall j \in N, i \neq j, \forall s \in K$$

where B is a maximum value.

Equation 8 may be used to express the time it takes for the taxi s to travel from point i to point j . Subring elimination is performed using Equation 9, which guarantees the arrival order of two neighboring points.

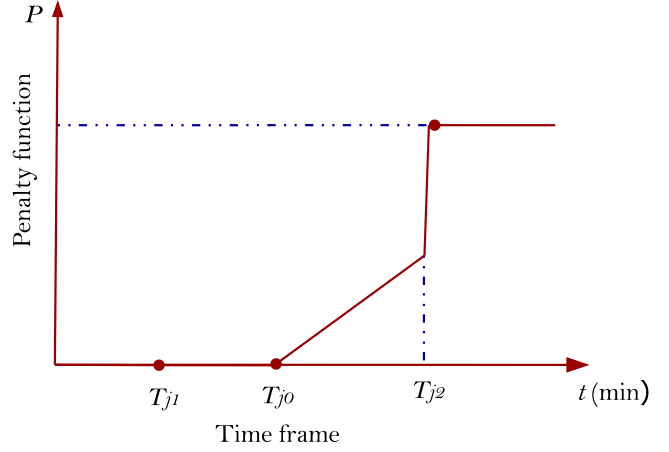


Fig. 3: Time window diagram.

$$t_{ij}^s = \frac{d_{ij}}{v_s} \quad \forall i, j \in N \quad (10)$$

where v_s is the taxi's average speed.

Equation 10 allows one to calculate the time it takes for a taxi to pass through points i and j .

$$ArrT_i^s \leq T_{i2} \quad \forall i \in N, \forall s \in K \quad (11)$$

where T_{i2} is the taxi's latest arrival time.

$$ArrT_i^s \leq T_{i1} \Rightarrow DepT_i^s = T_{i1} \quad (12)$$

$$\forall i \in N, \forall s \in K$$

where $ArrT_i^s$ is the actual arrival time of taxi s at point i , and T_{i1} is the taxi's earliest arrival time.

$$T_{i1} \leq ArrT_i^s \leq T_{i2} \Rightarrow DepT_i^s = ArrT_i^s \quad (13)$$

$$\forall i \in N, \forall s \in K$$

The time constraint specifies the rules for how cars should behave when they arrive at the service location at different times. It also represents the time window limitation in the car service procedure. The particular constraints are shown as Equations 11 through 13. The taxi's actual arrival time should be shorter than the passenger's desired last arrival time T_{j2} , according to equation 11. The instant the taxi s arrives at position i is less than the earliest arrival time the passenger provided, according to equation 12. Before departing from location i , the taxis must remain there until the passenger comes at the earliest possible moment. According to Equation 13, the taxi's arrival time is regarded as the same as the departure time when the s arrives at point

i between the passenger's earliest and the taxi's latest arrival times.

$$P_s^a = \begin{cases} 0 & ArrT_a^s < T_{a0} \\ r_1(ArrT_a^s - T_{a0}) & T_{a0} < ArrT_a^s < T_{a2} \end{cases} \quad (14)$$

where r_1 is the driver late penalty factor, T_{a0} indicates that the driver arrives at point a at the earliest without penalty, and P_s^a is the penalty function of taxi s timeout at location a .

When the taxi arrives later than the optimum arrival time T_{a0} , the passenger must wait for the taxi to get there, and their tolerance starts to decline. The driver should absorb the absence of passengers in this situation. Consequently, Equation 14 may be used to compute its penalty function P_s^a . The penalty function P_s^a , or the higher operating cost, will be taken into consideration for the real operation time $T_{a0} < ArrT_a^s < T_{a2}$. The fundamental presumptions of the ride-sharing route optimization model for ride-hailing are as follows: The time frame conditions for booking passengers are set and well-known, and all taxis operate on the road network at a consistent speed; It is possible to forecast when taxis will arrive at their destinations. The demand pair determines the passenger node positions. The right-of-way between two nodes is zero if the nodes are in the same location; the maximum number of passengers needed for each pair equals the taxi's capacity.

4 Narwhal Optimizer (NO)

This section explains the NO, including its mathematical model and source of inspiration.

4.1 Biological Fundamentals

The medium-sized toothed whale, or narwhal (*Monodon Monoceros*), is a remarkable marine animal that is distinguished by its long, spiral tusk. Compared to other whales, narwhals are middle in size, measuring between 4 and 6 meters without their tusks. As shown in Figure 4, they are distinguished by a speckled gray or brownish speckled skin that aids in their ability to blend in with the Arctic ice. They inhabit the Arctic waters around Greenland in Canada, Norway, and Russia all year round. They can move across sea ice to some extent.

4.1.1 Social Structure

Groups of narwhals are referred to as "Pods", which can contain anything from a dozen to a hundred individuals. Groups may include only of mothers and young, or



Fig. 4: Oceanic narwhals.

they may consist of both males and juveniles. It is also possible for a mixed group of juveniles, female, and male individuals to form at any time of year.

4.1.2 Migration

During the summer, the narwhals form pods of 10 to 100 individuals and approach the coast. They relocate to deeper waters covered by thick pack ice throughout the winter.

4.1.3 Diet

The narwhals shock fish with their long tusks before catching and devouring them. Greenland halibut, arctic, and Arctic cod make up their prey.

4.1.4 Communication and Coordination

Narwhals utilize sound to travel and seek for food. Their vocalization, which consists of whistles, knocks, and clicks, is a crucial component of pod communication. The clicks are employed for short-range obstacle location and prey detection. To plan the group actions, which include foraging for prey, contact with each narwhal is crucial. To navigate and locate prey, narwhals employ a variety of vocalizations, including clicks and whistles, which are employed in echolocation. Additionally, prey and other things in the water are identified by their clicks, which produce a brief plus sound. Narwhals hunt and move in pods, and they may communicate with one another to transmit essential information, such as the location of prey. By keeping the narwhals together while hunting and catching prey, these vocalizations aid in movement coordination and positional updating. Narwhals may become more vocal in their pod when they find prey. Vocalizations might be used to coordinate an assault, communicate the position of possible enemies, or indicate the existence of prey. We suggest a novel meta-heuristic algorithm called NO, which

draws inspiration from the ways narwhals communicate and seek their prey.

4.2 Mathematical Model

In NO, a narwhal's location within the prey's search area is a solution, which may be a way to solve optimization problems. Narwhals use signals to communicate with one another when they spot possible prey, and there may be an increase in communication within the pod. They will communicate the prey's position and utilize vocalization to indicate its presence. They also work together to be ready for the assault. The technique they use to find their prey is called echolocation. Narwhals use the echolocation technique to locate their prey by making clicks in the water and listening for them.

4.2.1 Initialization

First, a collection of random solutions representing narwhal locations (X) is used to launch the optimization method. These positions are continually changed in each algorithm iteration and are specified by the matrix below:

$$X = \begin{bmatrix} X_{1,1} & X_{1,2} & \dots & X_{1,d} \\ X_{2,1} & X_{2,2} & \dots & X_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n,1} & X_{n,2} & \dots & X_{n,d} \end{bmatrix} \quad (15)$$

where d is the dimension of the decision variables and n is the size of the narwhal population.

4.2.2 Signal Emission

When the narwhals move around, they communicate their positions and use signals to attempt to find the prey. We start by assuming that the signal's strength is extremely low, signifying the narwhals' exploratory stage in the search space. The position of the narwhal I and its perception of its surroundings determine the signal it emits. Emissions' function is explained as follows:

$$S_E(X_i) = \frac{0.1}{1 + \alpha \cdot \|X_i - X_{prey}\|} \quad (16)$$

where X_i represents the location of the i th narwhal and X_{prey} represents the location of the prey, which is also capable of detecting the signal and perhaps shifting its position (fish, for example, are narwhal prey that can

sense narwhal noises), the Euclidean distance between the location of the i th narwhal and the possible prey is $\|X_i - X_{prey}\|$, and the control factor α regulates the strength of the signal.

The signal emission at specific point X_i is represented by the function $S_E(X_i)$. The locations where the signal is measured in the water or on the surface are denoted by X_i . The position of the possible prey that the narwhal is pursuing is also X_{prey} . α may indicate parameters such as the narwhal's vocalization activity or the features of sound transmission in water, and it may regulate the way the emission strength varies with distance from the prey. The baseline emission rate, which might be the defining number for narwhal vocalizations, is determined by a scaling factor of 0.1. The emission function is normalized to an acceptable range or magnitude that is appropriate for the model using the value 0.1. By ensuring that the emission values are neither too high nor low, it facilitates their interpretation and manipulation. When the narwhal is nearer the prey, this function generates a greater signal; as the distance grows, the signal gets weaker.

4.2.3 Signal Propagation

The signal will travel across the water and may be represented as a function of the narwhals' separation from one another. Here is the definition of the propagation function:

$$S_p(X_i) = S_E(X_i) \times P_R(X_i, X_{prey}) \quad (17)$$

where the propagation function utilized to spread the signal is $P_R(X_i, X_{prey})$. The definition of this function is as follows:

$$P_R(X_i, X_{prey}) = \exp\left(-\frac{\|X_i - X_{prey}\|}{2 \times (\sigma^t)^2}\right) \quad (18)$$

where σ^t is the Gaussian's standard deviation at iteration t , which regulates how influence decreases with distance.

A small value of σ^t indicates more local communication, whereas a big value of σ^t indicates a more global communication that can adjust to a vast distance. Throughout the iterations, σ^t drops linearly. σ_0 is the initial value.

$$\sigma^t = \sigma_0 - \left(\frac{t}{T}\right) \quad (19)$$

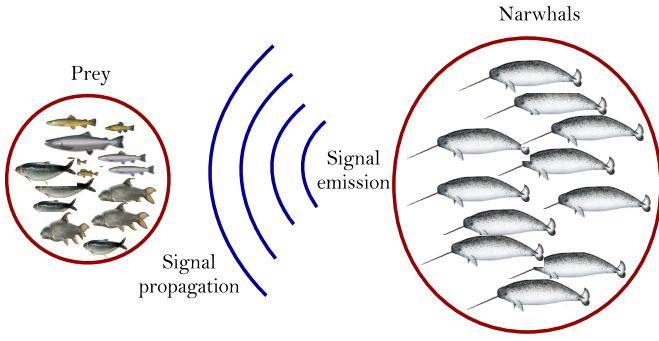


Fig. 5: Signal transmission and emission to find the prey.

4.2.4 Position Update

The narwhals' position is continually updated during each cycle. Following the signal's emission and propagation, it is updated. This may be modeled iteratively over time by using the following mathematical formula to update locations at each step:

$$X_i^{t+1} = X_i^t + \Delta^t \quad (20)$$

where the step at the t iteration is Δ^t , which may be identified through the following mathematical formula:

$$\Delta^t = \beta \times |S_p(i) \times X_{prey} - X_i| \quad (21)$$

$$\beta = r_1 - \frac{1}{\sigma^t + 1} \quad (22)$$

where β is an attribute associated with σ^t that governs the propagation's decline.

As was already explained, the narwhals' signal is detectable by prey. To put it another way, it may use $S_p(i) \times X_{prey}$ since the signal that is released may have an impact on the prey. Additionally, Figure 5 describes this method.

4.3 Algorithm of Narwhal Optimizer

In NO, the exploration or exploitation phase of the algorithm is determined by the σ^t parameter. The method begins with an initial value of σ^t , which lowers as iterations progress. The narwhals' signal's range of influence can be linked to the signal propagation parameter σ^t . A greater value of σ^t will expand the range of impact, which would enable a more thorough investigation of the surroundings. A lower value of σ^t , on the other

hand, favors a greater local exploitation of the information by limiting the signal's area of effect. The parameter β is another crucial one in the procedure. Slower position modifications caused by a modest amount of β encourage search space exploration. On the other hand, a high value of β can cause the position to change in response to the received signal more quickly and noticeably. This can lead to increased use of search space. Algorithm 1 describes the NO algorithm's pseudo code.

Algorithm 1 models the narwhals' prey-hunting behavior as an optimization method. There is a random solution at the beginning of the process. It computes the signal emission and propagation at each iteration. Using the value of signal propagation, the step Δ value is determined. We presume that the prey can sense the signal and will attempt to move to escape it. In every iteration, the algorithm modifies the narwhal's position in relation to the prey's position. Depending on the context of the problem being solved, the goal function is calculated. It stands for the number that the algorithm is trying to decrease or maximize. It is presumed that X is the vector whose dimension symbolizes the problem dimension, and that $f(X)$ is the objective function. To assess the algorithm's processing time and performance, complexity is a highly potent metric. The optimization issue has dimensions of d and there are n search agents (narwhals). The suggested algorithm's computational complexity is $O(n \times d)$.

5 Ant Colony Optimization

Dorigo et al. [70] introduced the Ant System (AS), a unique heuristic approach for tackling optimization problems. The traveling salesman problem (TSP) was the initial use of AS. Lots of problems were then addressed. The AS was then changed to the ACO, which increases its problem-solving capabilities. Since its introduction, the optimization community has been very interested in ACO and has applied it extensively to a variety of combinatorial optimization problems [70].

5.1 Inspiration of ACO

The ACO is an innovative method of computational optimization that takes its cues from ants' natural foraging habits. The ACO effectively gathers food by optimizing their pathways, much like the ant empire. Ants leave pheromone trails behind when they leave their nest to find food. The more ants that successfully navigate the route to the food source and return to the colony, the more intense this pheromone trail becomes.

Algorithm 1 A pseudo code that summarizes the Narwhal Optimizer’s iterative stages.

```

1: Randomly initialize the narwhals’ positions  $X_i$ 
2: Figure out the objective function’s value  $f(X)$ .
3: Set the initial values for  $\alpha$ ,  $\beta$ , and  $\sigma_0$ .
4: while ( $t \leq T$ ) do
5:   Utilize Equation 19 to update the value of  $\sigma^t$ .
6:   for each search agent narwhal  $i$  do
7:     Utilize Equation 16 to determine the signal emission  $S_E(X_i)$ .
8:     Determine the propagation of the signal  $S_P(X_i)$  using Equation 17.
9:     Update  $\beta$ ’s value using Equation 22.
10:    Utilize Equation 21 to update the value of  $\Delta^t$ .
11:    Update the narwhals’ location  $X_i^{t+1}$  utilizing Equation 20
12:   end for
13:   Figure out the objective function’s new value
14:    $t = t + 1$ 
15: end while
16: Provide the best solution got so far.

```

This organic method of problem-solving and optimization results in the creation of the quickest and most effective path between the nest and the food [70]. This intuitive approach is taken up by the ACO, which turns it into a potent computation tool. This method investigates possible solutions to an optimization problem using a group of artificial “ants”. Similar to their natural counterparts, these ants pursue and reinforce strong “pheromone trails”-which, in the context of the algorithm, indicate possible solution paths-in order to communicate and hone their search for the most effective solution. The ACO’s capacity for adaptation and optimal solution-finding in complex and dynamic contexts is one of its main advantages. This flexibility closely resembles how actual ants modify their courses in response to environmental changes, as when an obstruction obstructs their journey.

Because of ACO’s adaptive capability, it excels in addressing a wide range of optimization problems, from scheduling and network design to routing and logistics. Additionally, the decentralized decision-making process of ant colonies is like the decentralized nature of the ACO, where each artificial ant functions autonomously while contributing to a collective solution. This feature gives scalability, which is crucial for solving large-scale optimization issues, in addition to improving the algorithm’s resilience. The primary foraging process terms employed in the ACO’s design are depicted in Figure 6. The figure’s description is as follows:

- Beginning with Fig. 6(a), imagine that two ants (red and blue) are attempting to reach a food source via two different routes (A and B).
- As seen in Fig. 6(b), the probability of both ants traveling by path (A) or path (B) is precisely the same ($P(\text{Path}(A)) = P(\text{Path}(B))$).
- Assume that every ant follows a path, with the red ant following path (B) and the blue ant following

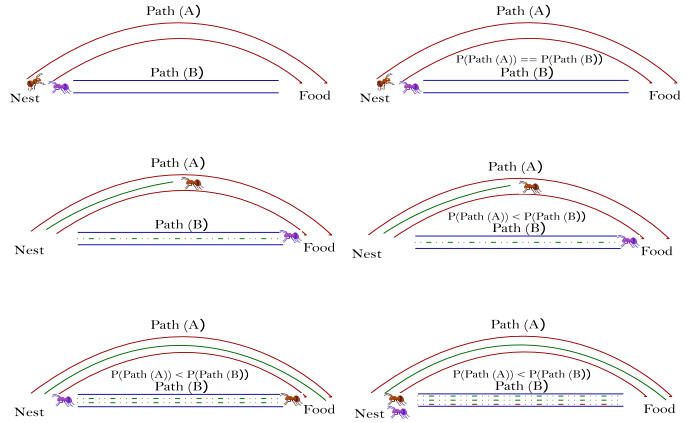


Fig. 6: Ant pheromone and likelihood of path.

path (A). The red ant arrives at the meal sooner since path (B) is shorter than path (A). Additionally, as seen in Fig. 6(c), when the ants move down the pathways, they generate a chemical called a pheromone (the dashed and dotted lines behind the ants) that facilitates ant communication.

- The situation shown in Fig. 6(d) is next examined. At this point, the red ant desires to return to its nest. A and B are its two pathways. But according to the pheromone of the pathways, the likelihood of following path B is greater than that of following path A. This is because the red ant cannot see the pheromone of path (A) because the blue ant has not yet arrived at the food source. Considering this, the red ant will follow route (B).
- The red ant arrived at the nest and generated more pheromones for route (B) in Fig. 6(e). Furthermore, the blue ant has just reached the position of the food supply. The blue ant then compares the quantity of pheromones in the two pathways when it wants to

return to the nest, and the probability of the paths are calculated accordingly.

- Lastly, the blue ant traveled through path (B), creating its pheromone there since path (B) has a greater probability than path (A).

According to the situation described, a new ant would most likely follow path (B) immediately if it wishes to get to the food source. This process uses probability and chemical signals called pheromones to determine the shortest path from the nest to the food supply. The pheromone will, however, eventually disappear, so bear that in mind.

5.2 Procedural Steps of ACO

In this section, the mathematical modeling for the earlier terminology is illustrated. The following sections go over each of the six stages that make up the ACO. The difficulty of the traveling salesman was taken into consideration when developing the discussion.

Step 1- Parameters Initialization: This step involves setting up the necessary parameters to execute the ACO algorithm, such as: (1) the number of ants in the colony (N), (2) the maximum number of iterations ($MaxItr$), (3) the evaporation rate (ρ), and (4) the pheromone and the desirability impact parameters (α , & β , respectively).

Step 2- Pheromone matrix construction: A matrix is created to hold the quantity of pheromone for every edge to imitate the pheromone. The pheromone matrix of dimension $v \times v$ is the τ matrix of Eq. 23, where the total number of vertices is denoted by v . For instance, the quantity of pheromone for an edge that connects the first and second vertices is denoted as $\tau_{1,2}$.

$$\tau = \begin{bmatrix} \tau_{1,1} & \tau_{1,2} & \dots & \tau_{1,d} \\ \tau_{2,1} & \tau_{2,2} & \dots & \tau_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ \tau_{n,1} & \tau_{n,2} & \dots & \tau_{n,d} \end{bmatrix} \quad (23)$$

Eq. 24 is used to compute the matrix values, where τ represents the rate of evaporation, $i \& j$ are the vertices of the edges, where $i = 1, 2, \dots, v$ & $j = 1, 2, \dots, v$, and $\Delta\tau_{i,j}^k$ represents the quantity of pheromone deposited on the edge i, j by the k th ant. The pheromones deposited on the edge (i, j) by every ant from $k = 1$ up to N , where N is the entire number of ants in the colony, are also added up to determine the overall quantity.

$$\tau_{i,j} = (1 - \rho)\tau_{i,j} + \sum_{k=1}^N \Delta\tau_{i,j}^k \quad (24)$$

Eq. 25 illustrates the conditions for computing $\Delta\tau_{i,j}^k$, which either indicates the edge length (cost) that the k th ant travels on (L_k) or 0 if the ant did not travel on that edge.

$$\Delta\tau_{i,j}^k = \begin{cases} \frac{1}{L_k} & \text{if } k\text{th ant travels on edge } i,j \\ 0 & \text{Otherwise} \end{cases} \quad (25)$$

Step 3- Solutions Construction (create colony):

Depending on the problem, N solutions are built as routes, with probability influencing the decision to choose one edge over another. so that the following formula is used to determine the likelihood of selecting the edge i, j :

$$p_{i,j} = \frac{(\tau_{i,j}^\alpha)(\eta_{i,j}^\beta)}{\sum (\tau_{i,j}^\alpha)(\eta_{i,j}^\beta)} \quad (26)$$

where α denotes the pheromone's influence and $\tau_{i,j}^\alpha$ is the pheromone quantity on the i, j edge. Conversely, the attractiveness of the i, j edge is represented by $\eta_{i,j}^\beta$, and the impact of the desirability is represented by β . The distance between $i \& j$ vertices is used to determine the attractiveness of the i, j edge ($\eta_{i,j} = \frac{1}{d_{ij}}$).

Step 4 - Calculate the Fitness Value: A fitness function (sometimes referred to as an objective function) is used to determine the cost of each solution.

Step 5 - Define the best solution (the queen ant):

According to the cost value of each solution, the better the solution, since the problem at hand is a minimization optimization problem. As such, the queen ant solution is the one that has the lowest cost.

Step 6 - Check the termination condition: Iteratively, steps two through five are carried out until the termination condition is satisfied. Once it reaches the queen ant, the answer is retrieved as the problem solution. The termination condition might be a predetermined number of iterations.

Algorithm 2 describes the key iterative steps of the ACO algorithm.

6 Proposed Taxi Ride Sharing Method

The queue technique is used in the current process. Therefore, regardless of pick-up and drop-off locations, a car with the bare minimum of passengers is allocated to a new pickup when requested. Only a certain number of cars are driven daily. Because the car does not choose the best route to drop off passengers, this procedure takes longer. First come, first served is the basis behind it. This lengthens the journey of each passenger and the waiting times of future passengers. Figure

Algorithm 2 A pseudo code that summarizes the ant colony optimization algorithm's iterative stages.

```

1: Input: Problem - certain parameters
2: Output: Near optimal or optimal solution
3: Step 1: Initialization of parameters
4: Set initial values for the following parameters: pheromone influence ( $\alpha$ ), heuristic influence ( $\beta$ ), evaporation rate ( $\rho$ ),
   maximum number of iterations ( $T$ ), pheromone matrix ( $\tau$ ), and number of ants ( $N$ ).
5: while Termination requirement not fulfilled do
6:   Step 2: Building of a pheromone matrix
7:   Update  $\tau$  for pheromone update using Eq. 24
8:   Step 3: building of solutions (colony creation)
9:
10:  for each ant  $k = 1$  to  $N$  do
11:    Build a solution using heuristic and pheromone information.
12:  end for
13:  Step 4: Determine the fitness value
14:
15:  for each ant  $k = 1$  to  $N$  do
16:    Use the selected objective function to evaluate the solution.
17:  end for
18:  Step 5: Identify the optimal solution
19:  Determine which solution has the highest objective function value.
20:  Update the pheromone levels according to the solutions' quality.
21:  Step 6: Verify the ending state
22:  Stop if the maximum number of iterations is achieved or if another stopping requirement is satisfied.
23: end while
24: Provide the best solution discovered so far

```

7 displays the flowchart for the proposed dynamic ride-sharing taxi.

Various forms of ride sharing can be employed, depending on the degree of freedom in selecting the locations for the pick-up and drop-off of passengers sharing a taxi. From the most static to the most dynamic, these models may be changed. Static trip scheduling is made possible by the static model, which uses the identical source and destination pairings for every passenger known before the trip begins. Requests arrive in real time and on demand in the dynamic model, which might be applied to any pairing of the source and destination of the passenger. Some models could assume that all passengers with various drop-off locations would be picked up at the same spot, while other models might think that all passengers with the same drop-off locations would be picked up at separate locations. With advance notice, the final two can be statically arranged. The second approach, on the other hand, is more dynamic and enables a continuous, real-time scheduling of requests.

6.1 The Optimization Process

The procedure is carried out as shown in the flowchart to increase efficiency. Starting the procedure at time $t = 0$ and there are no passengers in the taxis. Computations start when someone requests a random pick-up. The pick-up location is noted and contrasted with the original location of the taxis. A taxi will be allocated

to the passenger who is at least a short distance from the pickup site, and it will depart to pick up and drop off the passenger at the designated spot. Meanwhile, the second taxi is allocated whenever another request is made. This only applies to the first ride requests; more requests will be made as the journey continues.

A taxi is allocated, and the request's pickup location is indicated. Due to its continuous travel, a new method is used to allocate a taxi to the passenger. The total travel time for both taxis is compared, as is the waiting time for every passenger. The assigned passenger is the one with the least service time. If the trip time is optimized, this approach allows a passenger to be picked up and dropped off before another passenger gets in the taxi. The NO algorithm is used to optimize this procedure, where this procedure is covered in detail. When the taxis are empty once this loop is finished, it indicates that there are no passengers inside and that there is no further procedure; the taxis then wait for any more requests. Upon generating a request, the loop proceeds to the first step of determining the distance and allocating the taxi with the shortest distance. Until every passenger in the taxi has been dropped off, this procedure continues. The loop of this process finishes when the allotted time is over, no further requests will be taken into consideration, and the taxis will be empty.

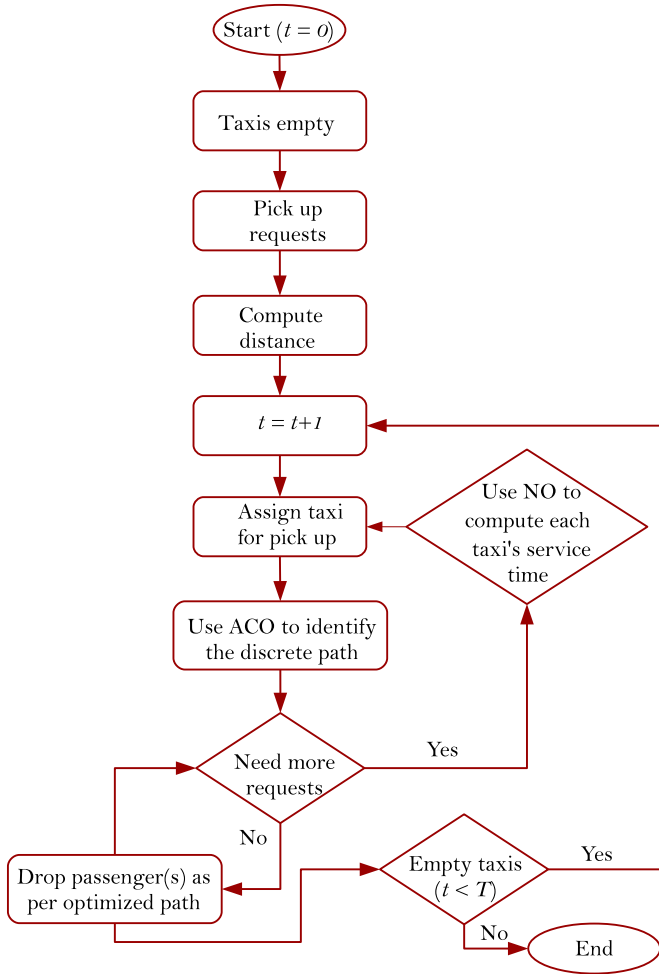


Fig. 7: A diagram showing the proposed dynamic ride-sharing method.

6.2 Taxi Assignment

Assume that there are n_1 passengers in taxi 1 and n_2 passengers in taxi 2 at any given time. Let the wait times for passengers in taxi 1 be $w_{11}, w_{12}, \dots, w_{1n_1}$. Likewise, $w_{21}, w_{22}, \dots, w_{2n_2}$ represent the passengers' wait times in taxi 2 at that time. The expected journey time for each passenger in taxi 1 is $t_{11}, t_{12}, \dots, t_{1n_1}$, which was retrieved from NO algorithm. Likewise, the anticipated journey time for passengers in taxi 2 is $t_{21}, t_{22}, \dots, t_{2n_2}$. The best route is determined by utilizing the Narwhal Optimizer (NO) using an approach analogous to the travel salesman problem, where the taxi position is kept as the first position and the passenger drop-off location is kept as other locations. Because it produces results quickly and with minimal effort, the NO algorithm is employed. It is most appropriate in our situation as we need a quick optimization tool to achieve the best results, and new requests are made at

random times. Because it is necessary for quicker convergent solutions, the overall population is assumed to be constant. According to statistics that the application reads, there are 500 passengers on average every run, where 500 passengers have tested the NO algorithm, and the results are encouraging. The taxi position is C , and the drop-off locations for passengers inside the position are d_1, d_2, \dots, d_n . Calculating the best route begins at C and proceeds via d_1, d_2, \dots, d_n . This may be expressed as follows:

$$C \rightarrow [d_1, d_2, \dots, d_n] \quad (27)$$

This determines each passenger's trip time (T) while keeping the average taxi speed constant. All current passengers in the taxi have an expected service time of S . Each passenger's service time is determined by adding his wait and travel times. Changes to waiting times are not possible because the passenger is currently inside the taxi W .

$$s_1 = w_{11} + t_{11} + w_{12} + t_{12} + \dots + w_{1n_1} + t_{1n_1} \quad (28)$$

$$s_2 = w_{21} + t_{21} + w_{22} + t_{22} + \dots + w_{2n_2} + t_{2n_2} \quad (29)$$

If a new passenger hires a taxi, taxis 1 and 2 will determine if they can accommodate the new passenger by looking at the higher S value. Only clients who are already in a taxi have their journey times altered by the new service time; waiting times remain the same.

$$n_{s_1} = w_{11} + n_{t_{11}} + w_{12} + n_{t_{12}} + \dots + w_{1n_1} + n_{T_{1n_1}} + w_{1p} + t_{1p} \quad (30)$$

$$n_{s_2} = w_{21} + n_{t_{21}} + w_{22} + n_{t_{22}} + \dots + w_{2n_2} + n_{T_{2n_2}} + w_{2p} + t_{2p} \quad (31)$$

In the event that the new passenger's service time in taxi 1 ($n_{s_1} - s_1$) is less than that of taxi 2 ($n_{s_2} - s_2$), taxi 1 will be allocated to them. Then s_1 will be replaced by n_{s_1} , but s_2 will stay unchanged. To calculate n_{s_1} , the NO method is utilized. However, the new passenger's pick-up point should be ahead of the drop-off location while calculating the best route. First, the best route is determined for all drop-off locations, including the new passenger drop-off location, which may be specified as follows, beginning from the taxi to $[d_1, d_2, d_3, \dots, d_n, d_p]$.

$$C \rightarrow [d_1, d_2, \dots, d_n, d_p] \quad (32)$$

Following that, a new pick-up location is added between each drop-off location, and the shortest overall distance is checked. Using the aforementioned strategy, the path with the shortest distance is the new optimal path.

7 Computational Results and Comparisons

The largest obstacle to testing a ride-sharing method is the lack of actual test data. Since ride sharing is always a spatiotemporal activity, our single-ride data is inappropriate for testing a ride sharing algorithm. Following extensive deliberation, we have chosen to use the random taxi dataset to test the proposed method. Consequently, the demand data for ride-sharing is simulated in computer studies using random taxi data. We investigate how the ride-sharing system is impacted by various allocation strategies, waiting time thresholds, participation rates, and the percentage of drivers among participants. We used spatial randomization of the passengers to address the coarse granularity of passenger and taxi locations and to more evenly distribute the requests. Additionally, the pick-up times and the first cab locations are both randomly selected. Programming was done using MATLAB 2021, and all tests were carried out on a Windows 10 computer with an Intel(R) Core(TM) i5-1135G7 @ 2.40GHz, 2.42 GHz, and 8.00GB of RAM.

7.1 Experimental Setup

The results of the proposed approach are contrasted with those of other meta-heuristic optimization techniques in order to demonstrate its overall effectiveness: Genetic Algorithms (GAs) [71], Particle Swarm Optimization (PSO) algorithm [72], Grey Wolf Optimizer (GWO) [73], Ant Colony Optimization (ACO) algorithm [74], Artificial Bee Colony (ABC) algorithm [51], and Grasshopper Optimization Algorithm (GOA) [75]. In Table 1, parameter settings for the suggested method and the other comparative algorithms are given. The aforementioned comparative meta-heuristics were chosen for this study due to their prior application in literature to the problem of interest, where they demonstrated encouraging results. Furthermore, these algorithms are comparable to the proposed method in many ways, such as simplicity, generality, and adaptability.

It was found that the parameter settings shown in Table 1 matched the ones often seen in the literature. For a fair comparison between the proposed algorithms

Table 1: The optimization methods' parameter settings.

Algorithm	Parameter	Value
All algorithms	Search agents	50
	Number of iterations	200
NO	α, σ_0	2
	The scaling factor	0.1
GWO	Control parameter (\vec{a})	[2, 0]
PSO	Inertia coefficient	0.75
	Cognitive coefficient	1.8
	Social coefficient	2
GA	Selection	Roulette wheel
	Crossover	0.9
	Mutation	0.05
ACO	β, ρ, α	2, 0.5, 0.3
	Number of colonies	5
ABC	Number of scott bees	30
	Number of follower bees	10
	Number of Onlooker bees	10
GOA	c_{max}	1
	c_{min}	0.00004

and those competing ones, the initialization procedure for the proposed algorithm is comparable to that of the others. The proposed approach employed six separate runs, and 50 search agents linked to 200 iterations.

7.2 Performance Evaluation

The assessment results of the proposed approach are shown in this part along with a comparison to other optimization techniques. The proposed framework is simulated using MATLAB, in which random requests were produced at random times with random pick-up and drop-off locations in the workspaces of X (0, 100) and Y (0, 100). To keep the simulation simple, taxis were initially assigned. Various numbers of passengers and taxis are used in simulations. Another set of simulations was conducted utilizing the currently employed queuing mechanism, taxis, and varying passenger counts.

Table 2 displays the duration of the six algorithm implementations. The GAACO and MOABC methods took the longest time to implement, while the proposed DNOACO approach required less time than the others. The computation complexity of the algorithms may be the cause of this discrepancy in algorithm implementation time. Compared to other algorithms, the MOPSO algorithm's computations are simpler. Furthermore, the proposed DNOACO algorithm found solutions with superior objective function values more quickly, which can ultimately aid in lowering the computational volume.

Table 2: The execution duration of each method in seconds across six implementations with comparable and consistent parameters.

Run number	DNOACO	GAACO	MOABC	MOPSO	MOGOA	MOGWO
1	13.27	34.51	55.34	8.21	26.72	19.11
2	14.32	33.05	60.60	9.58	27.63	20.93
3	14.48	33.42	61.27	10.13	27.82	21.16
4	14.35	33.12	60.72	9.68	27.67	20.97
5	14.75	34.05	62.43	11.08	28.16	21.56
6	13.27	35.64	56.18	8.96	26.34	19.40

In comparison to other algorithms, the proposed DNOACO algorithm's average implementation time is displayed in the Figure 8.

The found paths in Figure 8 need to be arranged such that they adequately cover the solution area. Furthermore, they shouldn't be overly focused on one location or deviating from the solution range. A few scenarios, including the current queue mode and the newly proposed DNOACO technique, are simulated. To compare the outcomes and determine the optimum passenger-to-taxi ratio, the number of passengers traveling has changed. Table 3 displays the simulation results of the proposed approach, in which each taxi travels at a constant pace, and Figure 9 shows the best tour length of the taxis.

According to the results shown in Table 3, the total distance driven by taxis falls as the number of passengers grows as compared to the queuing method. The creation of the optimum routes for the fleet's sustainability and high practicability has made the taxi routing challenge, as far as we know, one of the most difficult problems. In keeping with pertinent studies on vehicle routing, the ride-sharing ride-hailing problem likewise uses algorithms to organize suitable routes to satisfy passenger demands. Despite certain similarities, the ride-sharing ride-hailing problem is distinct from the vehicle routing problem. The taxis have set stations and starting places in the classic taxi routing problem, and they must go back to the origin after the service is finished. Different ride-hailing beginning places are used in this investigation. Waiting in the same station is not necessary because ride-hailing does not have a permanent station. The taxis may continue to operate after the service is complete without having to go back to where they started. Additionally, various starting and finishing places correlate to distinct passengers. As a result, passengers in taxis have trouble getting on and off in order.

The values of the target (objective) function pertaining to eight selected routes in nine distinct implementations of time and First Forward Path (FFP) num-

bers are displayed in Table 3. It is important to emphasize that minimizing objective function values is the aim of these evaluation experiments. The findings showed that the proposed DNOACO algorithm could, on average, identify superior values in the time objective function in terms of time implementations and FFP numbers. Eight non-dominated routes were chosen from among the nine implementations in Table 3. It is necessary for the find routes to adequately cover the solution space. Furthermore, they ought not to be overly focused on a single area or deviating from the range of solutions. The results were evaluated using the Error Ratio (ER) [76] and the Spacing Metric (SM) [77] indices, which were widely used in [24]. The optimum ER and SM values need to be low. SM is the diversity metric, and ER represents the convergence metric [24]. The values of ER and SM for all implementations in Table 3 are provided in Table 5. All the competing methods produce somewhat different results from ER and SM. Comparing the proposed DNOACO method to the competing algorithms, however, reveals that it has a superior convergence measure.

It is observed from Table 4 that the proposed DNOACO algorithm achieved the optimal time and the best first forward path numbers among all other competitors. This indicates the promising potential of the proposed algorithm as well as its efficiency in balancing exploration and exploitation abilities.

It is noted from Table 5 that the proposed DNOACO algorithm realized the optimal performance in terms of the ER and SM measures among all other competing algorithms. The difference in ER between the proposed DNOACO algorithm and the second-best algorithm is somewhat low, but the difference between DNOACO and the worst performing competing algorithm is relatively large. This demonstrates the promising potential of the proposed algorithm, as well as its efficiency in balancing exploration and exploitation capabilities.

Table 3: Results of the proposed algorithm's simulation, in which every taxi travels at a steady speed.

Total number of pickups	The total distance that taxis have gone	The total distance that taxis have gone (Queue)	%Decrease
10	1325	1870	7.14
20	2011	2192	9.56
30	2264	2773	21.42
40	2710	4182	37.22
50	2477	3965	40.91
60	3604	6051	42.69
70	4107	7251	45.63
80	4908	8274	51.97
90	5899	8454	61.83
100	6138	9155	72.87

Table 4: A comparison between the competing algorithms in terms of implementation time (in terms of seconds) and the number of solutions.

No	Ant/narwhal	Iteration count	DNOACO Time	DNOACO FFP numbers	GAACO Time	GAACO FFP numbers	MOABC Time	MOABC FFP numbers
1	15	50	5	11	23	7	33	9
2	15	100	6	8	26	6	39	6
3	15	200	8	10	39	5	34	6
4	35	50	7	15	26	6	40	7
5	35	100	9	15	31	7	46	9
6	35	200	12	16	42	8	55	9
7	70	50	10	18	35	9	48	13
8	70	100	13	19	48	10	56	15
9	70	200	16	19	58	12	70	16
Mean	-	-	9.5556	14.5556	36.4444	7.7778	46.7778	10.0000

No	Ant/narwhal	Iteration count	MOPSO Time	MOPSO FFP numbers	MOGOA Time	MOGOA FFP numbers	MOGWO Time	MOGWO FFP numbers
1	15	50	12	8	19	6	17	7
2	15	100	14	9	22	6	23	8
3	15	200	23	9	25	7	26	9
4	35	50	17	10	27	8	24	9
5	35	100	21	11	35	10	34	11
6	35	200	28	12	45	12	47	13
7	70	50	21	14	47	13	50	14
8	70	100	32	15	59	15	61	16
9	70	200	42	17	72	17	81	18
Mean	-	-	23.3333	11.6667	39.0000	10.4444	40.3333	11.6667

Table 5: A comparison between the competing algorithms in terms of ER and SM indices.

No.	DNOACO ER	DNOACO SM	GAACO ER	GAACO SM	MOABC ER	MOABC SM
1	0.5045	0.0710	0.5985	0.0794	0.5943	0.0846
2	0.5724	0.0867	0.6256	0.0710	0.6298	0.0668
3	0.5233	0.0668	0.6611	0.0689	0.7165	0.0700
4	0.5964	0.0637	0.7081	0.0742	0.7645	0.0783
5	0.5462	0.0762	0.7739	0.0762	0.8345	0.0710
6	0.5066	0.0564	0.8011	0.0700	0.8345	0.0689
7	0.5880	0.0752	0.7875	0.0825	0.7969	0.0825
8	0.6350	0.0637	0.7635	0.0700	0.7792	0.0668
9	0.6410	0.0642	0.7649	0.0731	0.7803	0.0670
Mean	0.5682	0.0693	0.7205	0.0739	0.7478	0.0729

No.	MOPSO ER	MOPSO SM	MOGOA ER	MOGOA SM	MOGWO ER	MOGWO SM
1	0.5559	0.0783	0.6595	0.0875	0.6549	0.0932
2	0.6307	0.0955	0.6894	0.0783	0.6940	0.0737
3	0.5766	0.0737	0.7285	0.0760	0.7895	0.0771
4	0.6572	0.0702	0.7803	0.0817	0.8425	0.0863
5	0.6019	0.0840	0.8528	0.0840	0.9196	0.0783
6	0.5582	0.0621	0.8827	0.0771	0.9196	0.0760
7	0.6480	0.0829	0.8678	0.0909	0.8781	0.0909
8	0.6997	0.0702	0.8413	0.0771	0.8586	0.0737
9	0.6999	0.0706	0.8417	0.0776	0.8589	0.0741
Mean	0.6253	0.0764	0.7938	0.0811	0.8240	0.0804

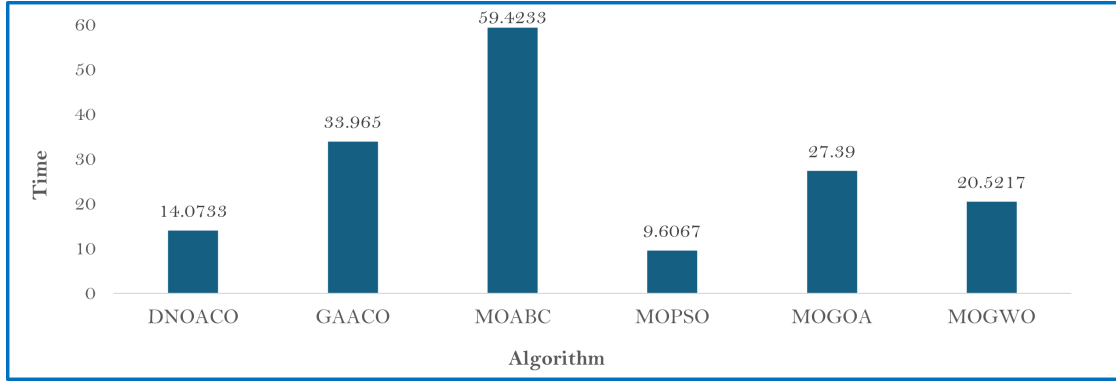


Fig. 8: An analysis of the presented algorithms' implementation timings was conducted by comparing their six implementations with identical parameters and averaging the duration (seconds) of each implementation.

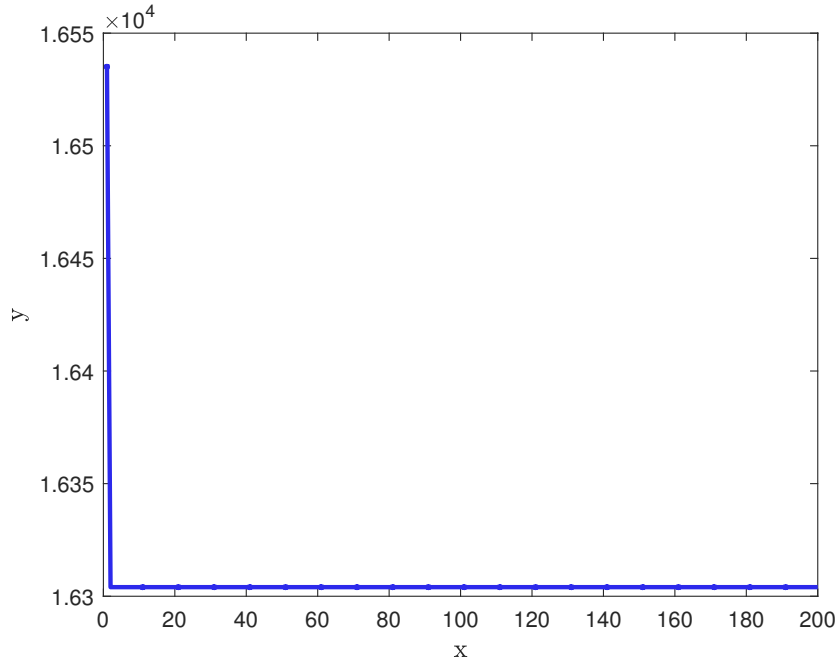


Fig. 9: Best tour length.

7.3 Statistical Test Analysis

Friedman's test, a non-parametric statistical test, was used to provide a thorough evaluation of the algorithms' performance. The goal of this investigation was to determine which optimization technique produced statistically better outcomes than the others. Table 6 and Figure 10 display the mean ranking results of the statistical comparison of the competing methods using Friedman's test based on implementation time (IT), error ratio, number of solutions in the first forward path numbers, and spacing measure.

As Table 6 and Figure 10 show, the result with the lowest ranking corresponds to a better performance

level. Friedman's test was used to determine the p -value, which is shown in Table 6. Several of the p -values were less than the significance level $\alpha = 0.5$. Significant findings from the statistical analysis supported the null hypothesis' rejection. This implies that there are statistically significant differences in the performance of rival optimization techniques. However, when applied to ride-sharing taxis, the null hypothesis assumed that there were no differences in the performance characteristics of the various systems. Based on the statistical findings shown in Table 6, the proposed DNOACO method is the best and most reliable method. This indicates that, in terms of performance outcomes, the proposed DNOACO approach outperformed GAACO,

Table 6: Average ranking results using Friedman’s test for all competing approaches.

Method	IT	FFP numbers	ER	SM
DNOACO	1.00	1.17	1.00	1.61
GAACO	3.89	5.72	3.00	2.89
MOABC	5.33	4.50	3.78	2.50
MOPSO	2.00	2.72	2.22	3.61
MOGOA	4.22	4.28	5.11	5.33
MOGWO	4.56	2.61	5.89	5.06
<i>p</i> -value	1.44E-06	1.97E-06	4.75E-08	4.35E-05

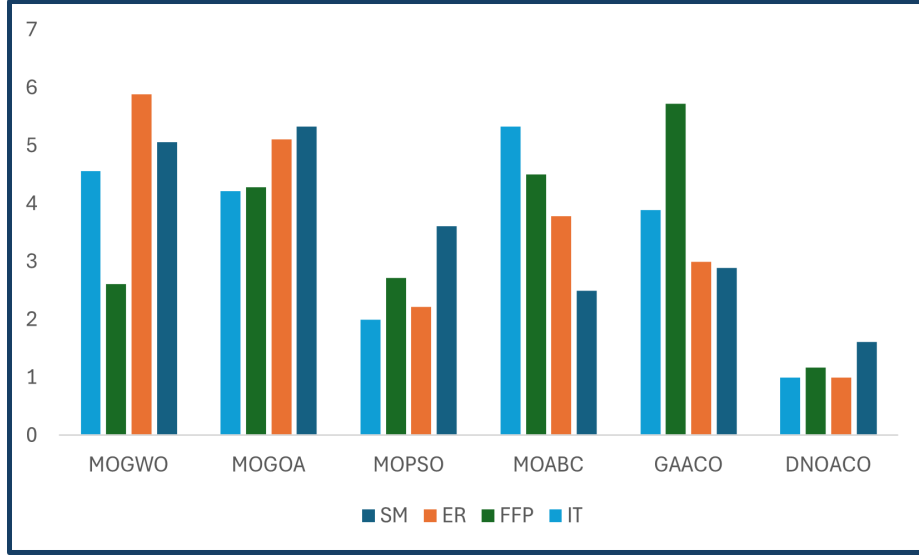


Fig. 10: Average ranking results using Friedman’s test.

MOABC, MOPSO, MOGOA, and MOGWO. These results demonstrate the DNOACO method’s greater scope in comparison to its competitors. Finally, the DNOACO technique received the greatest rank in terms of the overall duration of implementation time, with a rating of 1.0, the lowest of all the rankings that the other optimization methods displayed. A *post-hoc* statistical method was then employed to demonstrate the substantial difference between the control method and its rivals using Holm’s procedure. Accordingly, Friedman’s test findings demonstrate that the control optimization method outperforms the others in every evaluation criterion. Table 7 displays the findings of Holm’s statistical procedure. In this table, R_0 indicates the Friedman’s rank of the control optimization technique, R^i indicates the rank of the i th method, z indicates the statistical difference between two methods, and ES indicates the effect size of the control optimization method in the i th method.

Holm’s test allowed us to analyze the competing optimization techniques by rejecting hypotheses with p -values less than 0.05 for implementation time, less than 0.025 for FFP numbers, less than 0.05 for Error Ratio

(ER), and less than 0.01666 for Spacing Metric (SM), respectively. Table 7 demonstrates that, even in the absence of statistically significant differences between DNOACO and another optimization method (i.e., MOPSO), the proposed DNOACO approach outperforms GAACO, MOABC, MOGOA, and MOGWO in terms of implementation. The results of Friedman’s test and the Holm post-hoc analysis that followed showed that there were no statistically significant differences in FFP numbers between the DNOACO method and the MOPSO and MOGWO methods. However, the DNOACO method is notably different from the other optimization methods (GAACO, MOABC, and MOGOA). The ER findings indicate that DNOACO differs significantly from the rival method (AMOPSO), but not from the MOGWO, MOGOA, MOABC, and GAACO methods. DNOACO differs significantly from the MOGWO, MOGOA, and AMOPSO methods, but not from the other competing methods (GAACO and MOABC), according to the CM findings. The findings in Tables 6 and 7 demonstrate that the proposed DNOACO method significantly outperforms the other methods. The results of the statistical study above indicate that the DNOACO method

Table 7: Comparison of the optimization techniques using Holm’s test findings.

Implementation time (DNOACO is the control method)					
i	Algorithm	$z = \frac{(R_0 - R^i)}{SE}$	p-value	$\alpha \div i$	Hypothesis
5	MOABC	4.91353	8.944E-07	0.0100	Rejected
4	MOGWO	4.03162	5.539E-05	0.0125	Rejected
3	MOGOA	3.65365	2.585E-04	0.0166	Rejected
2	GAACO	3.27569	0.00105	0.0250	Rejected
1	MOPSO	1.13389	0.25683	0.0500	Not rejected

First forward path numbers (DNOACO is the control method)					
i	Algorithm	$z = \frac{(R_0 - R^i)}{SE}$	p-value	$\alpha \div i$	Hypothesis
5	GAACO	5.16551	2.397E-07	0.0100	Rejected
4	MOABC	3.77964	1.570E-04	0.0125	Rejected
3	MOGOA	3.52766	4.192E-04	0.0166	Rejected
2	MOPSO	1.76383	0.07775	0.0250	Not rejected
1	MOGWO	1.63784	0.10145	0.0500	Not rejected

Error ratio (DNOACO is the control method)					
i	Algorithm	$z = \frac{(R_0 - R^i)}{SE}$	p-value	$\alpha \div i$	Hypothesis
5	MOGWO	5.54347	2.965E-08	0.0100	Rejected
4	MOGOA	4.66156	3.138E-06	0.0125	Rejected
3	MOABC	3.14970	0.00163	0.0166	Rejected
2	GAACO	2.26778	0.02334	0.0250	Rejected
1	MOPSO	1.38586	0.16578	0.0500	Not rejected

Spacing metric (DNOACO is the control method)					
i	Algorithm	$z = \frac{(R_0 - R^i)}{SE}$	p-value	$\alpha \div i$	Hypothesis
5	MOGOA	4.22060	2.436E-05	0.0100	Rejected
4	MOGWO	3.90563	9.397E-05	0.0125	Rejected
3	MOPSO	2.26778	0.02334	0.0166	Rejected
2	GAACO	1.44886	0.14737	0.0250	Not rejected
1	MOABC	1.00790	0.31349	0.0500	Not rejected

outperformed several cutting-edge methods that have been documented in the literature, such as GAACO, MOABC, MOGOA, MOGWO, and MOPSO. This shows that the DNOACO approach performs reliably.

7.4 Challenges and Limitations

There are several challenges and limitations in the proposed work, presented as follows:

- Finding the NO algorithm’s fitness function, where the simulation’s guiding goal function may yield encouraging outcomes for distance minimization. Consequently, the proposed method was developed in MATLAB to reduce the overall path, which proved difficult.
- Determining the updated allocated pick-up location prior to the dropping point using the NO method. The pickup point must be ahead of the passenger’s drop-off location for the NO algorithm to find the best route; in a typical NO algorithm, the optimal path is solely for the drop-off locations. Therefore, figuring out how to formulate the argument for NO was a problem.
- Concurrently updating the taxis’ and passengers’ location values. A taxi is allocated to the pickup location when a new request is issued, and additional taxis get pickup requests while the ride is still in progress. It is necessary to compute and optimize the new pick-up values simultaneously for every taxi, not just one. One of the biggest challenges

was creating a loop that simultaneously changes the taxis' location information.

- This method has several disadvantages, such as lengthy wait times and lengthy service durations.

8 Conclusion and Future Work

The transportation system in metropolitan areas has grown overly complex in recent years, leaving residents perplexed about transportation and how to choose the best route, which makes it difficult to make decisions on route selection. In metropolitan areas, the transportation system is multimodal, and residents must select a path that combines many modes to get to work, the office, shopping, and other destinations. This study created a new model that works well for the current scenario of arbitrary taxi numbers, passengers, and routes. All passengers' optimal routes at any given time were obtained using the Discrete Narwhal Optimizer with Ant Colony Optimization (DNOACO) approach. According to this concept, each taxi determines how long it will take to deliver a passenger's request, and the taxi with the least amount of service time is allocated. This was accomplished by creating and promoting a meta-heuristic algorithm (DNOACO). The evaluation of the DNOACO algorithm involved a comparison with popular, standard, and widely used algorithms, including Genetic Algorithm-Ant Colony Optimization (GAACO), Multi-Objective Particle Swarm Optimization (MOPSO), Multi-Objective Artificial Bee Colony (MOABC), Multi-Objective Grasshopper Optimization Algorithm (MOGOA), Multi-Objective Gray Wolf Optimization (MOGWO), and Multi-Objective Artificial Bee Colony (MOABCO). A few non-dominated solutions allowed the user to select one path out of several best possibilities.

Simulation findings demonstrate that the created models provide outcomes with almost same efficiency for a reduced number of taxis. However, the created model outperforms the queue model in terms of efficiency as the number of passengers rises. The created model produced good results while considering the average number of passengers each independent run. The findings have significant theoretical and practical implications for the research and implementation of the combined route optimization problem, the enhancement of the intelligent level of urban taxi traffic, and the advancement and encouragement of sustainable urban traffic.

To sum up, people choose routes based on a variety of factors and goals, and they want the most efficient path from their starting point to their destination. The solution to the multi-modal problem will be a collection of routes, from which the user may select the

one that best suits their needs. Selecting a strong and quick method to tackle this problem can produce positive outcomes because each of these routes has unique and superior properties and is optimal.

The precision of the solutions found for multi-objective problems with discrete spaces is decreased by the continuous multi-objective narwhal optimizer method, which in some situations may not even find a solution at all. Discrete problem solutions are more accurate when the method is promoted to discrete.

Additional research may look at other ride-sharing system allocation techniques to encourage more individuals to sign up and maximize resource allocation, which would improve system effectiveness and user happiness. Also, this study does not deem the detrimental effect of ride-sharing on participants' utilities to keep the model simple. Further investigation into the impact of ride-sharing's detrimental impacts on system stability and the establishment of more realistic utility functions for participants would be interesting.

Acknowledgment

The authors gratefully acknowledge Zayed University for supporting this study under the Research Incentive Fund [RIF Grant No. 23202] with MT as the principal investigator.

Competing interests

The authors have no competing interests to declare that are relevant to the content of this article.

Author Contribution Declaration

M.T. and M.S. conceptualized the study and contributed to the design of the proposed algorithm. M.T. performed the literature review and prepared the initial draft of the manuscript. M.S. implemented the pre-processing techniques and conducted the experiments. M.B. carried out the hyperparameter optimization using the Capuchin Search Algorithm and prepared the performance evaluation metrics. M.A.A. supervised the overall research process, contributed to the critical revision of the manuscript, and provided technical and methodological guidance. All authors reviewed and approved the final manuscript.

Funding Support

We would like to thank Zayed University for supporting this work under the Research Incentive Fund [RIF Grant No. 23202].

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data availability Data available on request from the author.

References

1. Malik Braik and Heba Al-Hiary. A novel meta-heuristic optimization algorithm inspired by water uptake and transport in plants. *Neural Computing and Applications*, pages 1–82, 2025.
2. Malik Braik, Mohammed Awadallah, Mohammed Azmi Al-Betar, and Heba Al-Hiary. Enhanced whale optimization algorithm-based modeling and simulation analysis for industrial system parameter identification. *The Journal of Supercomputing*, 79(13):14489–14544, 2023.
3. Jesús Manuel Núñez-López, Juan Gabriel Segovia-Hernández, Eduardo Sánchez-Ramírez, and José María Ponce-Ortega. Integrating meta-heuristic methods and deterministic strategies for optimizing supply chain equipment design in process engineering. *Chemical Engineering Research and Design*, 214:93–104, 2025.
4. Muhammad Khahfi Zuhanda, Samsul A Rahman Sidik Hasibuan, Yose Yefta Napitupulu, et al. An exact and metaheuristic optimization framework for solving vehicle routing problems with shipment consolidation using population-based and swarm intelligence. *Decision Analytics Journal*, 13:100517, 2024.
5. Sahar Saeed Rezk and Kamal Samy Selim. Metaheuristic-based ensemble learning: an extensive review of methods and applications. *Neural Computing and Applications*, 36(29):17931–17959, 2024.
6. Malik Braik and Heba Al-Hiary. Rüppell’s fox optimizer: A novel meta-heuristic approach for solving global optimization problems. *Cluster Computing*, 28(5):1–77, 2025.
7. Malik Sh Braik, Mohammed A Awadallah, Osama Dorgham, Heba Al-Hiary, and Mohammed Azmi Al-Betar. Applications of dynamic feature selection based on augmented white shark optimizer for medical diagnosis. *Expert Systems with Applications*, 257:124973, 2024.
8. Malik Sh Braik. Modified chameleon swarm algorithm for brightness and contrast enhancement of satellite images. *Multimedia Tools and Applications*, 83(9):26819–26870, 2024.
9. Nima Khodadadi, Ehsan Khodadadi, Qasem Al-Tashi, El-Sayed M El-Kenawy, Laith Abualigah, Said Jadid Abdulkadir, Alawi Alqushaibi, and Seyedali Mirjalili. Baoa: binary arithmetic optimization algorithm with k-nearest neighbor classifier for feature selection. *IEEE Access*, 11:94094–94115, 2023.
10. Malik Braik, Heba Al-Hiary, Hussein Alzoubi, Abdelaziz Hammouri, Mohammed Azmi Al-Betar, and Mohammed A Awadallah. Tornado optimizer with coriolis force: a novel bio-inspired meta-heuristic algorithm for solving engineering problems. *Artificial Intelligence Review*, 58(4):1–99, 2025.
11. Ali Kaveh, Siamak Talatahari, and Nima Khodadadi. Stochastic paint optimizer: theory and application in civil engineering. *Engineering with Computers*, pages 1–32, 2022.
12. Malik Braik, Alaa Sheta, and Heba Al-Hiary. A novel meta-heuristic search algorithm for solving optimization problems: capuchin search algorithm. *Neural computing and applications*, 33(7):2515–2547, 2021.
13. Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2007.
14. John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.
15. Nima Khodadadi, El-Sayed M El-Kenawy, Francisco De_Caso, Amal H Alharbi, Doaa Sami Khafaga, and Antonio Nanni. The mountain gazelle optimizer for truss structures optimization. *Applied Computing and Intelligence*, 3(2), 2023.
16. Amit Kumar Das, Ankit Kumar Nikum, Siva Vignesh Krishnan, and Dilip Kumar Pratihari. Multi-objective bonobo optimizer (mobo): an intelligent heuristic for multi-criteria optimization. *Knowledge and Information Systems*, 62(11):4407–4444, 2020.
17. Nima Khodadadi, Siamak Talatahari, and Amir H Gandomi. Anna: advanced neural network algo-

- rithm for optimisation of structures. *Proceedings of the Institution of Civil Engineers-Structures and Buildings*, 177(6):529–551, 2023.
18. Djaafar Zouache, Yahya Ould Arby, Farid Nouioua, and Fouad Ben Abdelaziz. Multi-objective chicken swarm optimization: A novel algorithm for solving multi-objective optimization problems. *Computers & Industrial Engineering*, 129:377–391, 2019.
 19. Nima Khodadadi, Mahdi Azizi, Siamak Talatahari, and Pooya Sareh. Multi-objective crystal structure algorithm (mocrystal): Introduction and performance evaluation. *IEEE Access*, 9:117795–117812, 2021.
 20. Yi Cao, Shan Wang, and Jinyang Li. The optimization model of ride-sharing route for ride hailing considering both system optimization and user fairness. *Sustainability*, 13(2):902, 2021.
 21. Apurba Manna, Arindam Roy, Samir Maity, Sukumar Mondal, and Izabela Ewa Nielsen. A multi-parent genetic algorithm for solving longitude–latitude-based 4d traveling salesman problems under uncertainty. *Decision Analytics Journal*, 8:100287, 2023.
 22. Ufuk Dereci and Muhammed Erkan Karabekmez. The applications of multiple route optimization heuristics and meta-heuristic algorithms to solid waste transportation: A case study in turkey. *Decision Analytics Journal*, 4:100113, 2022.
 23. Omar Dib, Marie-Ange Manier, Laurent Moalic, and Alexandre Caminada. A multimodal transport network model and efficient algorithms for building advanced traveler information systems. *Transportation research procedia*, 22:134–143, 2017.
 24. Hamed Farooqi and Mohammad saadi Mesgari. Performance comparison between the multi-colony and multi-pheromone aco algorithms for solving the multi-objective routing problem in a public transportation network. *The Journal of Navigation*, 69(1):197–210, 2016.
 25. Shafiq Alam, Xin Zhao, Imran Khan Niazi, Muhammad Sohaib Ayub, and Muhammad Asad Khan. A comparative analysis of global optimization algorithms for surface electromyographic signal onset detection. *Decision Analytics Journal*, 8:100294, 2023.
 26. Gai-Ge Wang, Da Gao, and Witold Pedrycz. Solving multiobjective fuzzy job-shop scheduling problem by a hybrid adaptive differential evolution algorithm. *IEEE Transactions on Industrial Informatics*, 18(12):8519–8528, 2022.
 27. Wenxing Ye, Weiyang Feng, and Suohai Fan. A novel multi-swarm particle swarm optimization with dynamic learning strategy. *Applied Soft Computing*, 61:832–843, 2017.
 28. Mohammad Mahdi Nasiri, Hossein Mousavi, and Saeede Nosrati-Abarghoee. A green location-inventory-routing optimization model with simultaneous pickup and delivery under disruption risks. *Decision analytics journal*, 6:100161, 2023.
 29. GA Klunder and HN Post. The shortest path problem on large-scale real-road networks. *Networks: An International Journal*, 48(4):182–194, 2006.
 30. M Kheirikharzar. Shortest path algorithm in multimodal networks for optimization of public transport. In *FIG Congress 2010 Facing The Challenges-Building the Capacity*, 2010.
 31. Russell Tessier. Negotiated a* routing for fpgas. In *Proceedings of the 5th Canadian Workshop on Field Programmable Devices*, volume 6. Citeseer, 1998.
 32. Raka Jovanovic and Stefan Voß. A matheuristic approach for solving the 2-connected dominating set problem. *Applicable Analysis and Discrete Mathematics*, 14(3):775–799, 2020.
 33. Alisha Roushan, Amrit Das, Anirban Dutta, and Uttam Kumar Bera. A pentagonal type-2 fuzzy variable defuzzification model with application in humanitarian supply chains. *Decision Analytics Journal*, 8:100303, 2023.
 34. Ramin Talebi Khameneh, Kash Barker, and Jose Emmanuel Ramirez-Marquez. A hybrid machine learning and simulation framework for modeling and understanding disinformation-induced disruptions in public transit systems. *Reliability Engineering & System Safety*, 255:110656, 2025.
 35. Kavitha Sooda and TR Nair. A comparative analysis for determining the optimal path using pso and ga. *arXiv preprint arXiv:1407.5327*, 2014.
 36. James Kennedy and Russell C Eberhart. A discrete binary version of the particle swarm algorithm. In *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation*, volume 5, pages 4104–4108. iee, 1997.
 37. Mahboubah Farid, Mikael Palmblad, Hampus Hallman, and Johannes Vänngård. A binary decision tree approach for pharmaceutical project portfolio management. *Decision Analytics Journal*, 7:100228, 2023.
 38. Xiaoguang Lu, Yunhong Wang, and Anil K Jain. Combining classifiers for face recognition. In *2003 International Conference on Multimedia and Expo. ICME'03. Proceedings (Cat. No. 03TH8698)*, volume 3, pages III–13. IEEE, 2003.
 39. Panick Kalambay, Angela Kitale, Abdul Ngereza, Emmanuel Kidando, and Abimbola Ogungbire. Autonomous taxis and ride-sharing vehicles: A so-

- cial construct perspective for future mobility and infrastructure readiness. *Sustainable Cities and Society*, 118:106060, 2025.
40. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multi-objective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.
 41. Trung Vo-Duy, Dung Duong-Gia, Vinh Ho-Huu, Huy Cuong Vu-Do, and Trung Nguyen-Thoi. Multi-objective optimization of laminated composite beam structures using nsga-ii algorithm. *Composite Structures*, 168:498–509, 2017.
 42. Raka Jovanovic, Abdelkader Bousselham, and Stefan Voß. Partitioning of supply/demand graphs with capacity limitations: an ant colony approach. *Journal of Combinatorial Optimization*, 35:224–249, 2018.
 43. CA Coello Coello and Maximino Salazar Lechuga. Mopso: A proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC’02 (Cat. No. 02TH8600)*, volume 2, pages 1051–1056. IEEE, 2002.
 44. Zohreh Masoomi, Mohammad Sadi Mesgari, and Majid Hamrah. Allocation of urban land uses by multi-objective particle swarm optimization algorithm. *International Journal of Geographical Information Science*, 27(3):542–566, 2013.
 45. Harilaos N Psaraftis. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14(2):130–154, 1980.
 46. Ahmed Atahran, Christophe Lenté, and Vincent T’kindt. A multicriteria dial-a-ride problem with an ecological measure and heterogeneous vehicles. *Journal of Multi-Criteria Decision Analysis*, 21(5-6):279–298, 2014.
 47. Claudia Archetti, Martin Savelsbergh, and M Grazia Speranza. The vehicle routing problem with occasional drivers. *European Journal of Operational Research*, 254(2):472–480, 2016.
 48. Seyyid Ahmed Medjahed and B Fatima. Narwhal optimizer: A novel nature-inspired metaheuristic algorithm. *Int. Arab J. Inf. Technol.*, 21:418–426, 2024.
 49. Ramin Hedayatzadeh, Bahareh Hasanizadeh, Reza Akbari, and Koorush Ziarati. A multi-objective artificial bee colony for optimizing multi-objective problems. In *2010 3rd international conference on advanced computer theory and engineering (ICACTE)*, volume 5, pages V5–277. IEEE, 2010.
 50. E Emary, Waleed Yamany, Aboul Ella Hassanien, and Vaclav Snasel. Multi-objective gray-wolf optimization for attribute reduction. *Procedia Computer Science*, 65:623–632, 2015.
 51. Dervis Karaboga. Artificial bee colony algorithm. *scholarpedia*, 5(3):6915, 2010.
 52. Ye-qian Lin, Wen-quan Li, Feng Qiu, and He Xu. Research on optimization of vehicle routing problem for ride-sharing taxi. *Procedia-Social and Behavioral Sciences*, 43:494–502, 2012.
 53. Zhihai Xiang, Chengbin Chu, and Haoxun Chen. A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. *European journal of operational research*, 174(2):1117–1139, 2006.
 54. Roberto Baldacci, Vittorio Maniezzo, and Aristide Mingozzi. An exact method for the car pooling problem based on lagrangean column generation. *Operations research*, 52(3):422–439, 2004.
 55. Mustafa Lokhandwala and Hua Cai. Dynamic ride sharing using traditional taxis and shared autonomous taxis: A case study of nyc. *Transportation Research Part C: Emerging Technologies*, 97:45–60, 2018.
 56. Rui Chen and Christos G Cassandras. Optimization of ride sharing systems using event-driven receding horizon control. *IFAC-PapersOnLine*, 53(4):411–416, 2020.
 57. Mitja Stiglic, Niels Agatz, Martin Savelsbergh, and Mirko Gradisar. Enhancing urban mobility: Integrating ride-sharing and public transit. *Computers & Operations Research*, 90:12–21, 2018.
 58. Bin Cao, Louai Alarabi, Mohamed F Mokbel, and Anas Basalamah. Sharek: A scalable dynamic ride sharing system. In *2015 16th IEEE International Conference on Mobile Data Management*, volume 1, pages 4–13. IEEE, 2015.
 59. Yazhe Wang, Baihua Zheng, and Ee-Peng Lim. Understanding the effects of taxi ride-sharing—a case study of singapore. *Computers, Environment and Urban Systems*, 69:124–132, 2018.
 60. Pedro M d’Orey, Ricardo Fernandes, and Michel Ferreira. Empirical evaluation of a dynamic and distributed taxi-sharing system. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 140–146. IEEE, 2012.
 61. Kanika Bathla, Vaskar Raychoudhury, Divya Saxena, and Ajay D Kshemkalyani. Real-time distributed taxi ride sharing. In *2018 21st international conference on intelligent transportation systems (ITSC)*, pages 2044–2051. IEEE, 2018.
 62. Pradip Kumar Sharma, Seo Yeon Moon, and Jong Hyuk Park. Block-vn: A distributed

- blockchain based vehicular network architecture in smart city. *Journal of information processing systems*, 13(1):184–195, 2017.
63. Weicheng Zhao, Yajuan Qin, Dong Yang, Linjuan Zhang, and Wanting Zhu. Social group architecture based distributed ride-sharing service in vanet. *International Journal of Distributed Sensor Networks*, 10(3):650923, 2014.
 64. Dejan Dimitrijević, Vladimir Dimitrieski, and Nemanja Nedić. Prototype implementation of a scalable real-time dynamic carpooling and ride-sharing application. *Informatica*, 38(3), 2014.
 65. Shuo Ma, Yu Zheng, and Ouri Wolfson. Real-time city-scale taxi ridesharing. *IEEE Transactions on Knowledge and Data Engineering*, 27(7):1782–1795, 2014.
 66. Masayo Ota, Huy Vo, Claudio Silva, and Juliana Freire. A scalable approach for data-driven taxi ride-sharing simulation. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 888–897. IEEE, 2015.
 67. Markos Papageorgiou, Christina Diakaki, Vaya Dinopoulou, Apostolos Kotsialos, and Yibing Wang. Review of road traffic control strategies. *Proceedings of the IEEE*, 91(12):2043–2067, 2003.
 68. Hua Ke and Haoyang Li. Multi-rider ridesharing stable matching optimization. *Soft Computing*, 28(20):12005–12020, 2024.
 69. Rongge Guo, Wei Guan, Wenyi Zhang, Fanting Meng, and Zixian Zhang. Customized bus routing problem with time window restrictions: model and case study. *Transportmetrica A: Transport Science*, 15(2):1804–1824, 2019.
 70. Marco Dorigo and Luca Maria Gambardella. Ant colonies for the travelling salesman problem. *biosystems*, 43(2):73–81, 1997.
 71. Eric Bonabeau, Directeur de Recherches Du Fnrs Marco, Marco Dorigo, Guy Theraulaz, et al. *Swarm intelligence: from natural to artificial systems*. Number 1. Oxford university press, 1999.
 72. James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948, Perth, Australia, 1995. IEEE.
 73. Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Advances in engineering software*, 69:46–61, 2014.
 74. Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical computer science*, 344(2-3):243–278, 2005.
 75. Shahrzad Saremi, Seyedali Mirjalili, and Andrew Lewis. Grasshopper optimisation algorithm: theory and application. *Advances in engineering software*, 105:30–47, 2017.
 76. David Allen Van Veldhuizen. *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations*. Air Force Institute of Technology, 1999.
 77. Jason Ramon Schott. *Fault tolerant design using single and multicriteria genetic algorithm optimization*. PhD thesis, Massachusetts Institute of Technology, 1995.