

Article

# A Quick Pheromone Matrix Adaptation Ant Colony Optimization for Dynamic Customers in the Vehicle Routing Problem

Yuxin Liu \*, Zhitian Wang and Jin Liu

College of Information Engineering, Shanghai Maritime University, Shanghai 201306, China

\* Correspondence: liuyx@shmtu.edu.cn

**Abstract:** The path planning problem is an important issue in maritime search and rescue. This paper models the path planning problem as a dynamic vehicle routing problem. It first designs a dynamic generator that transforms the existing benchmark sets for the static vehicle routing problem into dynamic scenarios. Subsequently, it proposes an effective Dynamic Ant Colony Optimization (DACO) algorithm, whose novelty lies in that it dynamically adjusts the pheromone matrix to efficiently handle customers' changes. Moreover, DACO incorporates simulated annealing to increase population diversity and employs a local search operator that is dedicated to route modification for continuous performance maximization of the route. The experimental results demonstrated that the proposed DACO outperformed existing approaches in generating better routes across various benchmark sets. Specifically, DACO achieved significant improvements in the route cost, serviced customer quantity, and adherence to time window requirements. These results highlight the superiority of DACO in the dynamic vehicle routing problem, providing an effective solution for similar problems.

**Keywords:** maritime search and rescue; dynamic vehicle routing problem; ant colony optimization; pheromone matrix; simulated annealing



**Citation:** Liu, Y.; Wang, Z.; Liu, J. A Quick Pheromone Matrix Adaptation Ant Colony Optimization for Dynamic Customers in the Vehicle Routing Problem. *J. Mar. Sci. Eng.* **2024**, *12*, 1167. <https://doi.org/10.3390/jmse12071167>

Academic Editor: Mihalis Goliás

Received: 18 June 2024

Revised: 8 July 2024

Accepted: 8 July 2024

Published: 11 July 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The oceans cover over 70% of the Earth's surface, serving as not only a source of life and a treasure trove of resources, but also as a crucial domain for human activities, including marine fisheries, tourism, and resource exploitation. One of the reasons for the prosperity of maritime businesses is that, despite the increasing intensity of marine transportation, the number of maritime accidents has not sharply increased, thanks to improvements in safety measures and technology. This observation can be seen from a recent study showing that, with the overall increment in oil trading, oil spills have decreased [1]. Nonetheless, once maritime accidents happen, they could result in significant human casualties, property losses, and severe environmental impacts. Hence, in order to promote the long-term development of maritime businesses, it has become imperative to conduct research and implement maritime search and rescue missions [2]. To ensure the efficient execution of these missions, it is common to deploy a group of unmanned aerial vehicles (UAVs) to conduct surveillance in the target area and initiate rescue operations based on the real-time assessment of the disaster situation [3]. In maritime rescue operations, efficient path planning for UAVs becomes crucial. UAVs depart from the base station carrying a certain amount of rescue supplies, then they patrol multiple disaster sites, locate the targets in need of rescue, and drop the rescue equipment, such as lifebuoys. After completing the mission, these UAVs need to return to the base station. The path planning problem for UAVs in maritime search and rescue can be modeled as a vehicle routing problem (VRP) [4,5], aiming to find optimal paths that minimize search and rescue time while maximizing the effectiveness of the rescue efforts.

As a variant of the Traveling Salesman Problem (TSP) [6], the VRP [7] is an NP-hard combinatorial optimization problem that originated in the field of logistics. It was first

introduced by Dantzig and Ramser [8] in 1959 to address the challenge of route planning and resource utilization for delivery vehicles. The fundamental idea of the VRP is to determine the optimal routes for a set of delivery locations and a given number of vehicles, aiming to fulfill all customer demands while minimizing vehicle travel distances. In recent years, in order to model real-world scenarios more accurately, many variants of the VRP [9,10] have been proposed, in which the VRP with time windows (VRPTW) [11,12] is a typical one. In the VRPTW, each customer has a specific time window within which the delivery should be made. The objective is to find optimal routes for vehicles to satisfy customer demands while respecting the time window constraints.

In real-world applications, the dynamic nature of logistics operations is an unavoidable issue that poses significant challenges [13,14]. For example, new search and rescue targets can emerge at any time, while some other targets may be deemed unreachable due to either the impossibility of rescue or because they have been confirmed to be safe. The inherent unpredictability of a real-world VRP has resulted in the expansion of the static VRP to a category of problems known as the dynamic VRP (DVRP) [15,16]. In particular, this paper constructs and studies a dynamic vehicle routing problem with time windows (DVRPTW), since the rescue operation needs to be carried out within an urgent time frame [17].

Among the approaches to tackle the DVRPTW, meta-heuristics demonstrate efficient performance, as they draw inspiration from nature and can provide an approximate optimal solution within a reasonable time [9,18]. As one of the typical meta-heuristics, Ant Colony Optimization (ACO) [19] has been successfully applied to various VRPs. ACO mimics the behavior of ants searching for food in a distributed manner, which closely resembles the DVRP, and ACO can accommodate uncertainty in the DVRP by introducing stochastic elements; this enables ACO to address the DVRP, leading to more robust solutions [20]. However, ACO encounters a significant challenge when the environment changes, which has not been efficiently overcome, i.e., the pheromones from the previous environment tend to bias the algorithm towards the old optimal solution [21]. This poses a difficulty for the algorithm to adapt and find the new optimal solution. Once the algorithm converges, it may struggle to adjust to the changing environment. One potential solution is to re-initialize the pheromone matrix, but treating each dynamic change as a complete restart step is often inefficient.

Therefore, how to adjust the pheromone matrix to adapt to dynamic changes has become a crucial issue [20]. When a small portion of the dynamic environment changes, most of the information provided by the old pheromone matrix remains relevant to the new environment. In this case, only a small part of the pheromones needs to be adjusted. However, if the majority of the environment undergoes significant changes, then it becomes necessary to re-initialize the entire pheromone matrix.

Several strategies have been proposed and combined with ACO to reduce the re-optimization time while efficiently maintaining high-quality output. These strategies can be divided into four categories: increasing diversity after a dynamic change [21], maintaining diversity during the execution [22], memory-based schemes [23], and hybrid algorithms [24]. Among these strategies, solving the DVRP based on the immigration strategy [25] shows promising results. In the immigration strategy, some newly generated ants are called immigrant ants, which replace some ants in the current population and improve the performance of the overall algorithm. Based on the different ways of immigration generation, it can be divided into the random generation of immigrants (RIACO), the generation of immigrants based on elitism (EIACO), and the generation of immigrants based on memory (MIACO) [25].

RIACO generates  $n$  random immigrants to replace  $n$  ants that are the worst performing ants in the current environment to improve algorithm performance. EIACO selects the best performing ants as immigrants to replace the worst performing ants in the environment to improve the performance of the algorithm. MIACO is suitable for cyclic environments where the environmental changes are cyclic, storing several memories of migrants to replace the corresponding ants when the environment changes.

However, the algorithms mentioned above are not efficient in addressing the issue of the DVRPTW. For example, MIACO may perform well in cyclic scenarios, but poorly in others. EIACO might also suffer from local optima issues, where the algorithm converges to a local optimal solution and fails to find the global optimum. The RIACO algorithm typically relies on probabilistic models to address uncertainty, which may not always accurately represent real-world uncertainties. Hence, this paper designs a new strategy for adjusting the pheromone matrix to effectively address the random presence and disappearance of rescue targets in a dynamic environment.

Furthermore, in order to enhance the population diversity and search abilities of ants, simulated annealing (SA) is incorporated into ACO for solving the DVRPTW. SA is a metaheuristic algorithm inspired by the annealing process in metallurgy. It mimics the slow cooling of a material, allowing its atoms to settle into a low-energy state. In the context of optimization, SA can help ACO escape local optima and explore a wider search space by accepting suboptimal solutions with a certain probability.

The dynamic generators [26,27] utilized in the literature are typically based on known optimal solutions, meaning that the optimal solution remains unchanged throughout the environmental changes. While this approach ensures that we can compare the algorithm's performance against the optimal value, it does not accurately reflect the real-world scenario where changes are often unpredictable. In this paper, in order to better simulate real-world scenarios, we have designed a random dynamic generator that accounts for the unpredictable nature of environmental changes.

Given the dynamic nature of logistics, the DVRPTW emerges as a critical concern and opportunity for exploiting operational efficiencies. Recent developments in real-time data acquisition and intelligent transportation systems have only increased the relevance and complexities involved in this pursuit. Despite various efforts in the past and the application of metaheuristic algorithms, like ACO, there still remains a gap in effectively adapting to real-time changes and unpredictable elements in the routing problem. In this regard, the paper addresses the above research questions as follows:

- (1) How can the pheromone matrix for ACO be adapted more effectively so that it is responsive to changing environmental conditions without frequent complete restarts?
- (2) How would SA enhance the diversity and search capability of the ants in a given DVRPTW instance?
- (3) How can a dynamic generator be conceived in the light of the unpredictability of real-world scenarios, to make the simulation of environments realistic, hence offering a good testing ground for new algorithms?

This paper makes the following contributions:

- (1) A novel strategy is proposed to adapt the pheromone matrix and effectively handle random rescue targets in dynamic environments.
- (2) To enhance the overall performance of ACO for solving the DVRPTW, the powerful technique of SA is employed, and a local search operator is designed to further improve the performance of the generated routes.
- (3) Considering the limitations of the existing dynamic reference generator in accommodating real-world scenarios, this paper designs a random dynamic generator to provide a more realistic simulation, thereby achieving a better alignment with actual conditions.

The rest of this article is organized as follows. Section 2 introduces the related work including the basic VRP problems and their dynamic extension and the basic ideas of ACO and SA. Section 3 describes the proposed algorithm for solving the DVRPTW. Section 4 introduces the dynamic benchmark generator. It can generate DVRPTW instances dynamically. Section 5 gives the experimental results and analysis. Finally, Section 6 gives the discussion and future work.

## 2. Related Work

### 2.1. The Basic VRP and Its Variants

In the VRP [7], there are typically a set of vehicles and a set of customers. Each customer has a specific demand. The vehicles need to consider constraints such as vehicle capacity while fulfilling customer demands. The objective of the VRP is to find the optimal routes for a set of vehicles that satisfy all customer demands and minimize the overall costs or maximize the benefits, while meeting the constraints. Common costs include vehicle travel distance, vehicle utilization time, and customer satisfaction (e.g., waiting time).

The mathematical model of the VRP can be represented as follows:

1. Objective function: This measures the overall costs. Travel cost is considered as the core metric for evaluating the quality of a route. It is computed with a certain formula, as given by Equation (1). We can set the best path by assessing different available routes and the travel cost associated with each. The best route gives the least total traveling distance.

$$f(c) = \min \sum_{i=1}^n \sum_{j=1}^n d_{ij} \sum_{k=1}^u x_{ijk} \tag{1}$$

where  $i$  and  $j$  represent node  $i$  and node  $j$ , respectively. The variable  $d_{ij}$  represents the distance between node  $i$  and node  $j$ , and  $x_{ijk}$  indicates whether the  $k$ th vehicle passes through node  $i$  and node  $j$ . The symbols  $n$  and  $u$  represent the total number of nodes (i.e., depot and customers) and the total number of vehicles, respectively.

2. Constraints: These are conditions that restrict the vehicle routes. Common constraints include vehicle capacity limits, time window limits, and vehicle travel distance limits.

$$\sum_{i=1}^n q_i \sum_{k=1}^u x_{ijk} \leq Q, \forall i, j \in \{1, \dots, n\}, \forall k \in \{1, \dots, u\} \tag{2}$$

$$\sum_{i=1}^n x_{isk} - \sum_{j=1}^n x_{sjk} = 0, \forall k \in \{1, \dots, u\}, \forall s \in \{0, \dots, n\} \tag{3}$$

$$\sum_{i=1}^n x_{i0k} \leq 1, \forall k \in \{1, \dots, u\} \tag{4}$$

$$\sum_{j=1}^n x_{0jk} \leq 1, \forall k \in \{1, \dots, u\} \tag{5}$$

Equation (2) ensures that the vehicle load cannot exceed the maximum load, where  $q_i$  represents the demand of customer  $i$ . Equation (3) ensures that the vehicle serves the customer, as well as leaving the customer immediately, and Equations (4) and (5) ensure that each vehicle is dispatched at most once.

In order to consider the delivery time requirement, the VRPTW arises [11]. The VRPTW not only considers all the constraints of the VRP, but also needs to consider the time window constraint. That is, it is required that each customer be served within a specific time window  $[e_i, l_i]$ , where  $e_i$  and  $l_i$  represent the earliest and latest service time of node  $i$ , respectively. As shown in Equations (6) and (7),

$$t_i + w_i + s_i + travel_{ij} = t_j, \forall i, j = 0, \dots, n, i \neq j \tag{6}$$

$$e_j \leq t_j \leq l_j, \forall j = 0, \dots, n \tag{7}$$

where  $t_i$  indicates the arrival time of node  $i$ ,  $w_i$  indicates the waiting time of node  $i$ ,  $s_i$  indicates the service time of node  $i$ ,  $travel_{ij}$  indicates the time to pass the distance from node  $i$  to node  $j$ , and  $t_j$  indicates the arrival time of node  $j$ . Equation (6) calculates the arrival time of node  $j$ , and Equation (7) ensures that the arrival time is within the range  $[e_j, l_j]$ .

The characteristic of the DVRP [15,28] is that the problem is partially known in advance and changes dynamically during operation. Most DVRPs can be divided into three categories [29]: the DVRP with random requests (DVRPRR), the DVRP with random demands (DVRPRD), and the DVRPRD with random travel times (DVRPRTT). In the DVRPRR, the customer waiting for service is not fully aware in advance, but dynamically places the request within the planned scope during the process. In the DVRPRD, the location of the customer is known. However, the quantity required is only known when the vehicle arrives at the customer's location. In the DVRPRTT, the travel time between locations is related to the speed of the vehicle and random fluctuations in traffic conditions.

This paper focuses on the maritime search and rescue tasks using UAVs in a dynamic environment. We suppose that all UAVs start from the same base station. Drones can make round trips in the disaster area without considering flight distance limitations. They are, however, subject to payload restrictions, meaning the amount of rescue supplies they can carry is limited. When a disaster strikes, the affected area is divided into small rescue zones (which can be represented as nodes in a graph). UAVs depart from the base station to provide aid to these zones, delivering relief supplies based on the actual number of affected people in each area. Each rescue area has a time constraint, as rescue efforts become futile beyond the upper limit of the time frame. Additionally, as time progresses, new affected areas may emerge, as previously introduced, and some existing disaster sites may no longer require assistance, making this a dynamic problem. In summary, the maritime search and rescue problem is modeled as the DVRPTW, where the rescue targets fluctuate randomly. To describe dynamism, the degree of dynamism (DoD) is defined by Lund et al. [30], as shown in Equation (8).

$$DoD = \frac{n_d}{n_d + n_s} \in [0, 1] \quad (8)$$

where  $n_d$  indicates the dynamic requirement and  $n_s$  indicates the known static requirement.

## 2.2. Literature Review

The VRP is a fundamental problem in logistics and operations research, involving the design of optimal routes for a fleet of vehicles to deliver goods or services to a set of customers. Due to its NP-hard nature, researchers have developed various heuristic and metaheuristic algorithms to find near-optimal solutions within reasonable computation times. Traditional methods such as the Clarke–Wright Savings Algorithm [31] and Branch-and-Bound method [32] have been extensively studied. However, with the increasing complexity of modern logistics, researchers have begun exploring more sophisticated approaches, such as genetic algorithms (GAs), ACO, and Tabu Search (TS), which have significantly advanced the solutions to the VRP. Barbarosoglu and Ozgur (1999) proposed a TS algorithm for the single-depot VRP, generating new neighborhoods by considering the distribution pattern of the dealers' locations [33]. Baker and Ayechev (2003) developed a GA for the basic VRP, demonstrating its widespread application to combinatorial optimization problems [34]. Wu and Gao (2023) proposed an ACO-based method for solving the Vehicle Routing Problem with Simultaneous Pickup-Delivery and Time Window (VRP-SPDTW), incorporating destroy and repair strategies to enhance global search capabilities and avoid local optima [35]. Souza et al. (2023) introduced a hybrid algorithm combining Differential Evolution and local search for the Capacitated Vehicle Routing Problem (CVRP), demonstrating high efficiency across multiple classical datasets [36]. Vincent et al. (2024) developed a Simulated Annealing with Variable Neighborhood Descent (SAVND) algorithm for the Heterogeneous Fleet Vehicle Routing Problem with Multiple Forward/Reverse Cross-Docks (HF-VRP-MFRCD), providing optimal solutions for small-scale instances and outperforming the GUROBI solver for larger instances [37].

The VRPTW is an extension of the VRP where each customer must be served within a specific time window. This additional constraint increases the problem's complexity, but also makes it more applicable to real-world scenarios requiring strict delivery times. Frey et al. (2023) proposed the VRPTW with Flexible Delivery Locations (VRPTW-FL)

model, allowing deliveries at flexible alternative locations, applicable in parcel delivery and medical service scheduling [38]. Ahmed and Yousefikhoshbakht (2023) studied the Heterogeneous Fixed Fleet Open Vehicle Routing Problem with Time Windows (HF-FOVRPTW), introducing Mixed-Integer Linear Programming and an improved Tabu Search algorithm [39]. Wang et al. (2023) addressed the Multi-Depot VRP with Time Windows and Three-Dimensional Loading Constraints (MDVRPTW-TDLC), using customer-clustering-based resource-sharing methods and vehicle compartment partition policies to optimize operational costs [40]. Lehmann and Winkenbach (2024) presented a matheuristic for the Two-Echelon Multi-Trip VRP with Deliveries, Pickups, and Time Windows (2E-MT-VRP-PTW), integrating exact formulations for first-echelon routing and adaptive large neighborhood search for second-echelon routing [41]. Yu et al. (2024) proposed a simulated annealing solution for the Multi-Depot Waste Collection VRP with Time Windows and Self-Delivery Option (MDWCVRPTW-SDO) [42]. Cavecchia et al. (2024) developed a Decision Support System (DSS) based on a micro-service architecture for Multi-Trip VRP (MT-VRP), applied in pharmaceutical distribution [43]. Lee and Jeong (2024) introduced an optimized routing strategy for accessible taxis based on the travel behavior of people with disabilities, using Gaussian Mixture Models [44]. Luo et al. (2024) proposed a two-stage heuristic algorithm for the Electric Vehicle Routing Problem with Time Windows (E-VRPTW), incorporating dynamic programming and supercharging station factors [45].

The DVRP further extends the VRP by considering the dynamic nature of real-world logistics, where customer requests can change in real time, requiring adaptive routing adjustments. The DVRP is particularly challenging as it necessitates real-time decision-making and adaptation to constantly changing conditions. Ghannadpour et al. (2014) presented a multi-objective DVRP with fuzzy time windows, where real-time requests arrive randomly and are managed using a genetic algorithm to optimize fleet size, travel distance, and customer satisfaction [46]. Gholami-Zanjani et al. (2019) proposed a cooperative strategy for the DVRP in disaster relief, using a mixed-integer nonlinear model and a genetic algorithm to solve multi-vehicle routing problems [47]. Kucharska (2019) introduced an Algebraic Logical Meta-Model (ALMM) for the DVRP, considering the dynamic appearance of customers [48]. Zacharia et al. (2021) considered the VRP with fuzzy payloads to minimize travel distance and fuel consumption [49]. da Silva Junior et al. (2021) proposed a framework using ant colony systems and variable neighborhood descent for the DVRP with time windows [50]. Zajkani et al. (2021) developed a model predictive approach to the DVRP, accounting for traffic congestion and utilizing distributed cooperative predictive methods for optimization [51]. Sabar et al. (2021) proposed a population-based iterated local search method for the DVRP, significantly improving search performance through evolutionary operators [52]. Xu et al. (2022) studied the DVRP with limited supply and a variable neighborhood region in refined oil distribution, presenting a multi-objective optimization model and the Fuzzy C-means algorithm [53]. Zhang et al. (2023) introduced a method for solving a large-scale DVRP, using knapsack-based linear models for approximation and optimal acceptance and assignment decision rules [54]. Pan and Liu (2023) proposed a deep reinforcement learning framework for the DU-VRP, incorporating a partially observable Markov decision process for the real-time observation of customer demand changes [55]. Kim (2023) studied a DVRP model with fuzzy customer response, proposing routing strategies to reduce customer complaints and potential losses [56].

The above-formulated studies with all classified objective functions, constraints, and solution algorithms used are of the model type, objective functions, model characteristics, and solution algorithms. The following Table 1 itemizes the typical objective functions, which include time windows, customer satisfaction, homogeneous fleets, heterogeneous fleets, single depots, multiple depots, and the level of dynamicity. Model characteristics pertain to the nature of the decision made, that is allocation, routing, and/or inventory, and these characteristics translate to the following for the scope of the recent studies:

- (1) Most recent studies were solely on the optimization of vehicle routing with time windows; however, only a few research works took customer satisfaction into account.

- (2) Most studies deal with homogeneous fleets; only a few are based on heterogeneous fleets.
- (3) Most of the studies from the literature are based on single-depot routing, with very few based on multiple depots and dynamicity.
- (4) Most of the studies in the literature are based on mixed algorithms and metaheuristic algorithms such as the GA, TS, and ACO.

**Table 1.** Collection of relevant studies in the area of VRP variations.

Ref	Method	Time Windows	Customer Satisfaction	Homogeneous Fleet	Heterogeneous Fleet	Single Depot	Multiple Depots	Dynamic
[9]	HGA	✓	✓	✓		✓		✓
[12]	MOMFMA	✓		✓		✓		
[16]	GA-PSO	✓		✓			✓	✓
[22]	PPL-ACO			✓		✓		✓
[23]	MACO			✓		✓		✓
[31]	Savings			✓		✓		
[32]	Branch-and-Bound			✓		✓		
[33]	TS			✓		✓		
[34]	GA			✓		✓		
[35]	ACO-DR	✓		✓		✓		
[36]	DE-LS			✓		✓		
[37]	SAVND				✓		✓	
[38]	ALNS	✓		✓		✓		
[39]	MILP	✓			✓		✓	
[40]	ENSGA-II	✓			✓		✓	
[41]	Matheuristic	✓		✓		✓		
[42]	SA	✓		✓		✓		
[43]	DSS	✓		✓		✓		
[44]	VRP-GMM	✓		✓		✓		
[45]	Two-stage	✓		✓		✓		

Abbreviations—HGA: Hybrid Genetic Algorithm, MOMFMA: Multi-Objective Multi-Factorial Memetic Algorithm, GA-PSO: Genetic Algorithm-Particle Swarm Optimization, PPL-ACO: Pairwise Proximity Learning-based Ant Colony Optimization, MACO: Memetic Algorithm based on Ant Colony Optimization, TS: Tabu Search, GA: genetic algorithm, ACO-DR: Ant Colony Optimization with Destroy and Repair, DE-LS: Differential Evolution with Local Search, SAVND: Simulated Annealing with Variable Neighborhood Descent, ALNS: Adaptive Large Neighborhood Search, MILP: Mixed-Integer Linear Programming, ENSGA-II: Extended Non-dominated Sorting Genetic Algorithm II, Matheuristic: mathematical optimization with heuristic, SA: simulated annealing, DSS: Decision Support System, VRP-GMM: Vehicle Routing Problem with Gaussian Mixture Model.

These studies not only contribute to the theoretical development of the VRP, but also provide efficient algorithms to effectively address real-world maritime search and rescue missions, ensuring timely and effective rescue operations. These works also promote advanced optimization techniques that will be effectively used in the development of intelligent maritime rescue systems. Furthermore, they will ensure sustainable maritime safety management and provide decision support for smart maritime operations.

### 2.3. Ant Colony Optimization

ACO [19] is a meta-heuristic algorithm inspired by the foraging behavior of ants in search of food. It solves combinatorial optimization problems by simulating the behavior of ants in path selection and information exchange.

The basic idea of ACO is that ants guide each other’s choices by depositing pheromones during the search process. Ants release a chemical substance called a pheromone on the path as they move, and the concentration of the pheromone represents the goodness of the path. Other ants perceive and respond to these pheromones, tending to choose paths with higher concentrations. Over time, ants gradually concentrate on better paths, leading to the discovery of better solutions to the problem. The specific process is shown in Figure 1.

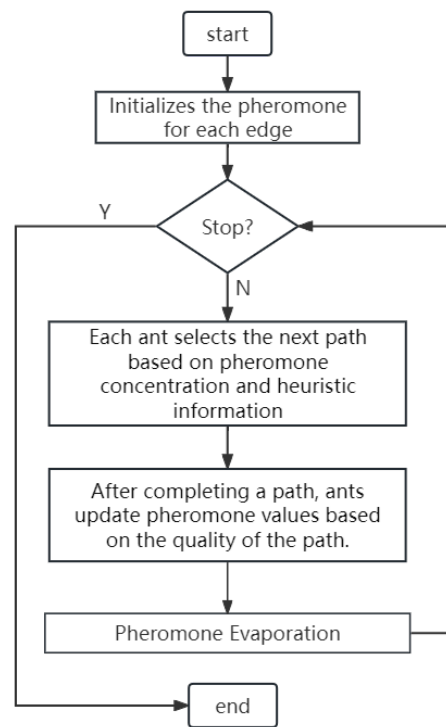


Figure 1. The flowchart of ACO.

Let  $\tau_{ij}$  be the pheromone intensity of the edge when an ant moves from  $i$  to  $j$ . The initial pheromone concentration is generally set to a constant. The intensity of the trail should be updated according to Equation (9).

$$\begin{cases} \tau_{ij}(t+n) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \\ \Delta\tau_{ij}(t) = \sum_{k=1}^n \Delta\tau_{ij}^k(t) \end{cases} \quad (9)$$

where  $\rho$  is the pheromone volatility factor, which lies within the value range of  $[0, 1]$ . The variable  $\tau_{ij}(t)$  represents the pheromone residual factor. The term  $\Delta\tau_{ij}(t)$  is the pheromone increment on the path  $(i, j)$  of this cycle, that is the sum of the amount of pheromone released by all ants on the path  $(i, j)$ . Initially,  $\tau_{ij}(t) = 0$ . The variable  $\Delta\tau_{ij}^k(t)$  represents the increment of pheromone released by the  $k$ th ant on the path  $(i, j)$ , which is calculated as Equation (10).

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k}, & \text{If ant } k \text{ visits the edge } (i, j) \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

where  $Q$  is a constant, which has a certain effect on the convergence speed of the algorithm. The variable  $L_k$  represents the total distance of the path taken by ant  $k$  in this cycle.

ACO for path optimization mainly includes two stages: path construction and pheromone updating. In the process of exploration, ants judge the next node to be visited by sensing the concentration of pheromone. Let us assume that the probability of ants moving from node  $i$  to node  $j$  is expressed, and the way for calculating the probability is shown in Equation (11).

$$P_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}(t)]^\beta}{\sum_{s \in J_k} [\tau_{is}(t)]^\alpha \cdot [\eta_{is}(t)]^\beta} & j \in J_k \\ 0 & j \notin J_k \end{cases} \quad (11)$$

where  $J_k$  represents the node set that the  $k$ th ant can select at time  $t$  and the variable  $\alpha$  is the information heuristic factor, reflecting the importance of the residual pheromone in the ant's exploration process. The variable  $\beta$  is an expectation heuristic factor, reflecting



the relative importance of path expectation in ant exploration. Both  $\alpha$  and  $\beta$  influence the balance between exploration and exploitation in ant behavior during path selection in ACO. The variable  $\eta_{ij}$  represents the expected degree of an ant moving from node  $i$  to node  $j$ , which can be calculated as Equation (12). Here,  $d_{ij}$  represents the distance of moving from node  $i$  to node  $j$ .

$$\eta_{ij}(t) = \frac{1}{d_{ij}} \quad (12)$$

When each ant walks through the paths, the pheromone matrix needs to be updated, and the calculation formula of pheromones on the edge  $(i, j)$  at time  $(t + n)$  is according to Equation (9).

#### 2.4. Simulated Annealing

SA [57] is a global optimization algorithm inspired by the annealing process of solids. It simulates the behavior of solid materials as they cool down, gradually reaching a low-energy state and finding the global optimal solution to a problem.

The basic idea of SA is to simulate the random movements of particles in a solid at high temperatures and gradually cool them down. During the search process, the algorithm accepts suboptimal solutions with a certain probability, allowing for exploration of the solution space and avoiding getting trapped in local optima. As time progresses, the algorithm reduces the probability of accepting suboptimal solutions, leading the search to converge towards the global optimum.

We outline the pseudocode for SA in Algorithm 1. It begins with the initialization in lines 1–3, where the point and temperature initializations are random, and such features form the core within this algorithm. The main loop (lines 4–10) generates a neighboring solution of the current solution, evaluates the probability of accepting the new solution with the computed difference in energy between the two solutions, and updates the current solution with the new solution if accepted. If this difference in energy  $\Delta E$  between a new solution and the current solution is less than 0 ( $\Delta E < 0$ ), then take the new solution as the current one. If the energy is higher, accept the new solution with a probability  $\exp(-\Delta E/T)$ , where  $T$  is the current temperature. It was accepted that the algorithm would be probabilistic in nature and allowed the ability of moving out from local optima, thus helping enhance the search ability of the algorithm toward the global solution. The cooling of the temperature should be performed after each iteration such that the algorithm can smoothly converge toward solutions of high quality according to the cooling rate. Line 11: Continue with the algorithm further until some stopping condition is met (e.g., the minimum temperature reached or the number of iterations completed). Finally, the search process terminates with the best solution available so far. The following structured description clearly states the core mechanics of the simulated annealing algorithm and how the algorithm is used to solve optimization problems.

---

#### Algorithm 1 Simulated annealing update procedure.

---

```

1: Initialize  $s$  with a random solution
2: Initialize temperature  $T$  to a suitable high value
3: Initialize cooling rate  $\alpha$ 
4: while termination conditions not met do
5:    $s' \leftarrow$  Generate a neighbor of  $s$ 
6:   Calculate  $\Delta E = E(s') - E(s)$ 
7:   if  $\Delta E < 0$  or  $e^{-\Delta E/T} \geq \text{random}(0, 1)$  then
8:      $s \leftarrow s'$ 
9:   end if
10:   $T \leftarrow \alpha \times T$ 
11: end while
12: return  $s$ 

```

---

### 3. The Proposed Dynamic Ant Colony Optimization for Dynamic Vehicle Routing Problem with Time Windows

To effectively address the proposed DVRPTW that is modeled based on maritime search and rescue, this paper designs a dynamic ACO, named DACO. One of the primary characteristics of DACO is that it dynamically adjusts the pheromone values in the matrix to adapt to the new conditions. This adaptation allows the ants to explore alternative routes that may be more efficient in the updated environment.

The proposed DACO is described in Algorithm 2. The variable  $S'$  is used to record the best routes in the evolutionary process. During the initialization stage (lines 1–4), the counter  $i$  is initialized to 0 and all the customers except the depot  $v_0$  are unserved initially. The initial pheromone value on each edge is set. Then, the optimization process begins. In this process, the environment may change, for example new rescue targets may emerge, while others, such as those that have been rescued or are no longer accessible, may disappear from the list of active rescue points. If environment changes are detected, the pheromone matrix will be updated dynamically based on the proposed strategy (line 7), which can be seen in Section 3.2. If it is not the first generation,  $S'$  will have a specific value that becomes inaccessible. We also update  $S'$  to ensure that we have feasible routes (line 9).

---

**Algorithm 2** The pseudocode of DACO for the DVRPTW.

---

**Require:** Graph  $G = (V, E)$ , where  $V$  is the set of vertices, which represent the rescue targets and the base station of the UAVs, and  $E$  is the set of edges, with each edge's length corresponding to the flight distance between its two vertices.

**Ensure:** A feasible solution  $S'$

```

1:  $i = 0$ , unserved targets set  $U \leftarrow V \setminus v_0$ ;
2: for edge  $e \in E$  do
3:   Set the initial pheromone value  $M(n, n)$ ;
4: end for
5: while  $i < max\_iter$  do
6:   if dynamic changes then ▷ Strategy Section 3.2
7:     Update( $M(n, n)$ );
8:     if  $S' \neq \emptyset$  then
9:       Update( $S'$ );
10:    end if
11:   end if
12:   for ant  $k \in POP$  do ▷ Construct a feasible routes
13:      $S_k \leftarrow (v_0)$ ;
14:     while  $U \neq \emptyset$  do
15:        $U' \leftarrow Filter(U)$ ; ▷ Filter the feasible targets
16:       if  $U' = \emptyset$  then
17:          $S_k \leftarrow (S_k, v_0)$ ;
18:       else
19:          $u \leftarrow Select(U')$ ; ▷ Strategy Section 3.3
20:          $S_k \leftarrow (S_k, u), U \leftarrow U \setminus u$ ;
21:       end if
22:     end while
23:     Calculate the total cost  $f(c_k)$  of  $S_k$ ;
24:   end for
25:    $S^* = \arg \min_{k \in POP} f(c_k)$ ; ▷ Record the best routes
26:    $S^* = \arg \min \{f(S'), f(S^*)\}$ ;
27:    $S' = Optimize(S^*)$ ; ▷ Strategy Section 3.4
28:   Update the pheromone matrix  $M(n, n)$  based on  $S'$ ;
29:    $i ++$ ;
30: end while
31: return  $S'$ ;

```

---

Then, each ant, who is located at the base station at the beginning, goes to construct a feasible route. In this process, a subset of candidate rescue targets is firstly selected (line 15). The selection criteria are that the supplies required at these rescue targets must be within the remaining payload capacity of the UAV, while also meeting the upper limit of its rescue waiting time. It explicitly removes the candidates whose time windows exceed the limits, as well as those whose demands exceed the capacity of the UAV. If the subset is empty, then the ant goes back to the depot to refill (lines 16–17). Otherwise, a rescue target is selected from the subset. In traditional ACO, Equation (11) is used to decide the next target that the ant should go to. In the proposed DACO, the Metropolis rule in SA is combined with the probability in ACO to select the next target. The details can be seen in Section 3.3. This process is repeated until all targets have been added to the routes (lines 13–22). Then, the total cost (i.e., flight distance) of the routes is calculated (line 23), and the best routes with the minimal costs are recorded (line 25). A comparison between the costs of  $S'$  and  $S^*$  is carried out, and the best one will be retained. After this, a local search operator is designed and used to further optimize the best routes (line 27). The details of the local search operator are introduced in Section 3.4. Then, the pheromone matrix is updated based on the best routes generated in the above steps (line 28). Then, the counter is incremented by one, and the new iteration is started, i.e., the routes are constructed according to the updated pheromone matrix. When the maximum number of iterations ( $max\_iter$ ) is reached, the algorithm stops.

### 3.1. Solution Generation

Solution generation is performed as follows: Ants leave the base station and scan targets who can be served subject to the time limit and the remaining capacity of the UAV. The selection probability of each target is computed according to Equation (11). Then, the next target to be served is determined by either the highest selection probability or randomly, as detailed in Section 3.3. As infeasible solutions are rejected in the screening process, all solutions generated are feasible. If no serving targets are found, then the ant returns to the base station, and a new ant is sent to serve targets again until all of them are served.

As shown in Figure 2, node 0 represents the base station, and nodes 1, 2, 3, 4, 5, and 6 represent the targets. Initially, start from 0. After filtering, all targets meet the constraint conditions. Then, calculate the selection probability based on Equation (11), and then select a target. Suppose target 1 is chosen. After filtering again, only targets 2, 4, and 6 meet the constraint conditions. Repeat the above steps, and choose target 2, followed by target 4. After that, no customers meet the constraint conditions, so return to 0.

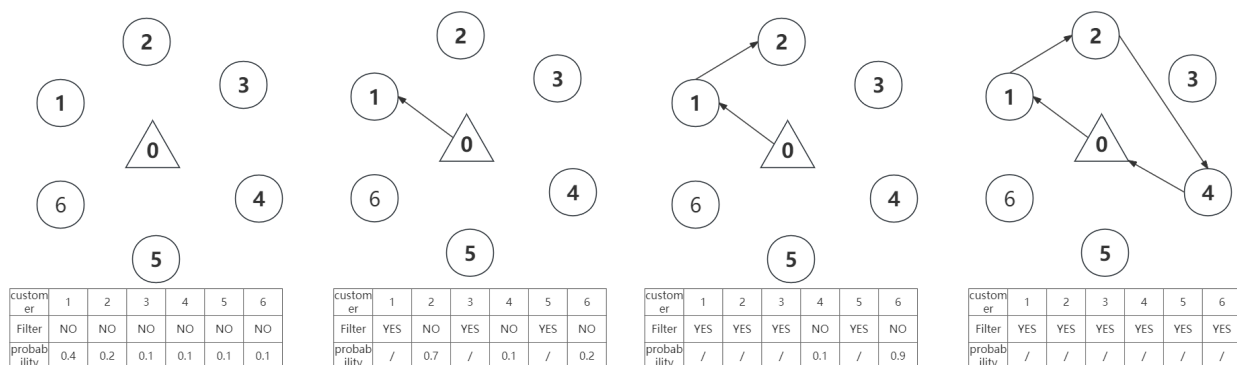


Figure 2. An example of solution generation.

### 3.2. Dealing with Dynamic Changes

The dynamic changes in the DVRPTW considered in this paper are divided into two types: (1) new targets appear and (2) the original targets disappear. The strategies

to deal with environment changes are to set a flag bit for each node. When dynamic changes occur, only the tag bits need to be processed. The specific strategies are shown in Algorithm 3, which can be explained as follows:

- (1) If it is a new target, then the pheromone matrix is expanded from the original  $n \times n$  to  $(n + 1) \times (n + 1)$ . The values in the columns and rows where the new nodes are located are filled according to Equation (12), and the remaining parts are filled by the original matrix (lines 1–9).
- (2) If it is to cancel a target, the values in the columns and rows in the pheromone matrix where the canceled target is located are set to 0 (lines 10–17).

The algorithm will continue to evolve based on the modified pheromone matrix.

---

**Algorithm 3** Update( $M(n, n)$ ).

---

```

1: if  $x$  new targets appear then
2:   for  $i \in [1, x]$  do
3:     for  $j \in [1, n + x]$  do
4:        $M[n + i][j] = 1/d_{(n+i)j}$ ;
5:        $M[j][n + i] = 1/d_{j(n+i)}$ ;
6:     end for
7:   end for
8:    $n \leftarrow n + x$ 
9: end if
10: if  $y$  targets cancel their requirements then
11:   for  $i \in [1, y]$  do
12:     locate the number  $s$  of the canceled target;
13:     for  $j \in [1, n]$  do
14:        $M[s][j] = 0, M[j][s] = 0$ ;
15:     end for
16:   end for
17: end if

```

---

In cases where there are dynamic changes, the optimal routes will be inaccessible. In order to ensure the existence of feasible routes at all times, the original optimal routes will be updated as follows:

- (1) For the addition of a new target, a new UAV will be assigned to serve it. The reason for this operation is to avoid violating constraint conditions, as assigning a new UAV ensures that the target can be serviced without conflicting with the constraints. Subsequently, in the optimization process, the targets served by the newly added UAV can be merged into the routes of the existing UAVs, thereby achieving a more optimal solution over time.
- (2) For the cancellation of a target, it is straightforward to remove that target from the original route. In this scenario, since it is guaranteed that the updated routes will always be compliant with the constraint conditions, there is no need to perform additional checks or considerations regarding constraint violations.

### 3.3. Improved Ant Colony Optimization Based on Simulated Annealing

The idea that the probability is proportional to the temperature in SA is applied to ACO. In the new algorithm, a random number  $x \in (0, 1)$  is generated firstly, which follows a uniform distribution. Then,  $x$  is used to compare with the probability  $\theta$  calculated in Equation (13), where  $i$  and  $max\_iter$  represent the current number of iterations and the maximum number of iterations, respectively. If  $x$  is less than  $\theta$ , the node with the largest probability, calculated as Equation (11), is selected. Otherwise, the roulette algorithm is used to randomly generate the next visited node.

$$\theta = e^{\frac{-i}{max\_iter}} \tag{13}$$

### 3.4. Local Search Operator

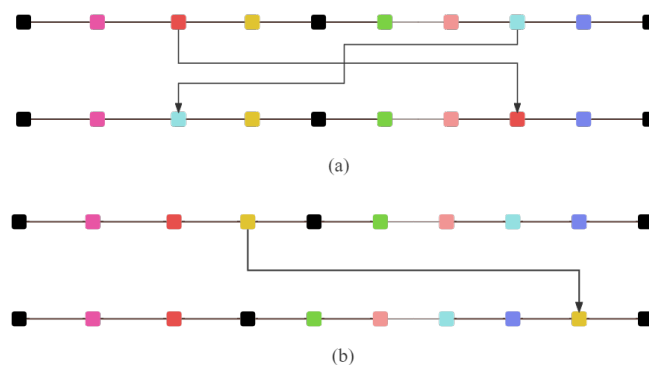
This section proposes a local search operator to further optimize the best routes obtained in each generation. Firstly, the routes are divided into high-load or low-load according to their load, which is calculated as the total demands of targets served by them. The division is given in Equation (14), where  $c\_load$  and  $max\_load$  represent the current load of the route and the maximal load of these routes, respectively. If  $\lambda$  is less than 0.5, it is considered as low-load. Otherwise, the route is high-load.

$$\lambda = \frac{c\_load}{max\_load} \tag{14}$$

For the high-load routes, the exchange strategy is carried out on them. The exchange strategy learns from the idea of 2-OPT [58], but restricts the exchange between two routes. That is, a random node from one route is chosen to exchange with the node of another route. If the new routes obtained through the exchange process can satisfy the constraints and result in lower total costs compared to the original routes, the new routes are accepted and the process is terminated. However, if the new routes fail to meet these criteria, the exchange process will be attempted a limited numbers of times.

Then, for the low-load routes, the split strategy is used on them. First, two low-load routes (e.g.,  $A$  and  $B$ ) are selected. Then, one of them (e.g.,  $A$ ) is chosen for splitting, i.e., a random node from  $A$  is selected to attempt to insert into the other route (i.e.,  $B$ ). The insertion starts from the first position, and after each insertion attempt, the constraint conditions including the time window and capacity of the UAVs are checked. The process continues sequentially until every node of this route has been attempted. If the conditions are not met, another node from  $A$  is tried until all nodes have been attempted. The goal is to reduce one route and transfer nodes in the low-load routes to other eligible low-load routes.

As can be seen from Figure 3a, the red node is exchanged with the blue node, and in Figure 3b, the yellow node is inserted from the original route to another route.



**Figure 3.** Illustrations of (a) the exchange strategy and (b) the split strategy of the local search operator. The black node represents the base station, and the other nodes represent rescue targets.

### 3.5. Superiority of Dynamic Ant Colony Optimization

The proposed DACO has three advantages:

- (1) Quickly generating feasible solutions to handle dynamic changes: DACO can swiftly generate feasible solutions to address dynamic changes in the environment. This rapid response capability allows it to adjust route planning more flexibly in real-time situations, ensuring the algorithm’s applicability and practicality.
- (2) Incorporating the Metropolis rule in SA to select the next target: DACO integrates the Metropolis rule from SA when selecting the next target. Compared to traditional ACO, this enhancement significantly improves the diversity of the solutions. By exploring more possibilities in the solution space, DACO can avoid local optima and enhance the overall quality of the solutions.

- (3) Designing a local search operator to further optimize route performance: To further improve the performance of optimal routes, DACO includes a local search operator. This operator fine-tunes the paths within a local scope, enabling adjustments based on the existing solutions to find better routes. This approach effectively enhances the efficiency and accuracy of route planning, ensuring the superiority of the final solutions.

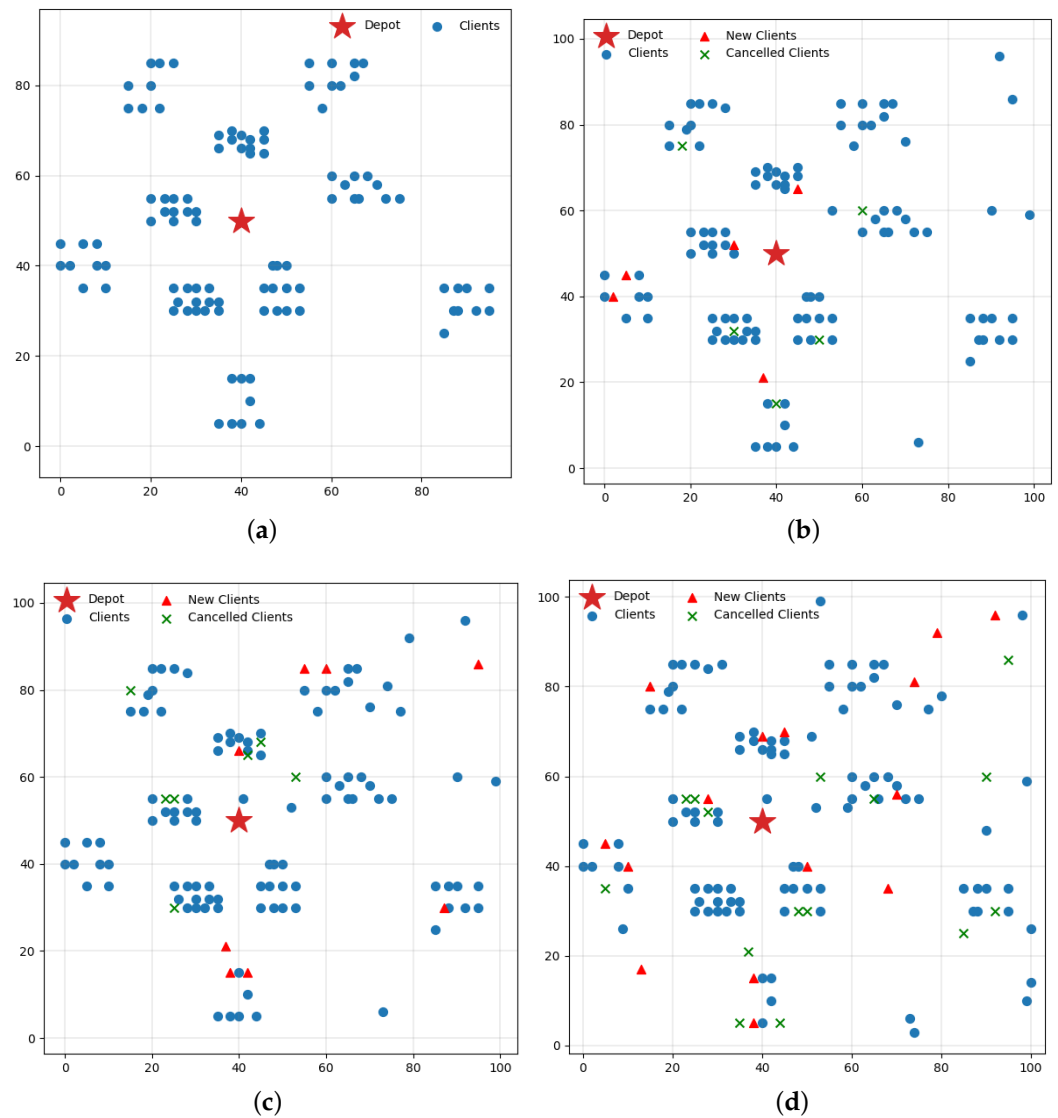
These three improvements enable DACO to not only increase the algorithm's response speed and flexibility, but also enhance solution diversity and optimization capability, making it perform excellently in solving dynamic-route-optimization problems.

#### 4. Dynamic Benchmark Generation

The existing generator of dynamic routing problems is DBGP [59], which can transform the static environment into a dynamic environment. However, the dynamic environment in DBGP is simulated by exchanging existing nodes to ensure that the optimal solution remains unchanged. This strategy can be seen as symmetric transformation. The benefit of DBGP is that it is easy to judge the gap between the obtained solution in a dynamic environment and the known optimal solution in a static environment, and thus easy to measure the quality of the algorithm in dealing with dynamic changes. However, the dynamic environment that happens in real life is basically asymmetrical, that is random. Hence, a random benchmark generator based on asymmetric variation for the DVRPTW is proposed in this paper. Specially, the dynamic changes are divided into the following two categories:

- (1) Dynamically add new target requirements: This corresponds to generating new nodes randomly. In the proposed generator, new nodes are generated by copying existing nodes and modifying their coordinates while keeping other features such as demand and time windows unchanged. The process begins by randomly selecting a node, let us say node  $i$ , from the existing nodes in the graph. The coordinates of node  $i$  are denoted as  $(X_i, Y_i)$ . A new node, denoted as  $i'$ , is then created at the position  $(\zeta_1 X_i, \zeta_2 Y_i)$ , where  $\zeta_1$  and  $\zeta_2$  are scaling factors determining the relative position of the new node compared to the original node.
- (2) Dynamically cancel target requirements: This corresponds to deleting the original nodes randomly. In this process, a random number generator is used to choose nodes uniformly at random, and then, the selected nodes are removed from the graph, including all their associated edges.

In the evolutionary process of the algorithm, dynamic changes are triggered at fixed iteration intervals. Figure 4 illustrates the use of the proposed generator to generate three dynamic environments based on the instance of C101 of Solomon's benchmark [11], which has a detailed description in Section 5.3. Figure 4a illustrates the coordinates of the original nodes. Figure 4b–d illustrate the progression of dynamic changes, depicting an increasing frequency from slower to faster. It can be seen that the addition of new requirements will increase the complexity of the problem, while the removal of the original requirements will decrease the complexity. The more frequent the addition and removal of requirements, the greater the deviation from the original problem, and then, it will be harder to solve.



**Figure 4.** The dynamic instances are based on Solomon’s benchmark C101 dataset. (a) shows the coordinates of the original nodes. (b–d) show the process in which the frequency of dynamic changes from slower to faster. The dynamic changes are triggered every 30, 15, and 10 iterations for (b), (c), and (d), respectively.

### 5. Experimental Studies

In this section, we first introduce the compared algorithms, and then provide the parameter settings and the DVRPTW benchmark instances. Finally, we present the experimental results.

#### 5.1. Experiment Design

In this paper, five algorithms were adopted for comparison, Max–Min Ant System (MMAS) [60], RIACO [25], EIACO [25], MIACO [25], and ACO [61].

The selection of these algorithms for solving the DVRPTW was based on the following considerations: The MMAS [60] is an enhanced version of ACO, known for its performance in solving combinatorial optimization problems, particularly in the context of the Traveling Salesman Problem (TSP). Its incorporation of local search may make it more adaptable to the DVRPTW. RIACO, EIACO, and MIACO [25] are specifically designed to handle the DVRP. The introduction of immigrant schemes allows the algorithms to adapt more effectively to real-time changes, which can be advantageous for the DVRPTW. ACO represents the

classical form of ACO with good generality. Its successful application to combinatorial optimization problems makes it an attractive choice.

Independent experiments were repeated 30 times on each instance. The algorithms were implemented via PyCharm (Python 3.10) programming, and all experiments were conducted on a computer configured with an Intel(R) Core(TM) i5-8300H CPU@2.30GHz PC with 8GB RAM.

### 5.2. Parameter Settings

The parameters used by DACO and the compared algorithms are set in Table 2. Two heuristic factors  $\alpha$  and  $\beta$  were set to 1 and 3, respectively, indicating that the concentration of pheromones is relatively more important than the path length in the path selection, but the expected path length is also considered with a lower weight. The pheromone evaporation factor variable  $\rho$  controls the rate of pheromone evaporation in each iteration. A smaller  $\rho$  value (i.e., 0.1) means slower pheromone evaporation, resulting in longer-lasting pheromones on paths, which can make the algorithm rely more on previous choices. The constant  $Q$  determines the amount of pheromone released by each ant on a path. In the MMAS algorithm, the maximum value  $max\_pheromone$  and the minimum value  $min\_pheromone$  for pheromone concentration were set to 2 and 0.01, respectively. These values help control the range of pheromone concentration to avoid spikes or rapid evaporation.

**Table 2.** Parameter settings for the algorithms investigated.

Parameter	Algorithm					
	ACO	EIACO	RIACO	MIACO	MMAS	DACO
$\alpha$	1	1	1	1	1	1
$\beta$	3	3	3	3	3	3
$\rho$	0.1	0.1	0.1	0.1	0.1	0.1
number of ants	20	20	20	20	20	20
max-iteration	300	300	300	300	300	300
$Q$	200	200	200	200	200	200
$k_{shortmemory}$	/	/	5	/	/	/
$k_{longmemory}$	/	/	/	/	/	5
$max\_pheromone$	/	/	/	/	2	/
$min\_pheromone$	/	/	/	/	0.01	/

Abbreviations—ACO: Ant Colony Optimization, EIACO: generation of immigrants based on elitism, RIACO: random generation of immigrants, MIACO: generation of immigrants based on memory, MMAS: Max-Min Ant System, DACO: Dynamic Ant Colony Optimization.

### 5.3. Benchmark Instances

The DVRPTW instances are constructed by the proposed benchmark generator in Section 4 based on Solomon’s benchmark [11]. Solomon’s benchmark is a typical set of problems widely used in the field of the VRPTW. It serves as a standard reference for evaluating the performance of various algorithms.

Table 3 shows the main features of these instances. We built upon the information provided by Solomon, including the coordinates of customers, demand sizes, service times, and vehicle capacities. Additionally, we introduced random customer demands, generating them based on the benchmark’s customer information. We defined two test problems based on Solomon’s benchmarks. These test problems were selected from benchmarks C and R, covering time window constraints for both short and long scheduling horizons.



Table 3. DVRPTW benchmark.

Instance	Number of Customers	Vehicle Capacity	Number of Depots	Description
C101-C109	100	200	1	Randomly generated customers, hard time window constraints, and long scheduling time.
R101-R109	100	200	1	Randomly generated customers, hard time window constraints, and short scheduling time.

5.4. Results and Discussions

Tables 4 and 5 show the average, best, worst, and standard deviations of the compared algorithms on the dynamic instances that trigger random events every 30 and 50 iterations, respectively. For each instance, the better algorithm in terms of “best” is highlighted in bold.

Table 4. The experimental results of triggering random events every 30 iterations. For each instance, the better algorithm in terms of “best” is highlighted in bold.

Dataset	DACO				EIACO				RIACO			
	Mean	Best	Worst	Std.	Mean	Best	Worst	Std.	Mean	Best	Worst	Std.
C101	1259.0	<b>1027.56</b>	2256.72	19.78	1720.39	1304.72	2795.17	36.84	1450.3	1136.35	2727.67	29.27
C102	1260.69	<b>1051.57</b>	2024.54	16.37	1633.47	1300.96	2428.33	26.62	1507.21	1139.38	2451.82	26.4
C103	1276.2	<b>1058.29</b>	1905.49	15.59	1615.74	1335.52	2115.71	20.34	1524.42	1252.99	2096.36	21.18
C104	1195.03	<b>1038.73</b>	1568.89	13.68	1403.03	1202.77	1731.1	13.2	1401.43	1153.11	1754.08	15.58
C105	1249.1	<b>995.56</b>	2291.17	18.78	1723.5	1309.07	2561.28	33.02	1417.35	1097.13	2536.22	31.18
C106	1213.21	<b>973.01</b>	1989.04	20.05	1638.87	1304.17	2393.19	26.3	1451.37	1148.1	2329.84	23.56
C107	1242.37	<b>991.36</b>	2212.63	21.11	1690.9	1338.02	2610.18	31.98	1448.28	1058.85	2617.72	28.52
C108	1156.87	<b>928.7</b>	1760.29	15.6	1544.83	1287.82	2087.45	17.9	1413.82	1065.58	2026.45	19.25
C109	1147.94	<b>959.69</b>	1763.95	13.73	1482.55	1230.22	1926.33	17.33	1376.94	1073.51	1869.28	18.36
R101	2392.71	2000.57	3199.45	31.5	2658.13	2261.57	3213.54	23.94	2219.66	1997.33	3199.54	24.17
R102	2147.98	1861.53	2833.97	21.7	2420.72	2178.33	2867.54	18.79	2042.71	<b>1805.44</b>	2870.74	23.61
R103	1767.92	1571.13	2268.98	21.77	2017.05	1779.67	2360.17	14.85	1808.68	1537.04	2406.33	17.38
R104	1356.61	<b>1196.35</b>	1701.29	11.84	1598.53	1386.09	1894.27	12.59	1552.16	1366.31	1866.3	12.98
R105	2080.56	1757.92	2687.82	26.15	2319.36	2051.24	2868.87	19.81	1985.36	1724.14	2799.33	20.37
R106	1866.97	1623.88	2416.02	22.33	2103.19	1827.61	2550.2	19.28	1892.57	1716.51	2525.57	15.74
R107	1588.85	<b>1377.89</b>	2006.94	18.44	1830.61	1567.59	2175.69	15.97	1681.8	1511.21	2170.33	14.44
R108	1284.5	<b>1087.16</b>	1621.81	12.24	1530.78	1337.66	1812.38	11.27	1506.32	1352.62	1804.89	11.67
R109	1705.72	1496.88	2146.61	17.0	1939.93	1723.99	2296.14	16.02	1758.86	1524.8	2241.21	16.08

Dataset	ACO				MIACO				MMAS			
	Mean	Best	Worst	Std.	Mean	Best	Worst	Std.	Mean	Best	Worst	Std.
C101	1316.33	1071.98	2606.63	22.46	2157.08	1898.93	2684.25	20.17	1394.91	1170.97	2532.14	22.92
C102	1334.22	1108.33	2271.95	18.59	1997.97	1748.45	2457.49	16.32	1429.46	1191.99	2265.24	18.09
C103	1384.51	1174.17	2068.08	16.32	1788.05	1515.49	2163.65	15.67	1466.84	1278.61	2081.82	15.69
C104	1296.04	1105.06	1608.96	12.15	1472.17	1269.4	1819.4	12.55	1311.76	1100.99	1650.8	12.98
C105	1321.07	1076.91	2524.37	22.92	2105.05	1709.86	2640.08	20.18	1389.39	1160.22	2665.18	20.95
C106	1336.05	1046.33	2172.76	19.76	1956.31	1668.13	2431.8	15.53	1392.85	1089.87	2185.18	17.64
C107	1347.82	1120.32	2421.19	23.09	2083.63	1725.97	2591.56	20.73	1386.91	1181.4	2446.44	19.77
C108	1272.34	1004.18	1977.89	19.32	1753.12	1508.41	2059.82	14.95	1338.86	1117.68	2055.39	15.73
C109	1249.75	1028.47	1861.24	14.27	1644.12	1445.49	1921.95	13.28	1293.86	1128.9	1812.04	13.31
R101	2178.61	<b>1975.16</b>	3130.69	22.23	2884.76	2604.43	3278.73	14.76	2251.42	2040.36	3164.49	23.18
R102	2001.89	1807.15	2766.18	22.53	2544.96	2295.28	2877.16	15.38	2077.98	1860.47	2789.32	16.61
R103	1733.04	<b>1530.48</b>	2289.7	16.37	2140.65	1867.37	2395.62	12.98	1768.39	1601.14	2342.59	14.55
R104	1411.07	1239.8	1862.71	12.95	1678.55	1529.98	1891.36	10.23	1432.26	1280.15	1793.98	11.32
R105	1926.83	<b>1708.36</b>	2687.1	18.37	2517.82	2306.84	2822.19	14.07	1958.31	1769.41	2673.45	16.94

Table 4. Cont.

Dataset	ACO				MIACO				MMAS			
	Mean	Best	Worst	Std.	Mean	Best	Worst	Std.	Mean	Best	Worst	Std.
R106	1806.57	<b>1613.27</b>	2461.61	17.02	2247.02	1956.74	2578.71	14.48	1851.47	1689.25	2434.77	16.66
R107	1597.02	1419.55	2117.71	14.25	1949.29	1728.84	2184.97	12.84	1627.77	1425.15	2156.49	13.18
R108	1363.2	1212.77	1768.49	11.55	1625.6	1466.58	1849.8	9.53	1389.05	1230.11	1765.27	10.48
R109	1655.26	<b>1445.72</b>	2179.98	15.8	2048.04	1876.51	2324.04	11.68	1682.91	1531.59	2237.61	13.54

Table 5. The experimental results of triggering random events every 50 iterations. For each instance, the better algorithm in terms of “best” is highlighted in bold.

Dataset	DACO				EIACO				RIACO			
	Mean	Best	Worst	Std.	Mean	Best	Worst	Std.	Mean	Best	Worst	Std.
C101	1250.14	1005.6	2281.07	21.76	1721.03	1379.81	2770.84	35.92	1379.73	<b>977.19</b>	2720.4	31.13
C102	1278.93	<b>1094.03</b>	2040.67	19.73	1657.85	1297.25	2457.41	26.59	1453.49	1191.06	2442.35	23.25
C103	1278.82	<b>1060.8</b>	1889.49	18.47	1625.29	1315.48	2189.89	19.12	1498.23	1199.53	2241.97	21.49
C104	1176.11	<b>1000.67</b>	1543.53	14.06	1383.65	1209.15	1694.16	13.22	1395.04	1199.33	1691.04	14.12
C105	1233.15	1084.32	2177.67	18.92	1704.4	1210.37	2631.59	39.57	1402.09	1121.3	2572.35	29.19
C106	1181.93	978.1	1968.03	19.24	1603.85	1355.5	2328.43	27.35	1415.32	1077.87	2371.7	24.16
C107	1231.13	<b>1046.57</b>	2264.28	16.98	1688.29	1332.72	2545.1	32.32	1379.98	1137.17	2536.56	27.89
C108	1149.6	<b>988.31</b>	1786.11	17.83	1519.86	1253.9	2176.32	21.87	1317.61	1047.72	2085.87	22.85
C109	1143.21	<b>953.16</b>	1680.13	13.96	1456.16	1202.69	1911.96	17.36	1308.46	1068.84	1851.35	20.58
R101	2347.78	2054.95	3172.43	27.7	2692.78	2350.74	3226.58	20.55	2213.94	2002.36	3197.16	22.88
R102	2088.41	1861.41	2845.97	17.71	2424.91	2155.42	2955.54	18.74	2025.08	<b>1793.0</b>	2899.36	22.35
R103	1701.74	<b>1454.1</b>	2267.15	16.89	2015.75	1829.62	2434.43	15.88	1787.92	1619.95	2379.67	16.25
R104	1349.0	<b>1166.56</b>	1687.1	13.21	1585.58	1421.79	1895.29	11.73	1527.74	1332.87	1891.24	14.34
R105	1992.67	<b>1716.34</b>	2677.53	19.88	2321.12	1986.62	2807.52	19.72	1978.8	1760.77	2844.46	20.35
R106	1820.64	<b>1587.27</b>	2411.93	18.44	2108.43	1883.96	2557.92	17.63	1853.08	1672.4	2598.4	18.54
R107	1579.46	<b>1397.29</b>	2009.16	16.35	1837.02	1677.62	2215.71	12.98	1674.65	1479.33	2181.88	15.4
R108	1308.74	<b>1144.23</b>	1615.4	10.04	1530.72	1376.43	1832.04	10.78	1496.95	1292.11	1842.23	12.55
R109	1659.38	<b>1476.41</b>	2125.6	15.13	1954.12	1752.9	2303.97	13.16	1742.81	1533.79	2336.83	15.96

Dataset	ACO				MIACO				MMAS			
	Mean	Best	Worst	Std.	Mean	Best	Worst	Std.	Mean	Best	Worst	Std.
C101	1281.95	1045.19	2537.12	21.9	2167.6	1904.7	2735.97	17.63	1378.49	1198.62	2589.5	19.77
C102	1302.36	1111.29	2310.83	19.69	1958.9	1718.66	2492.01	17.34	1404.55	1121.79	2258.2	19.3
C103	1364.87	1147.88	2096.68	16.38	1763.84	1555.04	2185.47	13.45	1454.03	1328.03	1983.86	13.1
C104	1264.79	1113.71	1669.07	10.77	1442.72	1291.36	1736.35	11.11	1298.64	1143.17	1667.92	11.51
C105	1271.1	<b>998.35</b>	2470.99	23.82	2079.77	1815.67	2684.02	18.97	1378.95	1181.14	2480.73	21.92
C106	1296.07	<b>934.29</b>	2200.08	22.06	1913.07	1703.89	2338.76	15.55	1359.98	1192.1	2187.98	16.9
C107	1303.96	1081.19	2401.99	21.47	2032.59	1709.72	2572.1	16.63	1347.34	1060.23	2419.41	18.35
C108	1249.27	1022.29	1977.8	15.97	1723.95	1389.65	2069.6	14.59	1296.65	1092.63	1938.36	15.4
C109	1216.78	1054.37	1846.57	15.69	1600.85	1417.73	1862.57	12.07	1270.03	1101.01	1856.91	13.27
R101	2183.78	<b>1964.68</b>	3114.8	22.07	2873.44	2632.2	3214.71	15.42	2256.62	2076.56	3097.07	22.4
R102	1999.32	1827.81	2833.78	20.64	2565.83	2344.67	2890.65	14.72	2070.88	1923.44	2883.17	19.3
R103	1727.39	1551.32	2262.37	16.79	2119.47	1915.69	2425.07	11.77	1773.52	1635.25	2268.37	12.83
R104	1400.76	1261.22	1831.92	11.9	1667.4	1457.35	1907.08	10.13	1415.75	1298.62	1822.69	10.6
R105	1930.81	1757.63	2711.44	20.98	2514.05	2281.45	2823.16	13.7	1951.74	1791.79	2689.99	15.4
R106	1808.99	1648.4	2446.19	16.7	2242.16	2082.76	2556.42	11.27	1842.31	1733.02	2446.14	13.5
R107	1590.09	1405.44	2106.44	14.41	1939.62	1767.4	2222.01	10.29	1621.56	1488.55	2148.09	13.32
R108	1353.69	1239.74	1761.09	10.59	1586.54	1430.26	1861.47	9.77	1365.77	1248.92	1763.38	9.69
R109	1650.24	1487.22	2258.27	16.24	2049.05	1891.1	2308.48	10.01	1666.01	1546.07	2273.55	13.26

In Table 4, it is obvious from the obtained results in the 30-iteration experiments that the DACO algorithm, in general, shows mean values that are quite low in comparison with the other algorithms, which indicates better, stable overall performance. For a number of instances, for example R106 and R109, DACO does not achieve the lowest means, but

achieves better worst performance than the other algorithms. This will be the evidence that DACO is consistent and very effective in solving optimization problems and very strong in frequently achieving the best lowest values among all algorithms, as in the instances C101 and C104. For the 50-iteration experiments in Table 5, the mean and best values of DACO dramatically improved, particularly for instances like R103 and R105, in terms of the mean values, making them more stable and reliable. This probably represents an increase in iteration number inside the algorithm that allows better chances for the algorithm to explore more and, hence, improve the quality of the average solutions. On the other hand, DACO was consistently good in finding the best values and kept competing quite well in instances such as C103 and C108, showing good global search capabilities.

In Figure 5, the comparison of algorithm performance on the C101 and R104 datasets, the DACO algorithm demonstrates superior performance among all algorithms, showing the lowest mean and narrower confidence intervals. This indicates its outstanding efficiency and stability in problem-solving. The ACO algorithm also exhibits good performance and stability. In contrast, the EIACO algorithm performs the worst on both datasets, displaying the highest mean and widest confidence intervals, suggesting relatively lower reliability. Other algorithms such as MIACO, MMAS, and RIACO perform moderately, showing some performance fluctuations, but overall averaging.

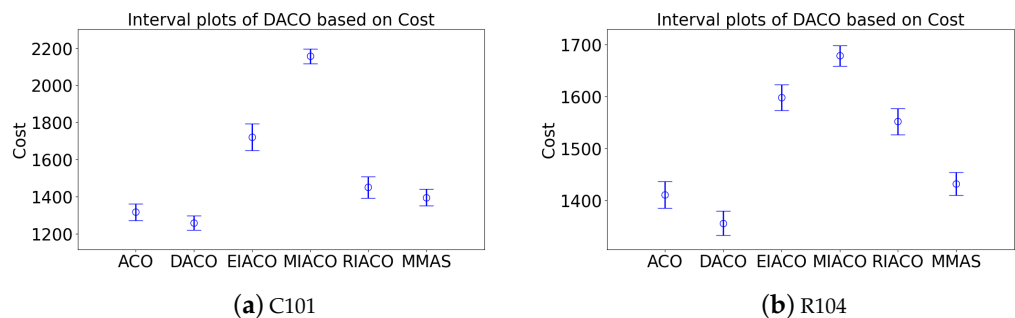


Figure 5. Interval plot of compared algorithms based on cost, 95% CI for the mean.

### 5.5. Further Analysis

This section provides further analysis based on the experimental results.

#### 5.5.1. Parameter Analysis

This section estimates the influence of the main parameters, i.e.,  $\alpha$ ,  $\beta$ ,  $\rho$ , the number of ants, and the probability  $\theta$  in SA, on the performance of DACO. We have considered values for  $\alpha$  between 0.1 and 5,  $\beta$  between 1 and 7,  $\rho$  between 0.1 and 0.8, and the number of ants between 10 and 20, because, in most cases, of the studies and estimations related to this, the radius of variation taking place within such ranges is very high. At the same time, in order to evaluate the effectiveness of the probability values calculated based on Equation (13), we compared it with some fixed values, i.e.,  $\theta = 0.25, 0.5$ , and  $0.75$ . For the sought dynamic influence in the results, we changed the value of one parameter at a time while the other parameters remained constant. In the summarized parameters, Table 2 displays the default values.

First, Figure 6 shows the relative importance of  $\alpha$  concerning the mean costs. Based on Figure 6a, there is an indication that, when  $0.1 < \alpha \leq 1.5$ , the mean values are much lower than those for  $\alpha \geq 2.0$ , which indicates that, right from the start, DACO seems to work out a better result when  $0.1 < \alpha \leq 1.5$ . As shown in Figure 6b, the costs of DACO are better if  $\alpha \in [0.5, 1.5]$  than other values of  $\alpha$ . Hence, we recommend  $\alpha \in [0.5, 1.5]$  for DACO.

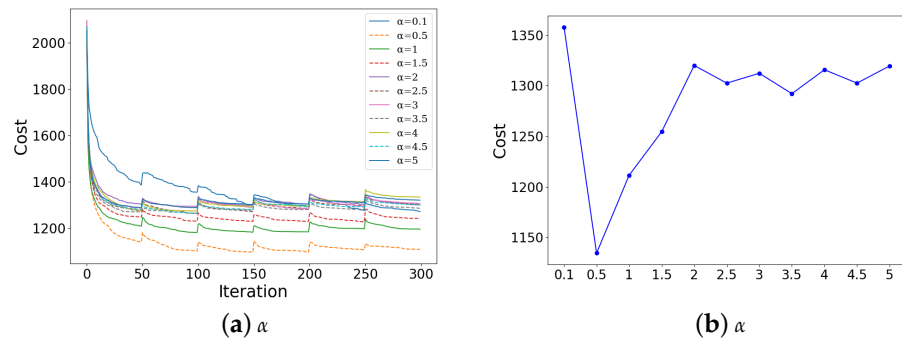


Figure 6. Parameter analysis of the relative importance of the pheromone trail  $\alpha$ .

Second, the relative importance of  $\beta$  is observed in Figure 7. As shown in Figure 7a, for  $\beta \geq 2.5$ , the mean costs reached in the first steps through the application of the initial treatments were lower compared to  $\beta \leq 2.0$ . That is, in the case of the first steps with the application of the initial treatments, it is most probable that  $\beta \geq 2.5$  will verify the best solutions of DACO compared to other values. Of greater importance, for  $\beta \geq 2.0$ , the convergence speed of the mean value is rather sufficient compared to  $\beta = 1.0$  and  $\beta = 1.5$ . Figure 7b shows that, when  $\beta \geq 2.5$ , the mean cost is very stable. Thus, we suggest, as for DACO, using  $\beta \in [2.5, 6.5]$  from the two figures.

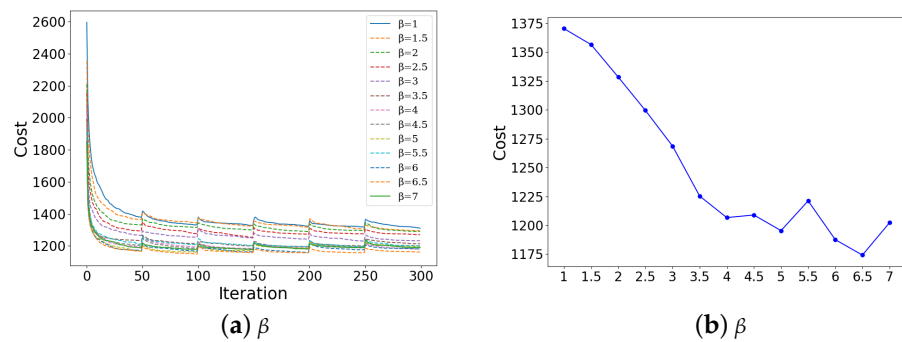


Figure 7. Parameter analysis of the relative importance of the heuristic information  $\beta$ .

Third, Figure 8 shows the effects of  $\rho$ 's prior sensitivity on the mean cost. In Figure 8a, the mean value almost shows insensitivity toward the speed of its convergence related to  $\rho$ . Figure 8b gives the dynamic variation of the mean about  $\rho$ . From the drawn graph, the mean is observed to attain a minimal value at its minimum when  $\rho = 0.2$ . It is relatively optimum in this mean value that  $\rho$  varies in the region of  $[0.1, 0.2]$ . This, therefore, means that  $\rho$  is recommended to be in this region for effective implementation of DACO.

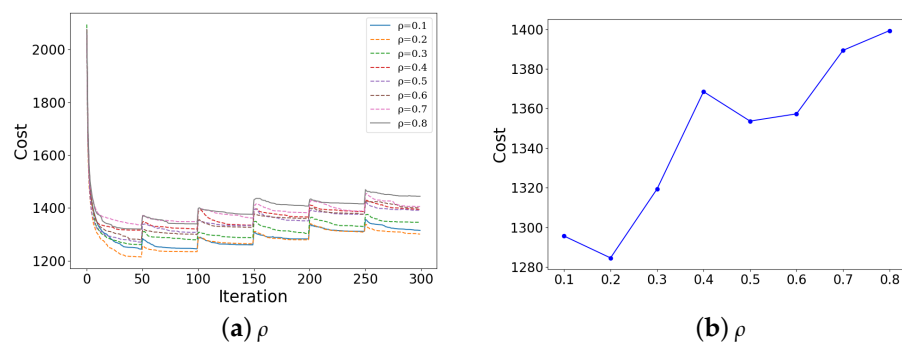


Figure 8. Parameter analysis of the pheromone evaporation rate  $\rho$ .

Fourth, Figure 9 indicates the relative importance of the number of ants concerning the mean cost. Based on Figure 9a, there is an indication that, when the number of ants is between 10 and 20, the mean value is relatively low, suggesting that the algorithm performs better with a higher number of ants. As shown in Figure 9b, the results across different number of ants values demonstrate that the mean value tends to stabilize as the number of ants increases. Therefore, it is recommended to use a higher number of ants for DACO for improved performance.

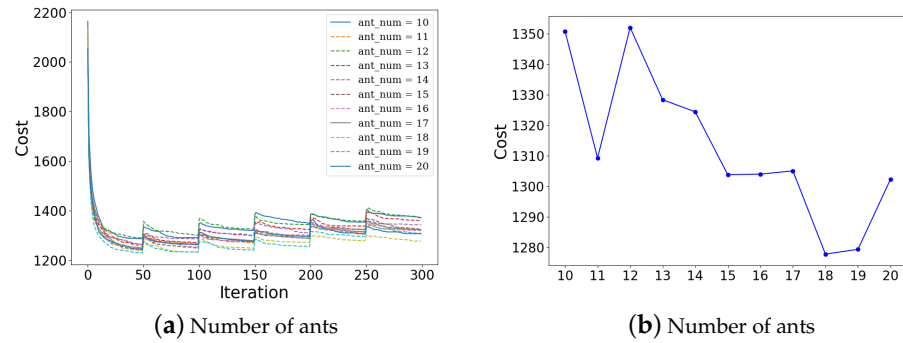


Figure 9. Parameter analysis of the number of ants.

Finally, the relative importance of  $\theta$  in the mean cost is observed in Figure 10. It can be seen that the results using Equation (13) outperform other fixed  $\theta$  values in all cases. This indicates that using Equation (13) can significantly enhance the performance of the algorithm, providing lower mean costs and better convergence. Therefore, it is recommended to use Equation (13) for parameter setting for DACO.

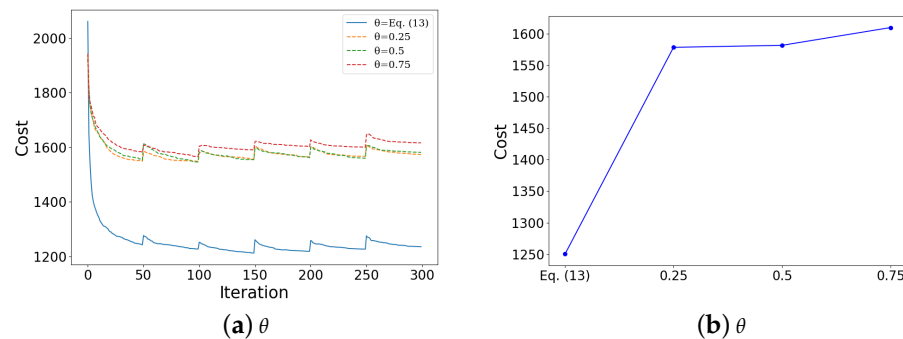
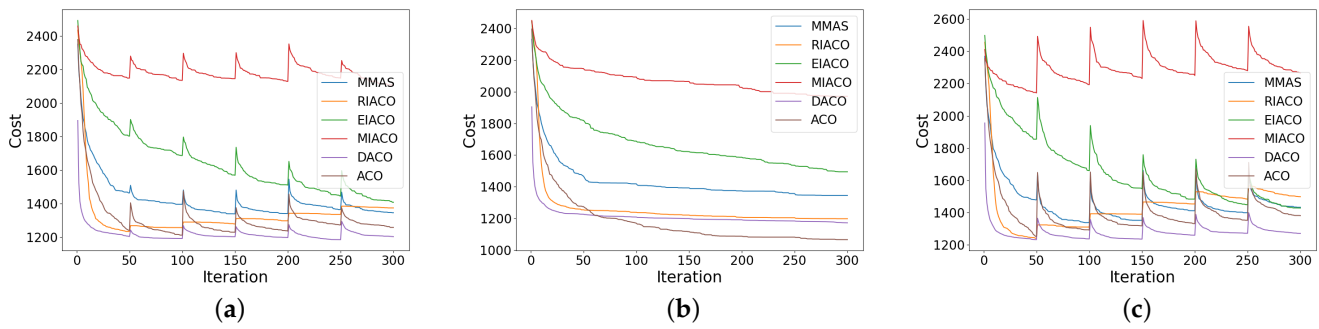


Figure 10. Parameter analysis of the probability  $\theta$  for SA.

### 5.5.2. Convergence Curve

As for the performance of DACO, Figure 11 shows the experimental results of triggering random events every 50 iterations. Specifically, in Figure 11c, the dynamic changes are limited to the appearance of new customers. It can be seen that when new customers appear, the costs have a rapid increase at first, and as the evolution progresses, the cost starts to decline. In Figure 11b, the dynamic changes are limited to customer cancellations, leading to a continuous decline in costs. In Figure 11a, the dynamic events are random, i.e., it includes scenarios where new customers appear and existing customers disappear.



**Figure 11.** The convergence curves of the algorithms on the C101 instance, where the x-axis represents the number of iterations and the y-axis represents costs. The dynamic changes are triggered every 50 iterations. (a) indicates that dynamic events are randomly triggered; (b) indicates that the dynamic changes are limited to customer cancellations; (c) indicates that the dynamic changes are limited to the appearance of new customers.

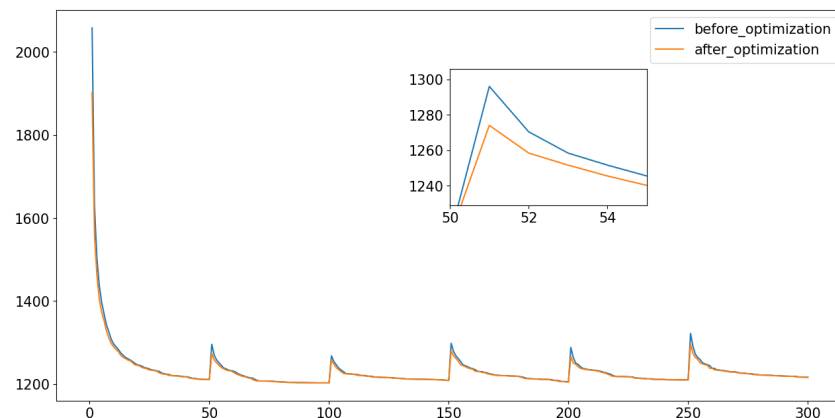
From Figure 11, we have the following observations:

- From Figure 11a, it can be seen that the convergence of DACO is faster than other algorithms, and the fluctuations when dynamic changes are triggered are smaller in magnitude compared to other algorithms.
- From Figure 11b, it can be seen that ACO has the best performance. However, the performance of DACO is very stable, that is the algorithm has strong stability.
- From Figure 11c, it can be seen that, when the demand increase is triggered, the fluctuations of the compared algorithms are greater than that of DACO. For example, ACO changes greatly after each time the demands change, and the convergence process is not as good as DACO, while DACO shows strong stability.

In general, DACO has stronger stability and convergence than the other algorithms and can quickly determine feasible solutions.

### 5.5.3. Effectiveness of the Local Search Operator

Figure 12 illustrates a clear improvement in algorithm performance resulting from the application of the local operator. In Figure 12, the blue line represents the path length without the local search, while the yellow line represents the path length with the local search. It is evident that, as the optimization operations progress, the path length exhibits a consistent decrease over time.



**Figure 12.** The compared results between the algorithm with (blue line) and without (yellow line) the local operator. The blue line segment indicates the results without the local search, and the yellow line segment indicates the result after optimization.

In terms of specific numerical values, we observed that, in the initial stages, the total path length was relatively long, resulting in higher path overhead. However, once we commenced the local optimization operations, the path length rapidly diminished, indicating an effective reduction in path cost. This improvement is crucial for enhancing resource utilization efficiency and expediting task execution.

This outcome demonstrates that, through our optimization operations, we have successfully fine-tuned route planning, making it more efficient. Such improvements have significant practical implications for addressing similar route planning problems.

#### 5.5.4. Illustration of Routes

Figure 13 illustrates the specific routes generated by the algorithms on C101 with the dynamic frequency of 50. It can be observed that MIACO exhibits the longest path length among the algorithms considered, utilizing a total of 28 vehicles. During the dynamic planning process, MIACO encounters constraint violation due to its heavy reliance on long-term memory. As a consequence, vehicles are unable to serve additional customers. RIACO, as shown in the graph, introduces the dynamism improvement by incorporating random migration, resulting in the generation of more diverse routes. While ACO produces a shorter path length compared to the other algorithms, it leads to overlapping vehicle services, contributing to higher complexity. In contrast, EIACO may have a longer path length than ACO, but it presents more straightforward and clearer routes. MMAS, although consistent with ACO in terms of paths, produces overly complex and longer routes. Among these algorithm, DACO demonstrates the lowest path cost, and it can be seen from the displayed path planning diagram that the use of fewer vehicles reduces the path cost.

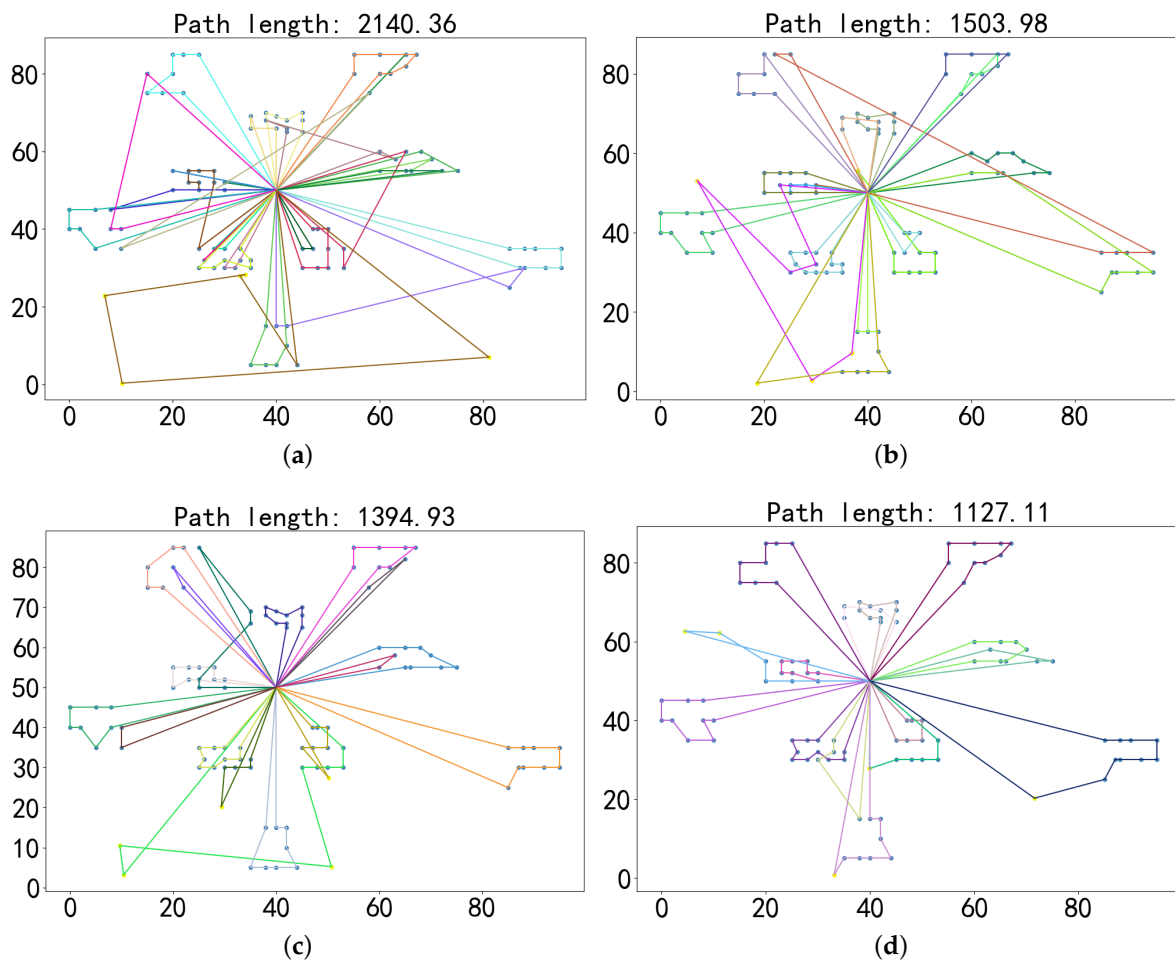
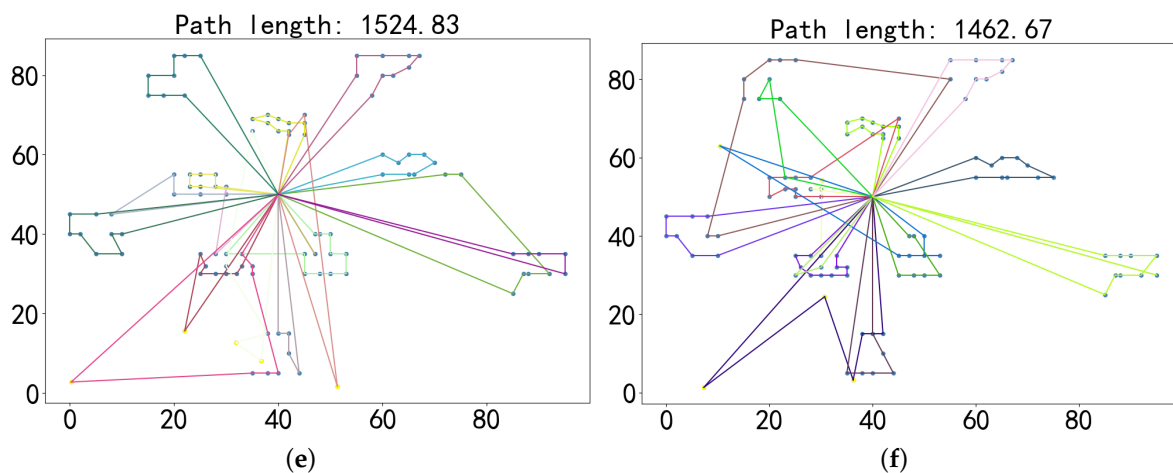


Figure 13. Cont.



**Figure 13.** The specific routes generated by the compared algorithms on C101 with the dynamic frequency of 50. (a) MICA0, (b) EIACO, (c) RIACO, (d) DACO, (e) ACO, and (f) MMAS.

## 6. Conclusions

This study preliminarily explores the application of DACO in maritime search and rescue, modeling the problem as a DVRPTW with randomly changing rescue targets. The results obtained from the experimental evaluation revealed several key findings. Firstly, DACO consistently outperformed five existing algorithms in terms of route quality, generating superior routes with minimal total costs across various dynamic scenarios. Additionally, the convergence curves demonstrated that DACO achieved faster convergence compared to the other algorithms, indicating its efficiency in finding optimal solutions within a shorter computational time. Furthermore, DACO exhibited greater stability in response to dynamic changes, highlighting its robustness in adapting to evolving environmental conditions. This is particularly evident in its ability to maintain high-quality routes even when faced with fluctuations in catastrophic situations. These findings underscore the effectiveness and superiority of DACO in addressing the DVRPTW.

In the future, research will focus on developing an online dynamic version to enhance the real-time optimization capability of DACO. This further upgrade will ensure that DACO responds to environmental changes as they occur, thus better handling emergencies and addressing the immediate needs of marine disaster events. Additionally, applying DACO to bi-layer maritime search and rescue, such as when the area of the sea that needs to be searched is quite large, requires the deployment of vessels first and then UAVs departing from the vessels to conduct localized searches, will enhance more complex and dynamic systems, initiating further research and development. Integrating DACO with other optimization techniques, such as machine learning methods, will increase the adaptability and performance of the algorithms.

**Author Contributions:** Methodology, Y.L. and Z.W.; validation, Z.W.; Formal analysis, Y.L.; writing—original draft, Z.W.; writing—review and editing, Y.L. and J.L.; supervision, Y.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the National Key Research and Development Program of China (No. 2021YFC2801001) and the Shanghai Sailing Program (No. 21YF1416700).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data are contained within the article.

**Conflicts of Interest:** The authors declare no conflicts of interest.



## References

- Oil Tanker Spill Statistics 2023. Available online: <https://www.itopf.org/knowledge-resources/data-statistics/statistics/> (accessed on 6 July 2024).
- Wu, J.; Cheng, L.; Chu, S.; Song, Y. An autonomous coverage path planning algorithm for maritime search and rescue of persons-in-water based on deep reinforcement learning. *Ocean. Eng.* **2024**, *291*, 116403. [[CrossRef](#)]
- Ma, Y.; Li, B.; Huang, W.; Fan, Q. An Improved NSGA-II based on multi-task optimization for Multi-UAV maritime search and rescue under severe weather. *J. Mar. Sci. Eng.* **2023**, *11*, 781. [[CrossRef](#)]
- Cho, S.W.; Park, H.J.; Lee, H.; Shim, D.H.; Kim, S. Coverage path planning for multiple unmanned aerial vehicles in maritime search and rescue operations. *Comput. Ind. Eng.* **2021**, *161*, 107612. [[CrossRef](#)]
- Ho, W.C.; Shen, J.H.; Liu, C.P.; Chen, Y.W. Research on optimal model of maritime search and rescue route for rescue of multiple distress targets. *J. Mar. Sci. Eng.* **2022**, *10*, 460. [[CrossRef](#)]
- Skinderowicz, R. Improving Ant Colony Optimization efficiency for solving large TSP instances. *Appl. Soft Comput.* **2022**, *120*, 108653. [[CrossRef](#)]
- Konstantakopoulos, G.D.; Gayialis, S.P.; Kechagias, E.P. Vehicle routing problem and related algorithms for logistics distribution: a literature review and classification. *Oper. Res. Int. J.* **2022**, *22*, 2033–2062. [[CrossRef](#)]
- Dantzig, G.B.; Ramser, J.H. The Truck Dispatching Problem. *Manag. Sci.* **1959**, *6*, 80–91. [[CrossRef](#)]
- Liu, Y.; Qin, Z.; Liu, J. An Improved Genetic Algorithm for the Granularity-Based Split Vehicle Routing Problem with Simultaneous Delivery and Pickup. *Mathematics* **2023**, *11*, 3328. [[CrossRef](#)]
- Torres, F.; Gendreau, M.; Rei, W. Crowdshipping: An open VRP variant with stochastic destinations. *Transp. Res. Part Emerg. Technol.* **2022**, *140*, 103677. [[CrossRef](#)]
- Solomon, M.M. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Oper. Res.* **1987**, *35*, 254–265. [[CrossRef](#)]
- Zhou, Z.; Ma, X.; Liang, Z.; Zhu, Z. Multi-objective multi-factorial memetic algorithm based on bone route and large neighborhood local search for VRPTW. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, UK, 19–24 July 2020; pp. 1–8.
- Liu, Y.; Wang, S.; Li, X. A New Cooperative Recourse Strategy for Emergency Material Allocation in Uncertain Environments. *Front. Phys.* **2022**, *10*, 835412. [[CrossRef](#)]
- Liu, Y.; Wang, J.; Zhao, J.; Li, X. Route Stability in the Uncertain Capacitated Arc Routing Problem. *Front. Energy Res.* **2022**, *10*, 933705. [[CrossRef](#)]
- Botros, A.; Gilhuly, B.; Wilde, N.; Sadeghi, A.; Alonso-Mora, J.; Smith, S.L. Optimizing Task Waiting Times in Dynamic Vehicle Routing. *IEEE Robot. Autom. Lett.* **2023**, *8*, 5520–5527. [[CrossRef](#)]
- Mohammadi, M.; Rahmanifar, G.; Hajiaghahi-Keshteli, M.; Fusc, G.; Colombaroni, C.; Sherafat, A. A dynamic approach for the multi-compartment vehicle routing problem in waste management. *Renew. Sustain. Energy Rev.* **2023**, *184*, 113526. [[CrossRef](#)]
- Zhang, J.; Teixeira, Á.P.; Soares, C.G.; Yan, X. Probabilistic modelling of the drifting trajectory of an object under the effect of wind and current for maritime search and rescue. *Ocean. Eng.* **2017**, *129*, 253–264. [[CrossRef](#)]
- Sarbijan, M.S.; Behnamian, J. A mathematical model and metaheuristic approach to solve the real-time feeder vehicle routing problem. *Comput. Ind. Eng.* **2023**, *185*, 109684. [[CrossRef](#)]
- Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [[CrossRef](#)]
- Su, Y.; Liu, J.; Xiang, X.; Zhang, X. A responsive ant colony optimization for large-scale dynamic vehicle routing problems via pheromone diversity enhancement. *Complex Intell. Syst.* **2021**, *7*, 2543–2558. [[CrossRef](#)]
- Xiang, X. Demand coverage diversity based ant colony optimization for dynamic vehicle routing problems. *Eng. Appl. Artif. Intell.* **2020**, *91*, 103582. [[CrossRef](#)]
- Xiang, X.; Tian, Y.; Zhang, X.; Xiao, J.; Jin, Y. A pairwise proximity learning-based ant colony algorithm for dynamic vehicle routing problems. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 5275–5286. [[CrossRef](#)]
- Shi, L.; Zhan, Z.; Liang, D.; Zhang, J. Memory-Based Ant Colony System Approach for Multi-Source Data Associated Dynamic Electric Vehicle Dispatch Optimization. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 17491–17505. [[CrossRef](#)]
- Euchi, J.; Yassine, A.; Chabchoub, H. The dynamic vehicle routing problem: Solution with hybrid metaheuristic approach. *Swarm Evol. Comput.* **2015**, *21*, 41–53. [[CrossRef](#)]
- Mavrovouniotis, M.; Yang, S. Ant algorithms with immigrants schemes for the dynamic vehicle routing problem. *Inf. Sci.* **2015**, *294*, 456–477. [[CrossRef](#)]
- Mavrovouniotis, M.; Yang, S.; Van, M.; Li, C.; Polycarpou, M. Ant Colony Optimization Algorithms for Dynamic Optimization: A Case Study of the Dynamic Travelling Salesperson Problem [Research Frontier]. *IEEE Comput. Intell. Mag.* **2020**, *15*, 52–63. [[CrossRef](#)]
- Xiang, X.; Tian, Y.; Cheng, R.; Zhang, X.; Yang, S.; Jin, Y. A benchmark generator for online dynamic single-objective and multi-objective optimization problems. *Inf. Sci.* **2022**, *613*, 591–608. [[CrossRef](#)]
- Pillac, V.; Gendreau, M.; Guéret, C.; Medaglia, A.L. A review of dynamic vehicle routing problems. *Eur. J. Oper. Res.* **2013**, *225*, 1–11. [[CrossRef](#)]
- Zhang, J.; Van Woensel, T. Dynamic vehicle routing with random requests: A literature review. *Int. J. Prod. Econ.* **2023**, *256*, 108751. [[CrossRef](#)]

30. Lund, K.; Madsen, O.B.; Rygaard, J.M. *Vehicle Routing Problems with Varying Degrees of Dynamism*; IMM, Institute of Mathematical Modelling, Technical University of Denmark: Lyngby, Denmark, 1996.
31. Clarke, G.; Wright, J.W. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Oper. Res.* **1964**, *12*, 568–581. [[CrossRef](#)]
32. Laporte, G. The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms. *Eur. J. Oper. Res.* **1992**, *59*, 345–358. [[CrossRef](#)]
33. Barbarosoglu, G.; Ozgur, D. A tabu search algorithm for the vehicle routing problem. *Comput. Oper. Res.* **1999**, *26*, 255–270. [[CrossRef](#)]
34. Baker, B.M.; Ayechew, M.A. A genetic algorithm for the vehicle routing problem. *Comput. Oper. Res.* **2003**, *30*, 787–800. [[CrossRef](#)]
35. Wu, H.; Gao, Y. An ant colony optimization based on local search for the vehicle routing problem with simultaneous pickup–delivery and time window. *Appl. Soft Comput.* **2023**, *139*, 110203. [[CrossRef](#)]
36. Souza, I.P.; Boeres, M.C.S.; Moraes, R.E.N. A robust algorithm based on Differential Evolution with local search for the Capacitated Vehicle Routing Problem. *Swarm Evol. Comput.* **2023**, *77*, 101245. [[CrossRef](#)]
37. Vincent, F.Y.; Anh, P.T.; Gunawan, A. A simulated annealing with variable neighborhood descent approach for the heterogeneous fleet vehicle routing problem with multiple forward/reverse cross-docks. *Expert Syst. Appl.* **2024**, *237*, 121631.
38. Frey, C.M.M.; Jungwirth, A.; Frey, M.; Kolisch, R. The vehicle routing problem with time windows and flexible delivery locations. *Eur. J. Oper. Res.* **2023**, *308*, 1142–1159. [[CrossRef](#)]
39. Ahmed, Z.H.; Yousefikhoshbakht, M. An improved tabu search algorithm for solving heterogeneous fixed fleet open vehicle routing problem with time windows. *Alex. Eng. J.* **2023**, *64*, 349–363. [[CrossRef](#)]
40. Wang, Y.; Wei, Y.; Wang, X.; Wang, Z.; Wang, H. A clustering-based extended genetic algorithm for the multidepot vehicle routing problem with time windows and three-dimensional loading constraints. *Appl. Soft Comput.* **2023**, *133*, 109922. [[CrossRef](#)]
41. Lehmann, J.; Winkenbach, M. A matheuristic for the Two-Echelon Multi-Trip Vehicle Routing Problem with mixed pickup and delivery demand and time windows. *Transp. Res. Part C Emerg. Technol.* **2024**, *160*, 104522. [[CrossRef](#)]
42. Yu, V.F.; Jodiawan, P.; Lin, S.W.; Nadira, W.F.; Asih, A.M.S.; Vinh, L.N.H. Using Simulated Annealing to Solve the Multi-Depot Waste Collection Vehicle Routing Problem with Time Window and Self-Delivery Option. *Mathematics* **2024**, *12*, 501. [[CrossRef](#)]
43. Cavecchia, M.; Alves de Queiroz, T.; Iori, M.; Lancellotti, R.; Zucchi, G. An Optimization-Based Decision Support System for Multi-trip Vehicle Routing Problems. *Comput. Sci.* **2024**, *5*, 225. [[CrossRef](#)]
44. Lee, E.H.; Jeong, J. Accessible taxi routing strategy based on travel behavior of people with disabilities incorporating vehicle routing problem and Gaussian mixture model. *Travel Behav. Soc.* **2024**, *34*, 100687. [[CrossRef](#)]
45. Luo, T.; Heng, Y.; Xing, L.; Ren, T.; Li, Q.; Qin, H.; Hou, Y.; Wang, K. A Two-Stage Approach for Electric Vehicle Routing Problem with Time Windows and Heterogeneous Recharging Stations. *Tsinghua Sci. Technol.* **2024**, *29*, 1300–1322. [[CrossRef](#)]
46. Ghannadpour, S.F.; Noori, S.; Tavakkoli-Moghaddam, R.; Ghoseiri, K. A multi-objective dynamic vehicle routing problem with fuzzy time windows: Model, solution and application. *Appl. Soft Comput.* **2014**, *14*, 504–527. [[CrossRef](#)]
47. Gholami-Zanjani, S.M.; Jafari-Marandi, R.; Pishvae, M.S.; Klibi, W. Dynamic vehicle routing problem with cooperative strategy in disaster relief. *Int. J. Shipp. Transp. Logist.* **2019**, *11*, 455–475. [[CrossRef](#)]
48. Kucharska, E. Dynamic vehicle routing problem—Predictive and unexpected customer availability. *Symmetry* **2019**, *11*, 546. [[CrossRef](#)]
49. Zacharia, P.; Drosos, C.; Piromalis, D.; Papoutsidakis, M. The vehicle routing problem with fuzzy payloads considering fuel consumption. *Appl. Artif. Intell.* **2021**, *35*, 1755–1776. [[CrossRef](#)]
50. da Silva Junior, O.S.; Leal, J.E.; Reimann, M. A multiple ant colony system with random variable neighborhood descent for the dynamic vehicle routing problem with time windows. *Soft Comput.* **2021**, *25*, 2935–2948. [[CrossRef](#)]
51. Zajkani, M.A.; Baghdorani, R.R.; Haeri, M. Model predictive based approach to solve DVRP with traffic congestion. *IFAC-Papersonline* **2021**, *54*, 163–167. [[CrossRef](#)]
52. Sabar, N.R.; Goh, S.L.; Turkey, A.; Kendall, G. Population-based iterated local search approach for dynamic vehicle routing problems. *IEEE Trans. Autom. Sci. Eng.* **2021**, *19*, 2933–2943. [[CrossRef](#)]
53. Xu, X.; Lin, Z.; Zhu, J. DVRP with limited supply and variable neighborhood region in refined oil distribution. *Ann. Oper. Res.* **2022**, *309*, 663–687. [[CrossRef](#)]
54. Zhang, J.; Luo, K.; Florio, A.M.; Van Woensel, T. Solving large-scale dynamic vehicle routing problems with stochastic requests. *Eur. J. Oper. Res.* **2023**, *306*, 596–614. [[CrossRef](#)]
55. Pan, W.; Liu, S.Q. Deep reinforcement learning for the dynamic and uncertain vehicle routing problem. *Appl. Intell.* **2023**, *53*, 405–422. [[CrossRef](#)]
56. Kim, G. Dynamic Vehicle Routing Problem with Fuzzy Customer Response. *Sustainability* **2023**, *15*, 4376. [[CrossRef](#)]
57. Aarts, E.H.L.; Van Laarhoven, P.J.M. Simulated annealing: An introduction. *Stat. Neerl.* **1989**, *43*, 31–52. [[CrossRef](#)]
58. Chiang, C.W.; Lee, W.P.; Heh, J.S. A 2-Opt based differential evolution for global optimization. *Appl. Soft Comput.* **2010**, *10*, 1200–1207. [[CrossRef](#)]
59. Mavrovouniotis, M.; Yang, S.; Yao, X. A benchmark generator for dynamic permutation-encoded problems. In Proceedings of the 12th International Conference on Parallel Problem Solving from Nature, LNCS, Taormina, Italy, 1–5 September 2012; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7492, pp. 508–517.

60. Stützle, T.; Hoos, H. MAX-MIN Ant System and local search for the traveling salesman problem. In Proceedings of the 1997 IEEE International Conference on Evolutionary Computation (ICEC), Indianapolis, IN, USA, 13–16 April 1997; pp. 309–314.
61. Stützle, T.; Dorigo, M. ACO algorithms for the traveling salesman problem. *Evol. Algorithms Eng. Comput. Sci.* **1999**, *4*, 163–183.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.