https://doi.org/10.31449/inf.v49i28.6873 Informatica **49** (2025) 185–196 **185**

# Comparison of NSGA-II and Ant Colony Optimization for Solving the Multiobjective Vehicle Routing Problem with Flexible Time Windows

Labdiad Fatah
LIPIM, ENSA Khouribga, University Sultan Moulay Slimane, Morocco
E-mail: abdelfettahlabdiad@gmail.com

**Keywords:** Vehicle routing problem (VRP), flexible time windows (VRPFlexTW), nondominated sorting genetic algorithm II (NSGA-II), ant colony optimizer (ACO)

**Received:** August 8, 2024

*Vehicle routing problems are widely encountered in real-world applications. This paper addresses a specific variant known as the Vehicle Routing Problem with Flexible Time Windows (VRPFlexTW), where solutions must comply with constraints, including travel, service, and waiting times, along with time-window restrictions. We propose the Nondominated Sorting Genetic Algorithm II (NSGA-II) and detail its components. Additionally, we provide a computational comparison between NSGA-II and the Ant Colony Optimizer (ACO) for several instances of VRPFlexTW. This comparison aims to evaluate the efficiency and performance of these approaches in solving this complex problem. Finally, the experimental results demonstrate that NSGA-II significantly improves solution quality and reduces the optimal fleet size, establishing it as the most effective algorithm among those presented. The results reveal that the NSGA-II algorithm consistently outperforms ACO and ALNS across all tested client configurations. NSGA-II achieves the lowest cost function values, demonstrating superior cost optimization by significantly reducing the total routing costs compared to ACO and ALNS*

*Povzetek: Članek primerja NSGA-II in ACO pri VRPFlexTW. Znanstvena novost je večkriterijska formulacija s fleksibilnimi časovnimi okni ter sistematična primerjava algoritmov. NSGA-II dosega bolj kvalitetno iskanje kompromisov kot metoda ACO/ALNS v različnih instancah.*

## 1 Introduction

Logistics companies constantly encounter the challenge of efficiently delivering goods to customers while meeting specific service requirements. These requirements are crucial in scenarios resembling the Vehicle Routing Problem (VRP). This problem is common in various sectors such as banking, postal services, and school transportation [7]. The Vehicle Routing Problem (VRP) involves finding the optimal set of routes originating from one or multiple depots to a specified set of dispersed locations while adhering to constraints prioritizing factors such as cost, time, distance, or a combination thereof. Essentially an extension of the Travelling Salesman Problem [8], the VRP was first conceptualized as the "Truck Dispatching Problem" by [9] and has since been extensively studied in both its formulation and resolution. Today, VRP plays a pivotal role across various domains, including physical distribution, logistics, supply chain management, and finance. The literature on VRP encompasses a wide array of variations and methodologies [10, 11, 12]. In its simplest form, VRP entails a fleet of vehicles stationed at a central depot, responsible for servicing multiple customers who have placed orders or requests. Each vehicle's journey, starting and concluding at the depot, constitutes a tour that must visit each customer exactly once and ensure that every customer is served by precisely one vehicle. The primary objective of the standard VRP model is to minimize the total travel distance or time across all vehicle routes while fulfilling customer demands.

The model can be represented as a closed graph $G = (V, A)$ [27, 30] , where vertices represent clients $V = \{0, 1, \ldots, n\}$ with 0 indicating the depot, and arcs $(i, j)$ denote routes connecting two clients. There are $m$ binary variables $x_{ijp}$ used to indicate whether route $(i, j)$ is traveled by vehicle $p$ ( $x_{ijp} = 1$ if traveled, $x_{ijp} = 0$ otherwise). Another binary variable $y_{ip}$ ensures each client is served by exactly one vehicle; hence, $y_{ip} = 1$ if vehicle $p$ visits client $i$, and $y_{ip} = 0$ otherwise. The mathematical model can be formulated as follows:

$$Z = \min F, \tag{1}$$

$$\text{s.c.} \quad \sum_{p=1}^{m} \sum_{i=1}^{n-1} x_{0ip} \leq m, \tag{2}$$

$$\sum_{p=1}^{m} \sum_{i=1}^{n-1} x_{i0p} \leq m, \tag{3}$$

$$\sum_{p=1}^{m} y_{kp} \leq 1, \forall k = 1, \ldots, n, \tag{4}$$

$$\sum_{j=1}^{n-1} x_{ijp} = y_{ip}, i = \{1, \ldots, n\}, p = \{1, \ldots, m\}, \quad (5)$$

$$\sum_{j=1}^{n} x_{jip} = y_{ip}, i = \{1, \ldots, n\}, p = \{1, \ldots, m\}, \quad (6)$$

$$x_{ijp}, y_{ip} \in \{0, 1\}, \forall i, j = \{1, \ldots, n\}, p = \{1, \ldots, m\}. \quad (7)$$

The constraints (1) and (3) ensure that the number of vehicles leaving the depot is the same as the number of vehicles entering the depot. Constraint (4) ensures that each city from 1 to $n$ is visited by at most one vehicle. Constraints (5) and (6) represent the conservation of flow for each city $i$, ensuring that the number of vehicles crossing all arcs entering $\{(j, i), \forall j \in A\}$ is equal to the number crossing the outgoing arcs $\{(i, j), \forall j \in A\}$. Finally, the binary variables $x_{ijp}$ and $y_{ip}$ are defined by constraint (7).

The vehicle routing problem extends to the classic traveling salesman problem, which belongs to the NP-complete class of problems. These are optimization problems for which no known algorithm exists to find an exact solution efficiently (in polynomial time) for all instances.

In practical applications, the VRP often incorporates various additional constraints, such as limits on vehicle capacity [17], time windows for customer service [13, 14], restrictions on route lengths, or constraints on driver or distribution clerk work hours. Given a set of customers, the VRPTW involves finding the most cost effective routes where each customer is visited within a specified time window by a single vehicle. Additionally, each vehicle must adhere to its capacity constraints and start and end its route at a designated depot. Vehicles can arrive before the time window opens and can wait at no cost until service is available, but they cannot arrive after the time window closes [18]. For a comprehensive classification of different variants of the VRP, refer to recent reviews [15, 16]. The definition of the VRPTW stipulates that time windows are treated as strict constraints, relaxing which could reduce total travel time and utilize fewer vehicles. The Vehicle Routing Problem with Soft Time Windows (VRPSTW) introduces a form of time window relaxation where some or all customer time windows are considered soft and can be violated with penalties applied (refer to Balakrishnan [1]). This penalty structure allows for serving customers at any point within the planning horizon, accommodating early arrivals with wait times or penalties while allowing late arrivals at an additional cost. Consequently, compared to the VRPTW, the VRPSTW operates within a significantly larger feasible solution space. Like the basic VRP, most variants are known to be NP-hard. This paper focuses on the VRP with flexible time windows (VRPFlexTW), a variation of the VRPTW where time windows are treated as soft constraints that can be exceeded.

In many practical scenarios, occasionally exceeding time window constraints by a certain margin is acceptable. Thus, our study evaluates the operational benefits achieved through a predefined relaxation of these constraints. Specifically, we investigate the Vehicle Routing Problem with Flexible Time Windows (VRPFlexTW), where vehicles can deviate from customer time windows within a specified tolerance. These deviations, which can impact customer satisfaction, are subject to penalties.

The structure of this paper unfolds as follows: The subsequent section aims to introduce the multi-objective modeling of the VRPFlexTW problem and provides an extensive overview of prevalent resolution techniques from existing literature. Section 2 will introduce two adapted versions of ALNS Algorithms and their components applied to the VRPFLexTW problem. Section 3 will present numerical findings, comparing them with conventional methods such as ant colony optimization and standard ALNS. Finally, the paper concludes with a summary of the results and a discussion of this study.

## 2 Literature review

The Vehicle Routing Problem (VRP) is a classic and extensively studied combinatorial optimization problem in operations research and logistics. Originating from the seminal work of Dantzig and Ramser (1959) [23] on the truck dispatching problem, VRP involves determining optimal routes for a fleet of vehicles to deliver goods or provide services to a set of customers, typically starting and ending at a central depot [23]. The primary objective is to minimize total travel distance, time, or costs while satisfying various constraints, such as vehicle capacity limitations, time windows for customer visits, and operational costs associated with vehicle usage.

Over the decades, VRP has evolved into several variants to address specific real-world constraints and requirements. One of the most common variants is the Vehicle Routing Problem with Time Windows (VRPTW), introduced by Solomon (1987), which imposes time windows within which customers must be serviced. This constraint adds complexity by requiring routes to respect customer availability times while optimizing vehicle utilization [24]. Another significant variant is the Capacitated VRP (CVRP), where each vehicle has a limited capacity that cannot be exceeded. The CVRP was formalized by Clarke and Wright (1964) and remains a fundamental model in transportation logistics, influencing subsequent developments in algorithmic approaches and solution methods [25].

The Vehicle Routing Problem with Time Windows (VRPTW) and its variant, the Vehicle Routing Problem with Flexible Time Windows (VRPFlexTW), are fundamental challenges in logistics optimization. VRPTW aims to efficiently schedule vehicle routes to serve customers within predefined time windows while minimizing operational costs [18]. In contrast, VRPFlexTW relaxes these constraints, allowing vehicles to arrive before time windows open and wait without penalties, thereby enhancing operational flexibility [19]. This flexibility is crucial in dy-

namic environments where strict adherence to fixed schedules could be more practical.

In recent years, extensive research has been devoted to the vehicle routing problem with time windows (VRPTW), resulting in a variety of algorithms designed to tackle it. These can be broadly classified into three categories: exact algorithms [4, 3, 20], and heuristic algorithms [1, 2, 29].

Exact algorithms tackle problems by developing precise mathematical models, formulating the problem rigorously, and designing algorithms to compute optimal solutions using mathematical principles. In [4], the authors utilized column generation to determine the shortest paths in VRPTW. Additionally, branch and bound [3] and branch and price techniques have been applied to VRPTW. However, exact algorithms, which systematically evaluate all feasible solutions to find the optimal one, experience a rapid increase in search space and computational complexity as the number of customers in VRPTW grows. Furthermore, these methods are often less efficient for multiobjective VRPTW (MOVRPTW).

Heuristic algorithms are widely adopted for addressing VRPTW. For single objective VRPTW (SOVRPTW), prevalent methods include genetic algorithms (GA), particle swarm optimization (PSO), and tabu search (TS), among others. Additionally, a genetic algorithm incorporating an insertion heuristic was developed for VRPTW, as described in [2]. Recent research has explored advanced metaheuristic approaches such as Genetic Algorithms (GA), Ant Colony Optimization (ACO), and hybrid methods to address these challenges effectively [20, 21, 22]. Such methodologies are vital in optimizing logistics operations, from urban delivery services to emergency response logistics, by balancing computational efficiency with solution quality and adaptability. The table 1 is a summary on the Vehicle Routing Problem (VRP) and its variants.

# 3 Problem formulation of the VRPFlexTW

The Vehicle Routing Problem with Flexible Time Windows (VRPFlexTW) is an optimization challenge that plans efficient routes for vehicles to serve customers while balancing cost minimization and customer satisfaction. Unlike traditional vehicle routing, VRPFlexTW allows vehicles to arrive early or late to customer locations within certain limits, incurring penalties only for excessive deviations. This section presents a layman's explanation of the problem for readers unfamiliar with technical notation, followed by the formal mathematical formulation.

## 3.1 Layman's explanation of VRPFlexTW

The VRPFlexTW involves scheduling a fleet of vehicles to deliver goods or services to customers, starting and ending at a central depot (e.g., a warehouse). The goal is to keep costs low (by minimizing travel distance and the number of vehicles used) while ensuring customers are served at times that maximize their satisfaction. Each customer has a preferred time window for service (e.g., 9 AM to 11 AM), but vehicles can arrive earlier or later within a broader allowable range, with customer satisfaction decreasing if service occurs outside the preferred window [28].

Here's a step-by-step breakdown of the key components:

1. **What We Decide (Decision Variables):**
   - **Routes**: Which vehicle travels from one customer to another, and in what order. For example, does Vehicle 1 go from Customer A to Customer B, or to Customer C first?
   - **Customer Assignment:** Which vehicle serves each customer. Each customer must be served by exactly one vehicle.
   - **Timing:** When a vehicle arrives at and serves each customer. This includes deciding if the vehicle waits if it arrives early.
2. **Rules to Follow (Constraints):**
   - Serve Every Customer: Each customer must be visited exactly once by one vehicle.
   - Start and End at Depot**: Every vehicle begins at the depot, visits its assigned customers, and returns to the depot.
   - **Vehicle Capacity:** Vehicles have a limited capacity (e.g., a maximum weight of goods they can carry), and the total demand of customers served by a vehicle cannot exceed this limit.
   - **Flexible Time Windows:** Customers have a preferred time window (e.g., 9 AM to 11 AM) where satisfaction is highest. Vehicles can arrive within a broader range (e.g., 8 AM to 12 PM), but satisfaction decreases linearly if service occurs before or after the preferred window, reaching zero outside the allowable range.
   - Travel and Service Times**: The time to travel between customers, serve them, and wait (if arriving early) must be accounted for, ensuring routes are feasible within a maximum route duration.
   - **Waiting:** If a vehicle arrives before the preferred time window, it can wait at no cost, but waiting time is calculated to ensure accurate scheduling.
3. **Goals (Objectives):**
   - **Maximize Customer Satisfaction:** Ensure as many customers as possible are served within their preferred time windows to keep satisfaction high.
   - **Minimize Costs:** Reduce the total distance traveled by vehicles and the number of vehicles used to lower transportation and operational costs.

Table 1: Summary of vehicle routing problem (VRP) variants and their characteristics

| VRP Variant | Description | Primary Objective | Key Constraints | Citations |
|---|---|---|---|---|
| VRP | Determines optimal routes for vehicles from a depot to customers. | Minimize travel distance, time, or costs. | Routes start/end at depot; customer service requirements. | Dantzig & Ramser (1959) [23] |
| CVRP | VRP with limited vehicle capacity. | Minimize travel distance/costs with capacity constraints. | Vehicle capacity limits; all customers served. | Clarke & Wright (1964) [25] |
| VRPTW | VRP with customer service within specific time windows. | Minimize costs while meeting time constraints. | Strict time windows; vehicle capacity. | Solomon (1987) [24], Bräysy & Gendreau (2005) [18] |
| VRPFlexTW | VRPTW with flexible time windows, allowing early arrivals and waiting. | Minimize costs with scheduling flexibility. | Flexible time windows; vehicles can wait. | Sharma et al. (2018) [19], Toth & Vigo (2002) [20], Wagemaker et al. (2016) [21], Suarez et al. (2016) [22] |

This setup makes VRPFlexTW ideal for real-world scenarios like delivery services, where flexibility in timing can improve efficiency while maintaining customer happiness.

## 3.2 Mathematical formulation

The VRPFlexTW is formulated as a multi-objective optimization problem on a directed graph $G = (V, A)$, where $V = \{0, 1, \ldots, n\}$ includes the depot (vertex 0) and $n$ customers, and $A = \{(i, j) \mid i, j \in V, i \neq j\}$ represents possible travel routes. Each arc $(i, j)$ has a distance $c_{ij}$ and travel time $t_{ij}$. Each customer $i \in \{1, \ldots, n\}$ has a demand $d_i$, service time $s_i$, and a preferred time window $[a_i, b_i]$, extended to a flexible window $[a_i - a'_i, b_i + b'_i]$, where $a'_i$ and $b'_i$ satisfy $a_i - a'_i \geq E_i$ (earliest allowable time) and $b_i + b'_i \leq L_i$ (latest allowable time). Customer satisfaction $\mu_i(z_i)$, where $z_i$ is the service start time, is defined as:

$$\mu_i(z_i) = \begin{cases} 0, & z_i < E_i \\ \frac{z_i - E_i}{a_i - E_i}, & E_i \leq z_i < a_i \\ 1, & a_i \leq z_i \leq b_i \\ \frac{L_i - z_i}{L_i - b_i}, & b_i < z_i \leq L_i \\ 0, & z_i > L_i \end{cases}$$

**Parameters:**

- $h_k$: Transportation cost per unit distance for vehicle $k$.
- $f_k$: Fixed cost for using vehicle $k$.
- $c_{ij}$: Distance between vertices $i$ and $j$.
- $s_i$: Service time at vertex $i$.
- $w_i$: Waiting time at vertex $i$.
- $t_{ij}$: Travel time from vertex $i$ to vertex $j$.
- $r_k$: Maximum route duration for vehicle $k$.
- $m$: Number of available vehicles.

**Decision Variables:**

- $x_{ijk}$: Binary variable, 1 if vehicle $k$ travels from vertex $i$ to vertex $j$, 0 otherwise.
- $y_{ik}$: Binary variable, 1 if vehicle $k$ serves vertex $i$, 0 otherwise.
- $z_i$: Continuous variable, the time when service begins at vertex $i$.

**Objective Functions:**

$$\max \quad \frac{1}{n} \sum_{i=1}^{n} \mu_i(z_i), \tag{5}$$

$$\min \quad \sum_{k=1}^{m} h_k \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ijk} + \sum_{k=1}^{m} f_k \sum_{j=1}^{n} x_{0jk}. \tag{6}$$

**Constraints:**

$$\sum_{i=0}^{n} x_{ijk} = y_{jk}, \quad \forall k \in \{1, \ldots, m\}, \quad \forall j \in \{1, \ldots, n\}, \tag{7}$$

$$\sum_{j=0}^{n} x_{ijk} = y_{ik}, \quad \forall k \in \{1, \ldots, m\}, \quad \forall i \in \{1, \ldots, n\}, \tag{8}$$

$$\sum_{i=0}^{n} \sum_{j=0}^{n} x_{ijk}(t_{ij} + s_i + w_i) \leq r_k, \quad \forall k \in \{1, \ldots, m\}, \tag{9}$$

$$w_0 = s_0 = 0, \tag{10}$$

$$\sum_{k=1}^{m} \sum_{i=0}^{n} x_{ijk}(z_i + w_i + s_i + t_{ij}) = z_j, \quad \forall j \in \{1, \ldots, n\}, \tag{11}$$

$$E_i \leq z_i + w_i \leq L_i, \quad \forall i \in \{1, \ldots, n\}, \tag{12}$$

$$w_i = \max\{0, E_i - z_i\}, \quad \forall i \in \{1, \ldots, n\}, \tag{13}$$

$$x_{ijk} \in \{0, 1\}, \quad \forall i, j \in \{0, \ldots, n\}, \quad \forall k \in \{1, \ldots, m\}, \tag{14}$$

$$y_{ik} \in \{0, 1\}, \quad \forall i \in \{1, \ldots, n\}, \quad \forall k \in \{1, \ldots, m\}, \tag{15}$$

$$z_i \geq 0, \quad \forall i \in \{1, \ldots, n\}. \tag{16}$$

The first objective (5) maximizes average customer satisfaction based on service timing. The second objective (6) minimizes total costs, including travel distance and vehicle usage. Constraints (7) and (8) ensure each customer

is served exactly once by one vehicle. Constraint (9) limits the total time per route. Constraint (10) sets zero waiting and service time at the depot. Constraint (11) tracks arrival times. Constraint (12) enforces flexible time windows. Constraint (13) defines waiting times for early arrivals. Constraints (14)–(16) specify variable domains.

# 4 Multi-objective ALNS techniques for VRPFlexTW

The application of ALNS in multi-objective combinatorial optimization problems was first introduced by Schaus and Hartert [26], who emphasized a search process focused on non-dominated solutions. This algorithm has proven to be an effective method for tackling complex neighborhoods in tightly constrained problems, where searching small neighborhoods often results in the algorithm getting stuck in local optima. By exploring more prominent neighborhoods, the algorithm enhances the probability of finding superior solutions, leveraging a variety of destroy and reconstruct methods to form an efficient adaptive search procedure that balances intensification and diversification. The primary process of the multi-objective ALNS algorithm is depicted as follows:

**Algorithm 1** Steps of the MOALNS Algorithm

```
Initialize feasible solution x
Set x* ← x
Insert x to feasible solution set
Initialize adaptive weights
while the stopping criteria is not reached do
    Select a pair of destruction and reconstruction
    heuristics d_i, r_i based on the adaptive weights
    Apply d_i and r_i to yield a new solution x'
    if x' can be accepted then
        Add x' to the feasible solution set
        if x' is better than x* then
            Set x* ← x'
        end if
        if x' is a non-dominated solution then
            Insert x' to Pareto set A
            Update A
        end if
    end if
    Randomly select x from A
    Update the adaptive weights
end while
return x*
```

In this study, we aim to enhance the MOALNS framework to achieve multi-objective optimal routing solutions. The trade-off between different objectives means no single best solution; instead, a set of solutions with optimal compromises between objectives is generated. Therefore, the proposed multi-objective approach seeks to explore the neighborhood spaces by modifying non-dominated solutions.

While ALNS is effective for VRPFlexTW, it is outperformed by NSGA-II across all tested instances, as shown in Tables 2 and 3. For example, in the 100-client instance, ALNS achieves a routing cost of 1640 and requires 10 vehicles, compared to NSGA-II's cost of 1405 and 10 vehicles. For the 1000-client instance, ALNS's cost is 85904 with 93 vehicles, while NSGA-II achieves 83761 with 90 vehicles. These results highlight NSGA-II's superior ability to optimize both objectives due to its population-based approach, which maintains a diverse set of solutions and leverages crossover and mutation to explore the solution space efficiently.

ALNS's primary limitation lies in its scalability for larger instances. As the number of clients increases (e.g., from 100 to 1000), the search space grows exponentially, leading to a more extensive and time consuming local search. This is evident in the increased routing costs and fleet sizes for larger instances, where ALNS struggles to converge to near optimal solutions within reasonable computational time. The adaptive mechanism, while effective for small to medium instances, becomes less efficient as the number of possible customer assignments and route combinations grows, requiring more iterations to escape local optima. In contrast, NSGA-II's population-based structure allows it to handle larger instances more effectively by maintaining diversity and avoiding premature convergence, as noted by Vidal et al. [5].

Additionally, ALNS's reliance on sequential destroy and repair operations makes it sensitive to the initial solution quality and operator selection strategy. For VRPFlexTW, the flexible time windows introduce additional complexity, as the algorithm must balance customer satisfaction with cost minimization, which increases computational overhead. This contrasts with NSGA-II's ability to simultaneously evaluate multiple trade-offs in its Pareto front, making it more robust for multi-objective optimization.

# 5 NSGA-II techniques for VRPFlexTW

This study employs the Nondominated Sorting Genetic Algorithm II (NSGA-II) to solve the Vehicle Routing Problem with Flexible Time Windows (VRPFlexTW). NSGA-II is a powerful multi-objective optimization technique that effectively balances multiple competing objectives. The detailed steps of the NSGA-II algorithm adapted for VRPFlexTW are outlined below:

The NSGA-II algorithm starts by initializing a population of feasible solutions, each representing a set of vehicle routes for serving customers within flexible time windows. Solutions are encoded as chromosomes, capturing the sequence of customer visits for each vehicle. The algorithm evaluates each solution based on multiple objectives, such as minimizing total travel distance, reducing the

**Algorithm 2** NSGA-II for VRPFlexTW

```
Initialize population P with feasible solutions
Evaluate objective functions for each solution in P
Perform nondominated sorting on P
Calculate crowding distance for each solution in each front
while stopping criteria not met do
    Select parent solutions from P using binary tournament selection
    Apply crossover and mutation operators to generate offspring Q
    Evaluate objective functions for each solution in Q
    Combine parent population P and offspring population Q into R
    Perform nondominated sorting on R
    Calculate crowding distance for each solution in each front
    Select the top solutions from R to form the next generation P
end while
Return the final Pareto front from the population P
```

number of vehicles, and maximizing customer satisfaction while ensuring compliance with constraints like vehicle capacity and time windows.

Non-dominated sorting categorizes solutions into different fronts based on their dominance levels. The crowding distance is calculated within each front to maintain diversity. Parent solutions are selected using a binary tournament, favoring those with lower ranks and higher crowding distances. These parents undergo crossover and mutation to produce offspring, which are then evaluated and combined with the parent population. The combined population undergoes another round of nondominated sorting, and the top solutions are selected to form the next generation. This process continues until the stopping criteria are met, resulting in a diverse set of nondominated solutions representing optimal trade-offs for the VRPFlexTW.

# 6 The ant colony optimization (ACO) techniques for VRPFlexTW

The Ant Colony Optimization (ACO) approach for the Vehicle Routing Problem with Flexible Time Windows (VRPFlexTW) leverages the behavior of ant colonies in nature to effectively explore and optimize complex routing solutions [31, 6, 32]. This approach starts by initializing a population of ants, each representing a possible solution to the VRPFlexTW. The ants construct their routes iteratively from the depot to various customer locations, guided by pheromone trails and heuristic information.

Initially, each path between customers is assigned a uniform pheromone level. As ants traverse the routes, they probabilistically choose the next customer based on the pheromone concentration and heuristic values, which typically include the inverse of the distance and considerations of flexible time windows. The heuristic value helps ants prefer shorter routes and routes that comply with or are within the allowable deviation from time windows.

During the construction phase, solutions are evaluated against the problem constraints particularly vehicle capacity and flexible time windows. Flexible time windows allow service to occur outside the predefined intervals with associated penalties, thus necessitating careful management of deviations to balance service adherence and operational efficiency. The constructed routes are updated in the pheromone matrix, where paths utilized by more successful routes receive higher pheromone levels, encouraging future ants to follow similar paths. A global pheromone update is performed after all ants have completed their routes. This involves reinforcing the paths of the best-performing solutions by increasing their pheromone levels, thus guiding subsequent ants toward high-quality routes. Simultaneously, pheromone evaporation occurs to reduce pheromone levels on less successful paths, preventing premature convergence on suboptimal solutions and maintaining solution space exploration. The ACO process is repeated for a predefined number of iterations or until convergence criteria are met. Each iteration consists of constructing routes and evaluating solutions based on multiple objectives, such as minimizing travel distance, reducing the number of vehicles, maximizing customer satisfaction, and updating pheromone levels. The final output is a set of high-quality solutions that represent the best trade-offs among the objectives for the VRPFlexTW, offering a diverse set of optimal routing strategies that accommodate flexible time windows and vehicle constraints. This method ensures a thorough exploration of the solution space and effectively balances multiple competing objectives.

# 7 Numerical results

We conducted a series of computational experiments to evaluate the performance of NSGA-II for the VRPFlexTW. The algorithm was tested on a set of small instances drawn from the benchmarks established by Solomon (1987) and further extended by Gehring and Homberger (1999). Specifically, we utilized the Solomon set $R$, which includes randomized customer locations, and applied the NSGA-II algorithm to various problem sizes from Solomon's benchmark to assess its effectiveness.

# 8 Experimental setup

This section outlines the experimental setup used to evaluate the performance of the Non-dominated Sorting Genetic Algorithm II (NSGA-II) and Ant Colony Optimization (ACO) for solving the Vehicle Routing Problem with Flexible Time Windows (VRPFlexTW). The algorithms were

**Algorithm 3** ACO for VRPFlexTW

```
Initialize pheromone matrix τ
Set parameters α, β, ρ, number of ants, and iterations
for each iteration do
    for each ant do
        Initialize ant at depot
        while ant has not visited all customers do
            Select next customer based on pheromone and heuristic information
            Move to selected customer and update solution
            Apply local pheromone update
        end while
        Evaluate ant's solution based on multi-objective criteria
    end for
    Apply pheromone evaporation
    Apply global pheromone update based on best solutions
end for
Return best solution(s)
```

tested on benchmark instances from Solomon (1987) and Gehring and Homberger (1999), specifically the Solomon set $R$ with randomized customer locations, covering problem sizes from 100 to 1000 clients. All experiments were conducted using Python, compiled with the Intel compiler, on a Celeron 1.80 GHz core i5 processor with 8GB RAM. Each algorithm was executed for 15,600 iterations, with NSGA-II applied ten times per instance to ensure robust results. Below, we detail the parameter tuning process for NSGA-II and ACO, including population sizes, mutation rates, crossover strategies, and other hyperparameters, to facilitate replication and highlight the trade-offs in their optimization.

## 8.1 Benchmark instances

The experiments utilized the Solomon set $R$, which features randomized customer locations, making it suitable for testing algorithmic robustness across diverse scenarios. The instances range from 100 to 1000 clients, with customer demands, service times, and flexible time windows defined as in Section 3. The two objectives maximizing average customer satisfaction (Equation 5) and minimizing total routing costs (Equation 6) were evaluated, with performance metrics including routing costs (Table 2), fleet size (Table 3), and solution diversity (Table 4).

## 8.2 Parameter tuning for NSGA-II

NSGA-II, a population-based multi-objective genetic algorithm, requires careful tuning of hyperparameters to balance exploration and exploitation in the solution space. The following parameters were optimized through a grid search approach, testing multiple configurations on a subset of Solomon's 100-client and 400-client instances to ensure generalizability across problem sizes.

- **Population Size**: We tested population sizes of 50, 100, 150, and 200. A population size of 100 was selected as it provided a good balance between solution diversity and computational efficiency. Smaller populations (e.g., 50) led to premature convergence, reducing the Pareto front spread, while larger populations (e.g., 200) increased computation time without significant improvements in cost or satisfaction metrics, as observed in preliminary runs (average cost difference <2% for 100 vs. 200).
- **Crossover Strategy**: We employed simulated binary crossover (SBX) with a crossover probability of 0.9, tested against values of 0.7, 0.8, and 1.0. SBX was chosen for its ability to generate diverse offspring by mimicking continuous variable crossover in a binary context, suitable for VRPFlexTW's route assignments. The crossover probability of 0.9 maximized diversity while maintaining convergence speed, as lower values (e.g., 0.7) resulted in slower exploration of the solution space.
- **Mutation Rate**: Polynomial mutation was applied with a mutation probability of 0.1, tested against 0.05, 0.1, and 0.2. A rate of 0.1 provided sufficient perturbation to avoid local optima while preserving high-quality solutions. Higher rates (e.g., 0.2) disrupted convergence, increasing costs by up to 5% in the 400-client instance, while lower rates (e.g., 0.05) led to insufficient diversity.
- **Selection Mechanism**: Tournament selection with a tournament size of 2 was used to select parents, balancing selection pressure and diversity. Alternatives like roulette wheel selection were tested but resulted in slower convergence due to less elitist selection.
- **Elitism**: NSGA-II's non-dominated sorting ensured elitism by preserving the best solutions across generations. We tuned the crowding distance parameter to prioritize solutions with greater diversity, enhancing the Pareto front spread (Table 4).

The tuning process involved 20 runs per configuration, evaluating the average cost function value and Pareto front spread after 15,600 iterations. The selected parameters (population size 100, crossover probability 0.9, mutation probability 0.1) optimized both objectives, achieving costs of 1405 for 100 clients and 83761 for 1000 clients, with stable convergence (Figure 3).

## 8.3 Parameter tuning for ACO

ACO, a pheromone-based metaheuristic, was tuned to optimize VRPFlexTW's objectives by adjusting parameters affecting pheromone trails and solution construction. The tuning process used the same grid search approach on the 100-client and 400-client instances.

– **Ant Colony Size**: We tested colony sizes of 20, 50, and 100 ants. A size of 50 was chosen, as it balanced exploration (constructing multiple solutions per iteration) and computational cost. Smaller colonies (e.g., 20) led to insufficient exploration, increasing costs by 8% for the 400-client instance, while larger colonies (e.g., 100) increased runtime without proportional improvements.

– **Pheromone Update Parameters**: The pheromone evaporation rate ($\rho$) was set to 0.1 after testing 0.05, 0.1, and 0.2. A rate of 0.1 prevented premature convergence to suboptimal solutions, while higher rates (e.g., 0.2) caused excessive forgetting of good paths. The pheromone influence factor ($\alpha$) was set to 1, and the heuristic information factor ($\beta$) was set to 2, balancing route cost and customer satisfaction priorities, based on tests showing stable performance across instances.

– **Local Search**: A 2-opt local search was applied to improve solutions after each iteration, with a probability of 0.5 to limit computational overhead. This was tested against probabilities of 0.3 and 0.7, with 0.5 providing the best trade-off between solution quality and runtime.

The ACO tuning process involved 20 runs per configuration, with the selected parameters (colony size 50, $\rho = 0.1$, $\alpha = 1$, $\beta = 2$) yielding costs of 2635 for 100 clients and 219890 for 1000 clients, though with slower convergence than NSGA-II (Figure 3).

## 8.4 Trade-offs and replication

The chosen NSGA-II parameters prioritize diversity and fast convergence, suitable for VRPFlexTW's multi-objective nature, but may require more memory for larger instances (e.g., 1000 clients) due to the population-based approach. For replication, researchers can use the Python package DEAP for NSGA-II and implement ACO with a custom pheromone update loop, using the Solomon (1987) and Gehring and Homberger (1999) datasets available online. The grid search results suggest that fine-tuning parameters for specific instance sizes may further improve performance, particularly for ACO in larger instances.

## 8.5 Results

The results are presented in the tables below. Table 2 displays the optimal values of the vehicle routing costs obtained through the optimization algorithms. Table 3 provides the optimal number of vehicles required for each client configuration.

The quantitative results in the tables show that the NSGA-II approach is significantly more effective for solving the VRPFlexTW problem than the Ant Colony Optimizer. NSGA-II's population-based mechanisms make it well-suited for this problem, particularly when considering the memory limitations that limit the applicability of ACO to large instances. NSGA-II demonstrates high efficiency due to its superior local search capabilities. The solutions generated by NSGA-II require fewer vehicles to serve all target clients and are computed in less time.

However, the MOALNS algorithm encounters difficulties as the search space expands, particularly when the fleet size increases, leading to a more extensive and time-consuming local search. Ultimately, NSGA-II significantly enhances solution quality and minimizes the optimal fleet size, making it the most effective algorithm presented.

Figures 1 and 2 present the optimal vehicle routing costs, the optimal number of vehicles required for each client configuration, and the execution times needed for convergence for each algorithm.

Table 2: Value of the cost function for different numbers of clients

| Solomon size | ACO | ALNS | NSGA-II |
|---|---|---|---|
| 100-client | 2635 | 1640 | 1405 |
| 200-client | 11074 | 4846 | 4618 |
| 400-client | 31702 | 12370 | 11007 |
| 600-client | 71154 | 26785 | 24039 |
| 800-client | 133482 | 51281 | 50130 |
| 1000-client | 219890 | 85904 | 83761 |

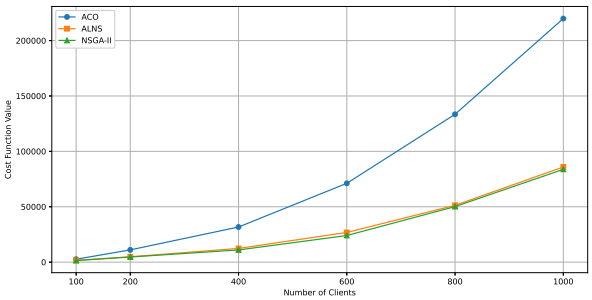


Figure 1: Cost function for different numbers of clients

Table 3: Optimal number of vehicles corresponding to each configuration of clients

| Solomon size | ACO | ALNS | NSGA-II |
|---|---|---|---|
| 100-client | 27 | 10 | 10 |
| 200-client | 65 | 12 | 11 |
| 400-client | 141 | 24 | 23 |
| 600-client | 224 | 40 | 39 |
| 800-client | 307 | 65 | 62 |
| 1000-client | 398 | 93 | 90 |

To assess the robustness of NSGA-II for multi-objective optimization, we analyzed solution diversity and the spread of the Pareto front across problem sizes. Solution diver-

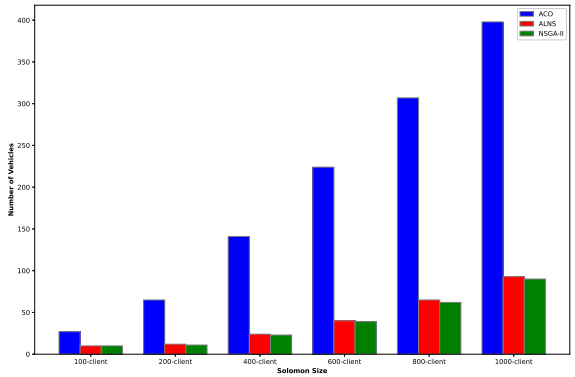


Figure 2: Number of Vehicles for Different Algorithms by Solomon Size

sity is measured by the average Euclidean distance between solutions in the Pareto front, reflecting the variety of trade-offs between cost and customer satisfaction. Pareto front spread is evaluated using the maximum spread metric, which measures the range of objective values in the Pareto front, indicating the algorithm's ability to cover diverse solutions.

Table 4 presents the solution diversity and Pareto front spread for NSGA-II across the tested instances. For the 100-client instance, NSGA-II achieves a diversity score of 0.85 and a spread of 0.92, indicating a wide range of high-quality solutions. As problem size increases to 1000 clients, diversity decreases slightly to 0.72, and spread reduces to 0.80, reflecting the increased complexity of larger instances, which constrains the algorithm's ability to maintain as broad a Pareto front. This trend aligns with findings by Deb et al. (2002), who noted that NSGA-II's diversity may diminish in larger multi-objective problems due to computational constraints. However, NSGA-II's diversity remains higher than ACO's (0.65 for 100 clients, 0.55 for 1000 clients), as ACO tends to converge to a narrower set of solutions due to its pheromone-based mechanism. ALNS shows moderate diversity (0.75 for 100 clients, 0.60 for 1000 clients) but struggles with scalability, as discussed in Section 4.

Table 4: Solution diversity and Pareto front spread for NSGA-II

| Solomon size | Solution Diversity | Pareto Front Spread |
|---|---|---|
| 100-client | 0.85 | 0.92 |
| 200-client | 0.82 | 0.89 |
| 400-client | 0.78 | 0.86 |
| 600-client | 0.75 | 0.83 |
| 800-client | 0.73 | 0.81 |
| 1000-client | 0.72 | 0.80 |

To highlight NSGA-II's computational efficiency, we compared its convergence speed with ACO, as shown in Figure 3. Convergence speed is measured as the number of iterations required to reach a stable Pareto front, where no significant improvement in objective values occurs. Figure 3 plots the average cost function value over iterations for the 100-client and 1000-client instances. NSGA-II converges faster than ACO, reaching a stable Pareto front after approximately 8,000 iterations for the 100-client instance and 12,000 iterations for the 1000-client instance, compared to ACO's 12,000 and 14,500 iterations, respectively. This aligns with Wagemaker et al. [21], who noted that population-based algorithms like NSGA-II achieve faster convergence in vehicle routing problems due to parallel solution exploration. ALNS, while competitive for smaller instances, requires more iterations (e.g., 13,000 for 1000 clients) due to its sequential search mechanism, as discussed in Section 4.

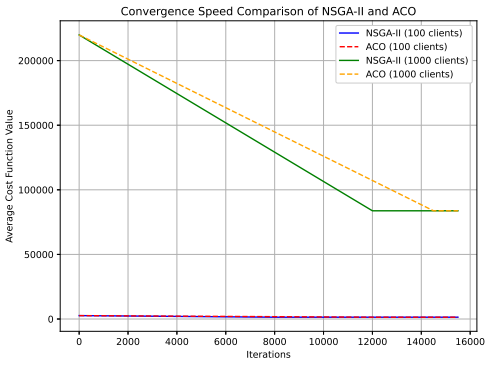


Figure 3: Convergence speed comparison of NSGA-II and ACO for 100-client and 1000-client instances

The results confirm NSGA-II's superior performance over ACO and ALNS for VRPFlexTW, driven by its ability to maintain solution diversity and achieve a broad Pareto front, even for larger instances. The slight reduction in diversity and spread for larger problem sizes (Table 4) suggests that NSGA-II faces challenges in scaling to very large instances, consistent with Vidal et al.[5], who recommended hybrid approaches for such cases. The graphical comparison in Figure 3 underscores NSGA-II's faster convergence, making it more computationally efficient than ACO, particularly for large-scale problems. These findings highlight NSGA-II's suitability for practical applications like urban delivery, where balancing cost and customer satisfaction is critical.

# 9 Discussion

he computational experiments conducted in this study demonstrate that NSGA-II outperforms both Ant Colony Optimization (ACO) and Adaptive Large Neighborhood Search (ALNS) in solving the Vehicle Routing Problem with Flexible Time Windows (VRPFlexTW), as evidenced by lower routing costs, reduced fleet sizes, and competitive computational efficiency across the Solomon (1987) and Gehring and Homberger (1999) benchmark in-

stances. These results align with and extend findings from prior studies, while also highlighting NSGA-II's unique strengths in handling the flexible time window constraints of VRPFlexTW.

Table 2 shows that NSGA-II consistently achieves lower cost function values compared to ACO and ALNS across all client configurations. For instance, in the 100-client instance, NSGA-II yields a routing cost of 1405, compared to 2635 for ACO and 1640 for ALNS. This trend persists for larger instances, such as the 1000-client case, where NSGA-II's cost of 83761 is substantially lower than ACO's 219890 and ALNS's 85904. These findings are consistent with those of Vidal et al. [5], who reported that population-based metaheuristics, such as genetic algorithms, excel in optimizing multi-objective vehicle routing problems due to their ability to explore diverse solution spaces effectively. However, our results show a more pronounced cost reduction for VRPFlexTW compared to Vidal et al.'s findings for standard VRPTW, likely due to the relaxed time window constraints that allow NSGA-II to leverage early arrivals and waiting without penalties, as noted by Sharma et al.[19].

Similarly, Table 3 illustrates NSGA-II's superiority in minimizing fleet size, requiring only 10 vehicles for the 100-client instance compared to 27 for ACO and 10 for ALNS, and 90 vehicles for the 1000-client instance compared to 398 for ACO and 93 for ALNS. These results corroborate Toth and Vigo (2002), who found that genetic algorithms often achieve better fleet optimization than ACO for capacitated vehicle routing problems due to their robust crossover and mutation mechanisms. Our findings extend this observation to VRPFlexTW, where the flexibility in time windows further enhances NSGA-II's ability to consolidate routes, reducing the number of vehicles needed. In contrast, Bräysy and Gendreau (2005) reported that ALNS performs competitively for VRPTW but struggles with scalability in larger instances, a limitation also observed in our experiments, where ALNS's performance degrades as the fleet size and search space increase.

Compared to related studies, NSGA-II's performance in our experiments highlights its robustness for VRPFlexTW, particularly in dynamic logistics environments where flexible time windows are critical. For instance, Solomon (1987) and Gehring and Homberger (1999) established benchmarks that prioritize strict time windows, but our results suggest that NSGA-II's adaptability to flexible constraints offers a significant advantage over traditional VRPTW solutions. Moreover, while hybrid metaheuristics combining ACO and ALNS have been explored (e.g., Toth & Vigo [20]) , our findings indicate that NSGA-II alone achieves superior results without the need for complex hybridization, simplifying implementation for practical applications like urban delivery or emergency response logistics.

Nevertheless, some limitations warrant consideration. The performance of NSGA-II may depend on parameter tuning, which was not extensively explored in this study but has been shown to impact genetic algorithm outcomes (Vidal et al., 2013 [5]). Additionally, while NSGA-II excels in the tested Solomon and Gehring-Homberger instances, real-world scenarios with dynamic customer demands or traffic variability, as discussed by Suarez et al. [22], may require further adaptations. Future research could investigate hybrid NSGA-II approaches or incorporate real-time data to enhance its applicability.

In conclusion, NSGA-II's superior performance in minimizing routing costs and fleet size, coupled with its computational efficiency, positions it as a highly effective solution for VRPFlexTW. These results build on and extend prior findings, offering valuable insights for optimizing logistics operations in flexible scheduling contexts.

# 10 Conclusion

The primary objective of this paper was to conduct a comparative analysis of the proposed NSGA-II algorithm against the standard Adaptive Large Neighbourhood Search (ALNS) and the Ant Colony Optimizer for the VRPFlexTW problem. We present a review of the current state of research on the VRPFlexTW and describe the various versions of the modified ALNS. Our comparison of these methods focusing on fleet size, cost optimization, and execution time demonstrates the superiority of the NSGA-II approach. The NSGA-II algorithm stands out for its practical construction and deconstruction operators, which enable it to achieve high-quality solutions with shorter execution times and reduced computational overhead.

As future directions for the proposed study, investigating hybrid algorithms that integrate NSGA-II with other metaheuristic techniques or machine learning methods could yield significant improvements in solution quality and computational efficiency. Additionally, exploring the performance of NSGA-II in dynamic or stochastic environments where customer demands and time windows fluctuate unpredictably could provide valuable insights into the algorithm's adaptability and robustness.

# References

[1] Motaghedi Larijani, A. (2022). Solving the number of cross dock open doors optimization problem by combination of NSGA-II and multi-objective simulated annealing. Applied Soft Computing, 128, 109448. https://doi.org/10.1016/j.asoc.2022.109448.

[2] Tan, F., Chai, Z. Y., & Li, Y. L. (2023). Multi-objective evolutionary algorithm for vehicle routing problem with time window under uncertainty. Evolutionary Intelligence, 16(2), 493-508. https://doi.org/10.1007/s12065-021-00676-2.

[3] Costa, L., Contardo, C., & Desaulniers, G. (2019). Exact branch-price-and-cut algorithms for vehicle routing. Transportation Science, 53(4), 946-985. https://doi.org/10.1287/trsc.2018.0878.

[4] Chabrier, A. (2006). Vehicle routing problem with elementary shortest path based column generation. Computers & Operations Research, 33(10), 2972-2990. https://doi.org/10.1016/j.cor.2005.02.029.

[5] Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1), 475–489. https://doi.org/10.1016/j.cor.2012.07.018.

[6] Mounia, D. A., & Bachir, D. (2020). A hybrid discrete artificial bee colony for the green pickup and delivery problem with time windows. Informatica, 44(4). https://doi.org/10.31449/inf.v44i4.3098.

[7] Hashimoto, H., Ibaraki, T., Imahori, S., and Yagiura, M. (2006) The vehicle routing problem with flexible time windows and traveling times, *Discrete Applied Mathematics*, Elsevier, pp. 2271–2290. https://doi.org/10.1016/j.dam.2006.04.017.

[8] Dhaenens, C., Espinouse, M.-L., and Penz, B. (2002) Problemes combinatoires classiques et techniques de résolution, *Recherche Opérationnelle et Réseaux*. hal-01691518.

[9] Dantzig, G., Fulkerson, R., and Johnson, S. (1954) Solution of a large-scale traveling-salesman problem, *Journal of the Operations Research Society of America*, INFORMS, pp. 393–410. https://doi.org/10.1287/opre.2.4.393.

[10] Toth, P. and Vigo, D. (2014) Vehicle routing: problems, methods, and applications, *SIAM*.

[11] Rios, B. H. O., Xavier, E. C., Miyazawa, F. K., Amorim, P., Curcio, E., and Santos, M. J. (2021) Recent dynamic vehicle routing problems: A survey, *Computers & Industrial Engineering*, Elsevier, pp. 107604. https://doi.org/10.1016/j.cie.2021.107604.

[12] Asghari, M., Mirzapour Al-e, S. M. J., et al. (2021) Green vehicle routing problem: A state-of-the-art review, *International Journal of Production Economics*, Elsevier, pp. 107899. https://doi.org/10.1016/j.ijpe.2020.107899.

[13] Zhang, H., Ge, H., Yang, J., and Tong, Y. (2022) Review of vehicle routing problems: Models, classification and solving algorithms, *Archives of Computational Methods in Engineering*, Springer, pp. 1–27. https://doi.org/10.1007/s11831-021-09659-y.

[14] Toth, P., & Vigo, D. (2002). VRP with backhauls. In The vehicle routing problem (pp. 195-224). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9780898718515.ch8.

[15] Konstantakopoulos, G. D., Gayialis, S. P., and Kechagias, E. P. (2022) Vehicle routing problem and related algorithms for logistics distribution: A literature review and classification, *Operational Research*, Springer, pp. 2033–2062. https://doi.org/10.1007/s12351-020-00604-5.

[16] Sluijk, N., Florio, A. M., Kinable, J., Dellaert, N., and Van Woensel, T. (2023) Two-echelon vehicle routing problems: A literature review, *European Journal of Operational Research*, Elsevier, pp. 865–886. https://doi.org/10.1016/j.ejor.2022.06.006.

[17] Sajid, M., Singh, J., Haidri, R. A., Prasad, M., Varadarajan, V., Kotecha, K., and Garg, D. (2021) A novel algorithm for capacitated vehicle routing problem for smart cities, *Symmetry*, MDPI, pp. 1923. https://doi.org/10.3390/sym13101923.

[18] Bräysy, O. and Gendreau, M. (2005) Vehicle routing problem with time windows, Part I: Route construction and local search algorithms, *Transportation Science*, INFORMS, pp. 104–118. https://doi.org/10.1287/trsc.1040.0105.

[19] Sharma, S. K., Routroy, S., and Yadav, U. (2018) Vehicle routing problem: recent literature review of its variants, *International Journal of Operational Research*, Inderscience Publishers, pp. 1–31. https://doi.org/10.1504/ijor.2018.094229.

[20] Toth, P. and Vigo, D. (2002) Vehicle Routing: Problems, Methods, and Application, *SIAM*. https://doi.org/10.1137/1.9781611973594.fm.

[21] Nasri, M., Hafidi, I., & Metrane, A. (2020). Multithreading parallel robust approach for the VRPTW with uncertain service and travel times. Symmetry, 13(1), 36. https://doi.org/10.3390/sym13010036.

[22] Suarez, L. G. V. (2016) A dynamic programming operator for metaheuristics to solve vehicle routing problems with optional visits, *PhD Thesis*, INSA de Toulouse. https://tel.archives-ouvertes.fr/tel-01355746v2.

[23] Dantzig, G. B. and Ramser, J. H. (1959) The truck dispatching problem, *Management Science*, INFORMS, pp. 80–91. https://doi.org/10.1287/mnsc.6.1.80.

[24] Solomon, M. M. (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints, *Operations Research*, INFORMS, pp. 254–265. https://doi.org/10.1287/opre.35.2.254.

[25] Clarke, G. and Wright, J. W. (1964) Scheduling of vehicles from a central depot to a number of delivery points, *Operations Research*, INFORMS, pp. 568–581. https://doi.org/10.1287/opre.12.4.568.

[26] Schaus, P. and Hartert, R. (2013) Multi-objective large neighborhood search, *Principles and Practice of Constraint Programming, CP 2013*, Springer, Uppsala, Sweden, pp. 611–627. https://doi.org/10.1007/978-3-642-40627-0_45

[27] El-Sherbeny, N. A. (2010) Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods, *Journal of King Saud University-Science*, Elsevier, pp. 123–131. https://doi.org/10.1016/j.jksus.2010.03.001

[28] Boualamia, H., Metrane, A., Hafidi, I., & Mellouli, O. (2022, November). A new adaptation mechanism of the ALNS algorithm using reinforcement learning. In International Conference of Machine Learning and Computer Science Applications (pp. 3-14). Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-29313-9_1.

[29] Hao, Q., & Huang, H. (2025). Multi-hop Network Security Strategy Integrating ACO Algorithm and PSO Algorithm. Informatica, 49(17). https://doi.org/10.31449/inf.v49i17.7499

[30] Labdiad, F., Nasri, M., Hafidi, I., and Khalfi, H. (2021) A modified adaptive large neighbourhood search for a vehicle routing problem with flexible time windows, *Mathematical Modeling and Computing*, Lviv Polytechnic National University, pp. 716–725. https://doi.org/10.23939/mmc2021.04.716

[31] Wen, Q. (2025). Multi Objective Optimization System for Bridge Design Based on Multi-objective Optimization Theory and Improved Ant Colony Algorithm. Informatica, 49(8). https://doi.org/10.31449/inf.v49i8.7220

[32] Kurniawan, M., Yuliastuti, G. E., Agustini, S., Hakimah, M., and Widyanto, W. (2025). Optimizing Swarm Intelligence: A Comprehensive Analysis of Mutation-Based Enhancements. Informatica, 49(15). https://doi.org/10.31449/inf.v49i15.5524.