Clustering-Based Enhanced Ant Colony Optimization for Multi-Trip Vehicle Routing Problem with Heterogeneous Fleet and Time Windows: An Industrial Case Study

Beom Sae (Shawn) Kim
beomsae.kim1@ucalgary.ca
University of Calgary
Calgary, Alberta, Canada

Arash Mozhdehi
arash.mozhdehi@ucalgary.ca
University of Calgary
Calgary, Alberta, Canada

Yunli Wang
yunli.wang@nrc-cnrc.gc.ca
National Research Council
Ottawa, Ontario, Canada

Sun Sun
sun.sun@nrc-cnrc.gc.ca
National Research Council
Waterloo, Ontario, Canada

Xin Wang*
xcwang@ucalgary.ca
University of Calgary
Calgary, Alberta, Canada

Abstract

This paper introduces a novel approach to solving a practical variant of the Vehicle Routing Problem (VRP), the multi-trip VRP with heterogeneous fleet and time windows (MTVRPHFTW). The approach integrates an improved Ant Colony Optimization (IACO) metaheuristic, a modified density-based spatial clustering of application with noise (DBSCAN-Plus) clustering, and a Micro-Cluster Fusion Scheme. The proposed framework aims to optimize vehicle routes by minimizing total travelling distance and time while ensuring a fair distribution of workload among the vehicles (drivers). To evaluate the proposed algorithm, referred to as the Ant Colony Optimization (ACO) algorithm with improvement mechanisms (Cluster Improved Ant Colony Optimization, CIACO), real-world data from a logistics company in Canada was utilized. This empirical testing aims to validate the algorithm's effectiveness in practical applications. The experimental results of CIACO demonstrate that the proposed algorithm outperforms existing methods in terms of reducing traveling distance, minimizing traveling time, optimizing the use of smaller vehicles to reduce CO2 emissions, achieving balanced workloads among drivers, and improving overall route optimization.

CCS Concepts

• **Mathematics of computing** → **Combinatorial optimization**;
• **Applied computing** → **Transportation**;

Keywords

Vehicle Routing Problem, Workload Balance, Ant Colony Optimization, DBSCAN+, Micro-Clustering, Route Optimization

*Corresponding author

IWCTS'24, October 29-November 1 2024, Atlanta, GA, USA

ACM ISBN 979-8-4007-1151-0/24/10
https://doi.org/10.1145/3681772.3698216

ACM Reference Format:
Beom Sae (Shawn) Kim, Arash Mozhdehi, Yunli Wang, Sun Sun, and Xin Wang. 2024. Clustering-Based Enhanced Ant Colony Optimization for Multi-Trip Vehicle Routing Problem with Heterogeneous Fleet and Time Windows: An Industrial Case Study. In *17th ACM SIGSPATIAL International Workshop on Computational Transportation Science GenAI and Smart Mobility Session (IWCTS'24), October 29-November 1 2024, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3681772.3698216

1 Introduction

The Vehicle Routing Problem (VRP) is a significant optimization challenge in logistics and transportation, focused on determining the most efficient routes for a fleet of vehicles to deliver goods to customers. While traditional VRP models primarily aim to minimize transportation costs, real-world applications present additional complexities, such as balancing workloads among drivers, managing heterogeneous fleets, and adhering to strict time windows. Addressing these complexities requires advanced approaches that ensure both operational efficiency and fairness.

VRP-solving approaches are generally categorized into exact algorithms and heuristics, with heuristics further divided into constructive and improvement methods [8]. Given that the Multi-Trip VRP with Real Heterogeneous Fleet and Time Windows (MTVRPHFTW) is an NP-Hard problem, exact algorithms are computationally infeasible for large-scale, real-world applications [3]. Improvement heuristics can be effective in practice, but they typically require an initial feasible solution, which is often challenging to obtain for most VRP variants [3]. These challenges require the development of advanced algorithms capable of efficiently managing the complexity and scale of real-world VRPs.

The main difference between the traditional VRP and the MTVRPHFTW lies in the latter's ability to handle a heterogeneous fleet with varying capacities. Unlike many studies that focus solely on vehicle weight constraints, our approach also considers the number of skids as a crucial capacity factor, offering a more comprehensive solution. Additionally, the MTVRPHFTW allows each vehicle to perform multiple trips within a specified planning horizon while ensuring that customer-specific time windows are respected [8].

To address these challenges, we propose CIACO, designed specifically for solving the MTVRPHFTW. CIACO integrates clustering techniques with an enhanced Ant Colony Optimization (ACO)

method. It begins by employing a modified DBSCAN algorithm (DBSCAN-Plus) to group customers into micro-clusters based on spatial proximity, demand and time windows while considering vehicle capacity constraints. These micro-clusters are then fused to ensure balanced workloads across all vehicles. The enhanced ACO algorithm optimizes vehicle routes by incorporating a return mechanism to handle multi-trip scenarios and by considering vehicle types to maximize the use of smaller vehicles.

The main objectives are to minimize the total traveling distance and time, reduce load imbalance to ensure fair workload distribution among vehicles and maximize the use of smaller vehicles. Using real-world data from a Canadian logistics company, the proposed framework demonstrates significant improvements in transportation cost reduction and workload balance compared to traditional methods. The approach enhances the use of smaller vehicles, contributing to overall efficiency and sustainability.

This paper is structured as follows: Section 2 reviews related work in VRP optimization, focusing on workload balance and heterogeneous fleets. Section 3 presents the problem definitions and assumptions for MTVRPHFTW. Section 4 details the proposed methodology and algorithms, introducing novel integration of clustering and optimization techniques that form the core of our approach. Section 5 discusses the experimental setup and results, highlighting the improvements achieved. Finally, Section 6 concludes the paper and outlines potential directions for future research.

2 Related Works

Despite extensive research on traditional VRPs, studies specifically addressing MTVRPHFTW are limited, particularly in terms of investigating this problem under realistic conditions and benchmarking performance using real industrial data. This gap highlights the need for more research focused on practical solutions to MTVRPHFTW.

In the Multi-Trip VRP (MTVRP), drivers made multiple trips per day, influenced by factors like perishable goods or maximum driving hour regulations [2, 16]. This variant was relevant in city logistics with small vehicles and multiple trips, and it included a time horizon to record the duration of each trip [2, 16].

The Multi-Trip Vehicle Routing Problem with Time Windows and Heterogeneous Fleet (MTVRPTWHF) introduced a complex variation of the classic VRP by incorporating additional constraints related to time windows for each customer, the capacity of the vehicle, and the number of skids [3, 4, 8, 14, 15]. This problem variant considered not only the scheduling of deliveries within specific time windows for each customer but also the operational aspect of utilizing a heterogeneous fleet capable of executing multiple trips [11]. A critical constraint was the overall time horizon for completing all routes, which could not be exceeded, reflecting the real-world logistical challenge of meeting customer demands within a finite operational period. The solution approach combined Local Search and Simulated Annealing techniques, aiming to optimize route planning and fleet utilization efficiently [4]. This methodology was tested against widely recognized VRP benchmarks to validate its effectiveness and performance in addressing the intricacies of MTVRPTWHF [4, 5].

ACO algorithms, initially proposed by Dorigo et al. (1996), were widely applied to various combinatorial optimization problems, including VRPs [6]. The algorithm's ability to find optimal paths through probabilistic decision-making and pheromone trails made it particularly suitable for dynamic and complex problems like MTVRP [1]. The specific application of ACO in heterogeneous fleet scenarios became a topic of increasing interest. When ACO was tailored to account for different vehicle types and capacities, it led to more cost-effective and practical routing solutions [1, 6, 8, 17]. This research highlighted the adaptability of ACO to diverse logistical requirements.

Beginning with the foundational work by Blum and Roli [1], which offered a broad overview of metaheuristics, this section set the groundwork for understanding the significant role of ACO in combinatorial optimization. Their analysis highlighted the critical balance between intensification and diversification strategies, essential for effective solution space exploration and exploitation in complex optimization problems, including those in vehicle routing.

Phuc and Thao [12] applied ACO to the complex Multi Pickup and Multiple Delivery Vehicle Routing Problem with Time Window and Heterogeneous Fleets (MPMDVRPTWHF), illustrating ACO's ability to adapt to logistical constraints such as varying vehicle capacities and strict delivery schedules, aiming to minimize total travel costs. This study demonstrated ACO's potential in optimizing routes to accommodate a wide range of operational constraints and fleet diversities, indicating its suitability for complex logistical challenges. Furthering the application of ACO, Yu et al. [19] introduced an improved variant for the VRP, incorporating strategies like the "ant-weight strategy" and mutation operations to boost the algorithm's performance. Their approach confirmed ACO's capacity to produce solutions that were competitive with traditional methods, highlighting its adaptability and efficiency in routing optimization across different problem settings.

Mazzeo and Loiseau [13] concentrated on the Capacitated Vehicle Routing Problem (CVRP), exploring various aspects of the ACO algorithm, such as route building, transition rules, pheromone updates, and the implementation of improvement heuristics. Their findings pointed to ACO's competitive advantage over other metaheuristics, especially in handling problems up to 50 nodes and showing promising applicability for larger issues.

Gupta and Saini [7] enhanced the traditional ACO framework for the Vehicle Routing Problem with Time Windows (VRPTW) by introducing a new pheromone reset and update function alongside a 2-opt method for path improvement. Their version of the enhanced ACO demonstrated substantial improvements in routing optimization, emphasizing the algorithm's effectiveness in managing time-constrained routing challenges.

Wu et al. [18] proposed a Hybrid Ant Colony Optimization (HACO) for the VRPTW, incorporating a unique blend of strategies, including a novel pheromone update method, adaptive parameters, and mutation operations. This approach aimed to overcome the limitations of traditional ACO by enhancing solution diversity and avoiding premature convergence to local optima. Based on Solomon's instances, their experimental results demonstrated HACO's effectiveness in optimizing routes within specified time windows, highlighting its practical implications for complex routing problems. This study further solidified the adaptability and potential of ACO-based methods in addressing the intricacies of VRPTW,

contributing to the ongoing development of efficient logistical and transportation solutions.

These studies demonstrated the versatility and efficiency of ACO in solving complex vehicle routing problems, particularly in scenarios that involved different types of fleets and multiple trips. ACO continuously evolved through improvements and adaptations and remained a powerful tool in logistics and transportation, significantly reducing the number of vehicles required and the overall distance travelled, thereby improving operational efficiency and sustainability in the logistics sector. Unfortunately, these studies mostly focused on reducing total distance or total traveling time but did not address the important aspect of improving load imbalance for drivers or vehicles, which was crucial for ensuring fair workload distribution and operational equity.

3 Problem Definition and Assumptions

In this section, we define the problem of solving an MTVRPHFTW and outline the core assumptions and notations used in the formulation. This problem is a practical extension of the VRP designed to accommodate multiple trips, vehicles of varying sizes with different capacities, strict service time windows, and customers' specific preferences.

The MTVRPHFTW is modeled as a complete graph $G = (N, E)$, where:

- $N = \{n_0, n_1, n_2, \ldots, n_n\}$ represent the set of geographically distributed nodes, with n_0 denoting the depot and the remaining nodes representing customer locations.
- $E = \{(i, j) | i, j \in N\}$ is the set of edges defining the possible routes connecting these nodes.

Each customer i has a demand d_i consisting of both weight (mass of goods to be delivered) and the number of skids (pallets used for transport). Customers must be serviced within specific time windows $[t_i^{\text{start}}, t_i^{\text{end}}]$, where t_i^{start} and t_i^{end} represent the earliest and latest allowable service times for customer i, respectively. The service time at each node n_i, denoted by s_i, refers to the time required to complete the service, with s_0 specifically indicating the loading time at the depot. The travel time between nodes i and j is indicated by t_{ij}. We define the set $H^f = \{0, 1, 2, \ldots, |H^f|\}$ to represent the heterogeneous fleet types based on their capacities. In our industrial case, the fleet consists of three different vehicle types, denoted by f, where the largest truck is labelled as 1, the mid-sized truck is labelled as 2, and the smallest truck is labelled as 3. The capacity of a vehicle $v \in H$, in terms of weight and number of skids, is denoted by C_v. Additionally, each customer i may have a preference for a specific vehicle type, indicated by p_i. If customer i has no preference, p_i is set to 0. The maximum allowable working hours per day for each vehicle is represented by $T_{\max}$.

4 Methodology

4.1 Overview

In this section, we propose the use of the CIACO algorithm for practical application in the industry for MTVRPHFTW. The approach integrates clustering techniques and enhanced ACO to achieve balanced workloads, minimize travel costs, optimize the use of smaller vehicles to reduce CO_2 emissions and adhere to time window constraints. The framework is structured into four phases: customer clustering (DBSCAN-Plus), micro-cluster fusion, and route optimization, accommodating practical logistics constraints such as multiple trips per vehicle, varying vehicle sizes and capacities, and customer-specific service time windows.

Figure 1 illustrates the structured phases of the CIACO framework, which systematically optimizes vehicle routing by clustering customers, merging micro-clusters, and refining routes to meet practical logistics constraints. Each phase will be discussed in more detail in Sections 4.2, 4.3, 4.4, and 4.5.

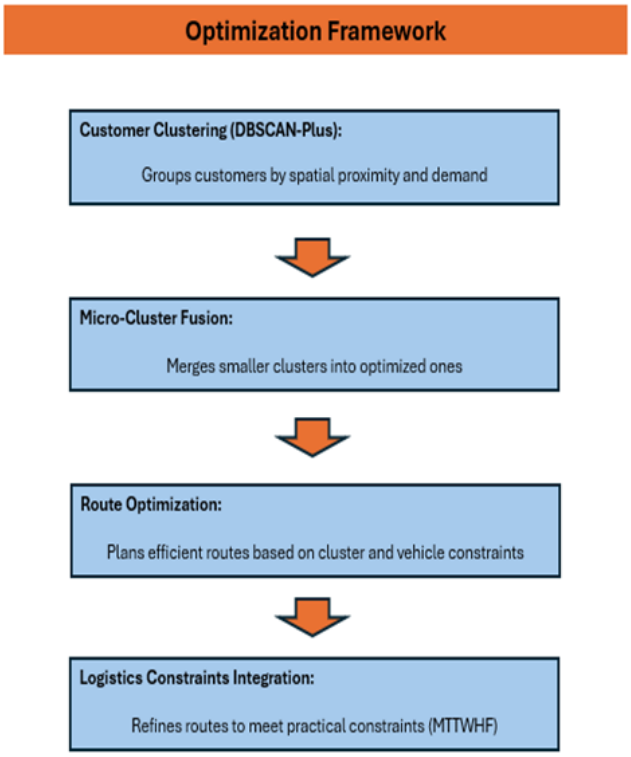

Figure 1: CIACO optimization framework

4.2 Clustering and Micro-Cluster Fusion

4.2.1 *DBSCAN-Plus Clustering.*

To reduce computational complexity and ensure balanced workload distribution in the context of MTVRPHFTW, we developed the DBSCAN-Plus algorithm. This approach builds upon the traditional DBSCAN clustering method, previously used in various vehicle routing optimization frameworks, such as the work by Li, Fang, and Tang [10]. However, our adaptation—DBSCAN-Plus—incorporates several key modifications to better handle the specific challenges posed by MTVRPHFTW. The main procedure of DBSCAN-Plus is outlined in Algorithm 1.

The DBSCAN-Plus algorithm begins by normalizing customer demands relative to vehicle capacities, as shown in Line 3 of Algorithm 1. This normalization is expressed as $D_i^{\text{normalized}} = \frac{D_i}{C}$ for

each customer i, ensuring that the clustering process accurately reflects the varying demands of different customers, similar to the technique used by Li et al. [10]. Following this, DBSCAN is applied to these normalized demands (Lines 4-5 in Algorithm 1), producing initial clusters. The total demand for each cluster k is then calculated as Total Demand$_k = \sum_{i \in \text{Cluster}_k} D_i$ (Lines 11-12 in Algorithm 1).

While the foundational clustering process is similar to that described in [10], our DBSCAN-Plus algorithm includes specific modifications to address the more complex scenario of multi-trip vehicle routing with a heterogeneous fleet and time windows. The original work in [10] dealt with a single-trip problem, whereas our approach must accommodate multiple trips per vehicle, varying vehicle capacities (weight and skid) and strict time windows for deliveries. These additional constraints add significant complexity to the clustering and optimization processes.

First, Enhanced Demand Normalization (Lines 3-5 in Algorithm 1) adjusts for the presence of multiple vehicle types with differing capacities for both weight and skids. This adjustment allows the clustering process to more accurately reflect real-world constraints. Second, during the Cluster Fusion stage (Lines 13-21 in Algorithm 1), the algorithm checks if any cluster's total demand exceeds the vehicle's capacity C. If it does, the cluster is split into smaller sub-clusters to ensure that each sub-cluster's demand remains within manageable limits.

This sub-clustering approach is specifically tailored to address the MTVRPHFTW scenario, where multiple vehicle types and constraints must be considered simultaneously. Finally, the Integration with ACO Algorithm 2 uses these Micro-Clusters as input for an ACO algorithm. The structured initial conditions provided by the clustering phase help in achieving faster convergence and better optimization outcomes, enabling our approach to handle the multi-trip, heterogeneous fleet scenario more effectively.

In summary, the DBSCAN-Plus algorithm effectively breaks the problem into smaller, more manageable clusters, leading to faster convergence and consistent initial conditions for the ACO algorithm. By improving the utilization of smaller vehicles, this approach reduces the need for larger vehicles and lowers CO_2 emissions, as vehicle weight is a critical factor in emission calculations [9]. Compared to [10] and traditional DBSCAN, which may not fully account for demand (weight and skid) and time windows or efficiently merge clusters in multi-trip scenarios, Cluster-ACO offers a more robust and adaptable solution. It is specifically designed to address the complexities of the multi-trip, heterogeneous fleet scenario in ways that traditional approaches do not, making it a more comprehensive and effective method.

4.2.2 *Micro-Cluster Fusion.*

Following the initial clustering phase, the Micro-Cluster Fusion process is applied to optimize the cluster configuration further. This step involves merging smaller Micro-Clusters into larger ones while ensuring that the total demand within each fused cluster does not exceed the vehicle's capacity. The fusion process not only enhances the efficiency of the delivery routes by reducing the number of trips required but also helps in minimizing the overall travel distance. By carefully managing the fusion of Micro-Clusters, the algorithm achieves a more cohesive and balanced set of customer clusters,

Algorithm 1 DBSCAN-Plus Clustering

1: **Input:**
Set of customer locations (coordinates x and y) $\mathbf{X}$
Demands $\mathbf{D}$
Vehicle capacity C
DBSCAN parameters: ϵ (eps) and *min_samples*
2: **Output:** Clusters satisfying capacity constraints *fused_clusters*
3: Normalize demands by vehicle capacity:

$$\mathbf{D}_{\text{normalized}} = \frac{\mathbf{D}}{C}$$

4: Perform DBSCAN clustering on $\mathbf{X}$, weighted by $\mathbf{D}_{\text{normalized}}$:
5: labels $\leftarrow$ DBSCAN($\mathbf{X}, \epsilon$, min_samples, sample_weight = $\mathbf{D}_{\text{normalized}}$)
6: Extract initial clusters:
initial_clusters $\leftarrow$ {label : indices of points with label in labels}
7: Adjust cluster indices for correct customer referencing:
8: adjusted_clusters $\leftarrow$ Adjust indices of initial_clusters
9: by adding 1
10: Initialize an empty list for valid clusters:

$$\text{fused\_clusters} \leftarrow []$$

11: **for** each cluster in *adjusted_clusters* **do**
12: Compute the total demand for the current cluster: total_demand $\leftarrow \sum_{i \in \text{cluster}} \mathbf{D}[i]$
13: **if** total_demand exceeds vehicle capacity C **then**
14: Sort customers in the cluster by demand in descending order: sorted_cluster $\leftarrow$ Sort(cluster)
15: Initialize a new sub-cluster: sub_cluster_demand $\leftarrow$ 0, sub_cluster $\leftarrow$ []
16: **for** each customer i in sorted_cluster **do**
17: **if** Adding customer i does not exceed capacity **then**
18: Add customer i to the sub-cluster: sub_cluster $\leftarrow$ sub_cluster $\cup$ $\{i\}$
19: Update sub-cluster demand: sub_cluster_demand $\leftarrow$ sub_cluster_demand + $\mathbf{D}[i]$
20: **else**
21: Add the completed sub-cluster to fused_clusters: fused_clusters $\leftarrow$ fused_clusters $\cup$ sub_cluster
22: Start a new sub-cluster with customer i: sub_cluster $\leftarrow$ $\{i\}$, sub_cluster_demand $\leftarrow$ $\mathbf{D}[i]$
23: **end if**
24: **end for**
25: **if** sub_cluster is not empty **then**
26: Add the remaining sub-cluster to fused_clusters: fused_clusters $\leftarrow$ fused_clusters $\cup$ sub_cluster
27: **end if**
28: **else**
29: Add the entire cluster to fused_clusters: fused_clusters $\leftarrow$ fused_clusters $\cup$ cluster
30: **end if**
31: **end for**
32: **Return** *fused_clusters*

Algorithm 2 Microcluster-Fusion

1: **Input:**
   List of fused clusters *fused_clusters*
   Demands of each cluster *demands*
   Maximum vehicle capacity *C_max*
2: **Output:** Final list of fused microclusters *final_fused_clusters*
3: Initialize an empty list for fused microclusters: *final_fused_clusters* ← []
4: Sort *fused_clusters* in descending order based on their total demand
5: **while** there are fused clusters left to process **do**
6: Select the largest fused cluster *current_cluster* from *fused_clusters*
7: Remove *current_cluster* from *fused_clusters*
8: Initialize a new cluster *new_cluster* with *current_cluster*
9: Set *current_demand* as the total demand of *current_cluster*
10: **for** each *candidate_cluster* in *fused_clusters* **do**
11: Calculate *candidate_demand* as the total demand of *candidate_cluster*
12: **if** *current_demand* + *candidate_demand* ≤ *C_max* **then**
13: Merge *candidate_cluster* into *new_cluster*
14: Update *current_demand* ← *current_demand* + *candidate_demand*
15: Remove *candidate_cluster* from *fused_clusters*
16: **end if**
17: **end for**
18: Add *new_cluster* to *final_fused_clusters*
19: **end while**
20: **Return** *final_fused_clusters*

Algorithm 3 Environment Setup and Update

1: **Initialization:**
2: Initialize pheromone matrix τ with ones
3: Set diagonal of τ to zero
4: Compute heuristic matrix η as inverse of distance matrix
5: Normalize η to [0, 1] scale using min-max normalization
6: Initialize time window constraints and load balancing factors
7: Compute time window restriction matrix
8: **End Initialization**
9: **function** UpdatePheromone(C)
10: Decay existing pheromones in τ using evaporation rate
11: **for** each solution in C **do**
12: Update pheromones on paths taken by solution based on fitness
13: **end for**
14: **end function**
15: **function** UpdateProbability
16: **for** each customer i **do**
17: Compute normalization factor Z_i as:

$$Z_i = \sum_{j=1}^{n} \tau[i,j] \cdot \eta[i,j]$$

18: **for** each customer j **do**
19: Calculate probability of moving from i to j using:

$$P[i,j] \leftarrow \frac{\tau[i,j] \cdot \eta[i,j]}{Z_i}$$

20: **end for**
21: **end for**
22: **end function**

ultimately contributing to a more efficient and cost-effective delivery operation. The main procedure of Micro-Cluster is given in Algorithm 2.

In the Micro-Cluster Fusion process, clusters are merged based on their total demand. For each pair of clusters considered for merging, the combined demand is calculated as Combined Demand = $\text{Total\_Demand}_{\text{current}} + \text{Total\_Demand}_{\text{candidate}}$ (Lines 10-11 in Algorithm 2). This merger occurs only if the combined demand does not exceed the vehicle's capacity (Combined Demand ≤ $C_{\max}$) (Line 11 in Algorithm 2). After merging, the demand for the new cluster is updated to $\text{Total\_Demand}_{\text{new cluster}} = \text{Total\_Demand}_{\text{current}} + \text{Total\_Demand}_{\text{candidate}}$ (Lines 12-16 in Algorithm 2), ensuring that the resulting clusters are both efficient and within capacity limits.

4.3 Route Optimization

4.3.1 *IACO Environment Setup and Update.*

The Environment Setup and Update algorithm is a critical component of the Ant Colony Optimization (ACO) process, ensuring that the ants' environment is properly initialized and maintained throughout the optimization. This algorithm aims to set up the initial pheromone levels and heuristic information, which guide the ants in constructing their solutions. Additionally, the algorithm includes mechanisms for updating the pheromone trails based on the quality of solutions found and recalculating transition probabilities to reflect changes in the environment. This dynamic adjustment helps in balancing the exploration and exploitation aspects of the search process [6]. The detailed steps of the Environment Setup and Update are presented in Algorithm 3.

The Algorithm 3 begins by setting up the pheromone matrix τ with initial values of 1, where the diagonal elements are set to 0 to prevent self-loops (Lines 2-3 in Algorithm 3). The heuristic matrix is computed as the inverse of the distance matrix and normalized to a [0,1] range (Lines 4-5 in Algorithm 3). Additionally, time window constraints and load-balancing factors are initialized, along with the time window restriction matrix (Lines 6-7 in Algorithm 3).

The *Update-Pheromone* function (Lines 9-13 in Algorithm 3) updates the pheromone values, applying pheromone evaporation and reinforcing paths taken by high-quality solutions. The *Update-Probability* function (Lines 15-22 in Algorithm 3) recomputes the probabilities of transitioning between customers by first computing the normalization factor Z_i as the sum of pheromone and heuristic products (Lines 17-18 in Algorithm 3). These probabilities guide the ants in constructing their routes based on the updated pheromone and heuristic information.

4.3.2 *IACO for MTVRPHFTW.*

The IACO for MTVRPHFTW with Time Window Penalty algorithm is designed to iteratively optimize solutions for the MTVRPHFTW. This algorithm applies the principles of ACO to find the

Algorithm 4 IACO for MTVRPHFTW with Time Window Penalty

1: **Data:** MTVRPHFTW instance
2: **Result:** Best solution
3: Initialize *env* // (ACO environment) Initialize pheromone matrix, heuristic matrix, and other parameters
4: *colony* ← [] // Collection of solutions and their fitness
5: **for** each $\alpha \in \mathcal{A}$ **do**
6: *sol* ← InitializeSol()
7: *colony.append*(*sol*)
8: **end for**
9: *best* ← *colony*[*argmin*(colony.fitness)] // Track the best solution
10: **for** each iteration **do**
11: **for** each $\alpha \in \mathcal{A}$ **do**
12: *sol* ← InitializeSol() // Reinitialize solution for each ant
13: *sol.route* ← AntMovement(*env*) // Construct solution routes
14: *sol.fitness* ← EvaluateSol(*sol.route*)
15: Calculate time window penalty for *sol.route*
16: *sol.fitness* ← *sol.fitness* + time_window_penalty // Apply the penalty
17: Sort *colony* by fitness // Sort colony to maintain the best solutions
18: *colony*[*worst*] ← *sol* // Replace the worst solution if the new solution is better
19: **end for**
20: Update *env.pheromone* // Update pheromone levels based on current solutions
21: Update *env.probability* // Update transition probabilities for the next iteration
22: **if** *min*(colony.fitness) < *best.fitness* **then**
23: *best* ← *colony*[*argmin*(colony.fitness)] // Update the best solution found
24: **end if**
25: **end for**
26: **return** *best* // Return the best solution found

most efficient routing solutions that minimize travel costs while meeting operational constraints, such as vehicle capacities and customer time windows. To handle violations of time constraints, a time window penalty is applied to the fitness of each solution, guiding the search toward feasible and optimal routes. The main steps of this process are outlined in Algorithm 4.

The ACO algorithm simulates the behavior of an ant colony in searching for food sources by depositing pheromones along paths, which in turn influences the decisions of other ants [6]. In this context, the set of ants is denoted by A, where each ant $\alpha \in A$ constructs a solution by iteratively selecting the next node to visit, forming a sequence of nodes $i \in N$ that begins and ends at the depot.

After all ants have constructed their solutions, the pheromone levels on the paths are updated through both evaporation and deposition processes. As shown in equation (1), the pheromone level $\tau_t(i,j)$ on the path from the node i to node j at iteration t decreases by an evaporation rate ρ (where $0 < \rho < 1$). It is also increased by the amount of pheromone deposited by each ant $a \in A$, denoted as $\Delta\tau_t^a(i,j)$, which is calculated as the inverse of the length of the solution L_a ($\tau_t = \frac{1}{L_a}$) [6, 8].

$$\tau_t(i,j) = (1-\rho) \cdot \tau_{t-1}(i,j) + \sum_{a \in A} \Delta\tau_t^a(i,j) \tag{1}$$

The ants determine the next node to visit based on a transition probability that considers both the pheromone level and the proximity (or visibility) of the nodes [6, 8]. The visibility from node i to j is represented as h_{ij}^b, where h_{ij} denotes the visibility between nodes i and j, and β is a constant reflecting the importance of visibility. The set of feasible neighbor nodes that can be visited from node i is denoted by N_i. The probability of transitioning from node i to node j at iteration t is calculated as [8]:

$$P_t(i,j) = \frac{\tau_t(i,j)^\alpha h_{ij}^\beta}{\sum_{j \in N_i} \tau_t(i,j)^\alpha h_{ij}^\beta} \tag{2}$$

Algorithm 4 begins by initializing the ACO environment (env) and an empty colony to store the solutions (Lines 3-4). Each $\alpha \in A$ is assigned an initial solution using the 'InitializeSol' function, and the best solution is identified and recorded (Lines 5-8).

In the main loop (Lines 10-25), the algorithm iteratively constructs solutions for each ant by generating routes using the 'AntMovement' function, as detailed in Algorithm 5. The fitness of each solution is evaluated, incorporating a time window penalty to account for any violations (Lines 15 in Algorithm 4). The colony is updated by replacing the worst solution with a better one if a superior solution is found (Lines 16-18 in Algorithm 4). Pheromone levels are adjusted based on the quality of the solutions, influencing decisions in subsequent iterations (Lines 20-21 in Algorithm 4). If a new best solution is discovered, it replaces the current best solution (Lines 22-23 in Algorithm 4). Algorithm 5 is critical in this process, as it determines the route construction for each ant. It considers the current environment, including pheromone levels, heuristic information, and the constraints on load and time. The algorithm ensures that the ants construct feasible routes that respect the vehicle capacity and time window constraint while also allowing for the return to the depot when necessary. The process continues until the termination criteria are met, at which point the algorithm returns the best solution found (Line 26 in Algorithm 4)

4.4 Constraints Handling

The MTVRPHFTW involves complex operational constraints that must be carefully managed to produce feasible and optimized solutions. This section outlines how these constraints are handled within the proposed framework, mainly focusing on the AntMovement function (Algorithm 5) and the initial clustering phases with DBSCAN-Plus and Micro Cluster-Fusion.

4.4.1 *Multi-Trip Constraint.*

The multi-trip nature of the MTVRPHFTW problem requires efficient vehicle utilization, where each vehicle may perform multiple trips. The 'AntMovement' function in Algorithm 5 manages this by tracking the current vehicle index (`curr_vidx`), cumulating load (`L`) and the cumulative time (T_{total}) of the vehicle as it serves customers. After each trip, the algorithm evaluates whether to continue serving more customers or return to the depot to start a new trip based on

Algorithm 5 AntMovement

1: **Input:**
   Environment variables: pheromone matrix τ, heuristic matrix η, probability matrix $\mathbf{P}$
   Set of customers $\mathbf{C}$, demands $\mathbf{D}$, capacity $\mathbf{C}_{\max}$
   Time matrix $\mathbf{T}$, service times $\mathbf{S}$
   Maximum time window max_time
2: **Output:** Constructed route for each ant
3: Initialize route $\mathbf{R} \leftarrow [0]$ // Start at depot (node 0)
4: Initialize cumulative load $\mathbf{L} \leftarrow \mathbf{0}$
5: Initialize cumulative time $\mathbf{T_{total}} \leftarrow 0$
6: Initialize vehicle index curr_vidx $\leftarrow 0$
7: Initialize vehicle type curr_vtype $\leftarrow 0$
8: Initialize set of unvisited customers $\mathbf{U} \leftarrow \mathbf{C}$
9: **while** $\mathbf{U} \neq \emptyset$ **do**
10: Calculate probabilities $\mathbf{P}(i, j)$ for each unvisited customer $j \in \mathbf{U}$
11: Select next customer j based on $\mathbf{P}(i, j)$
12: **if** $\mathbf{L} + \mathbf{D}_j \leq \mathbf{C}_{\max}$ and $\mathbf{T_{total}} + T(i, j) + S_j \leq$ max_time **then**
13: Add customer j to route $\mathbf{R} \leftarrow \mathbf{R} \cup \{j\}$
14: Update cumulative load $\mathbf{L} \leftarrow \mathbf{L} + \mathbf{D}_j$
15: Update cumulative time $\mathbf{T_{total}} \leftarrow \mathbf{T_{total}} + T(i, j) + S_j$
16: Remove j from $\mathbf{U}$
17: **else**
18: Return to depot and start a new trip
19: $\mathbf{R} \leftarrow \mathbf{R} \cup \{0\}$
20: Reinitialize $\mathbf{L}$ and $\mathbf{T_{total}}$
21: Update vehicle index curr_vidx $\leftarrow$ curr_vidx + 1
22: Set curr_vtype $\leftarrow 0$ //Reset vehicle type
23: **end if**
24: **end while**
25: **Return** route $\mathbf{R}$, vehicle indices **curr_vidx**, vehicle types **curr_vtype**

these variables (Lines 18-22 in Algorithm 5). This approach ensures that vehicles are optimally deployed across multiple trips to meet customer demands effectively.

4.4.2 *Time Window Constraint.*

Managing time windows is critical to ensuring deliveries occur within the specified time frames. The algorithm handles this by continuously updating the cumulative time (T_{total}) for each trip and checking if the addition of a customer would violate their time window. If adding the customer would result in exceeding the maximum time allowed (max_time), the vehicle returns to the depot to start a new trip (Lines 12-18 in Algorithm 5). This mechanism ensures that all deliveries respect the time constraints.

4.4.3 *Heterogeneous Fleet and Capacity Constraints.*

The heterogeneous fleet constraint involves varying vehicle capacities and capabilities, which is addressed by selecting the most appropriate vehicle based on the cumulative load (L). As the route is constructed, the algorithm continuously checks whether the cumulative load (L) remains within the vehicle's capacity ($C_{\max}$). If a customer's demand exceeds the vehicle's capacity, the vehicle returns to the depot to begin a new trip with an updated load (L_{updated}) and time (T_{total}) (Lines 14-18 in Algorithm 5).

The algorithm also uses the variable `curr_vtype` to ensure that only suitable customers are included in the route. If a customer does not specify a fleet type (indicated by a zero label), the algorithm defaults to the largest fleet type to prevent capacity violations (Lines 18-19 in Algorithm 5).

This approach ensures that each vehicle is optimally utilized, matching the vehicle's capacity with the demands of the assigned route while preventing overloads. By handling the heterogeneous fleet and capacity constraints in a unified process, the algorithm maintains feasible routes that respect each vehicle's operational limits.

4.4.4 *Improvement Mechanisms.*

Utilization of Smaller Vehicle Sizes

This strategy optimizes fleet utilization by assigning smaller vehicles to routes where feasible, potentially reducing CO_2 emissions. During the DBSCAN-Plus clustering phase, customer demands are normalized relative to vehicle capacities, enabling the identification of clusters suitable for smaller vehicles (Lines 3-5 in Algorithm 1). The Micro-Cluster Fusion step further refines these clusters by merging smaller clusters while ensuring the total demand remains within the capacity limits of the smallest feasible vehicle (Lines 6-13 in Algorithm 2). In the 'AntMovement' function (Algorithm 5), the algorithm dynamically assigns the most appropriate vehicle based on the cumulative load (L) and customer requirements, prioritizing smaller vehicles when possible (Lines 12-14 in Algorithm 5). To quantify the reduction in CO_2 emissions, the following equation is used [9]:

$$\text{CO}_2 \text{ Emissions (Kg)} = \frac{\theta(v)}{1000} \times SL$$

Where SL is the route segment length (km), v is the average speed of the vehicle (km/h), and $\theta(v)$ is the CO_2 emission factor (g/km), computed as:

$$\theta(v) = k + av + bv^2 + cv^3 + d\frac{1}{v} + e\frac{1}{v^2} + f\frac{1}{v^3}$$

The coefficients k, a, b, c, d, e, and f are derived based on the vehicle type and weight. The algorithm, therefore, optimizes route efficiency and aims to minimize environmental impact by selecting vehicles that produce lower emissions over the route segment length.

Time Window Penalty

The algorithm continuously checks for potential time window violations during route construction in the 'AntMovement' function (Algorithm 5). If adding a customer would breach their time window, a penalty is applied to the solution's fitness (Lines 13-16 in Algorithm 4). This discourages the selection of infeasible routes in future iterations, guiding the algorithm toward solutions that respect time constraints.

Load Balancing

Reducing load imbalance across vehicles is a critical improvement mechanism that the algorithm addresses through multiple stages. Initially, in the DBSCAN-Plus clustering phase, clusters are formed with careful consideration of demand distribution relative to vehicle capacities, ensuring that no vehicle is initially assigned an

excessive load (Lines 3-5 in Algorithm 1). During the Microcluster-Fusion process, these clusters are further refined to maintain a balanced distribution of total demand across all vehicles, minimizing the risk of overloading any single vehicle (Lines 5-15 in Algorithm 2).

In the 'AntMovement' function, load balancing is dynamically managed by reassigning vehicles to routes based on the current cumulative load (L) and cumulative time (T_{total}). The algorithm also tracks the number of trips and the cumulative working time for each vehicle to ensure fairness among drivers. This ensures that each vehicle carries a load as close as possible to its capacity without exceeding it and that the workload (in terms of both load and time) is equitably distributed among all vehicles and drivers (Lines 14-18 in Algorithm 5). By integrating these multiple load balancing factors, the algorithm prevents overloads and promotes fairness across the fleet.

5 Experiments

5.1 Experimental Setup

In this study, experiments were conducted using 31 real-world instances provided by a logistics company in Canada, each representing a typical day of operations. The company's fleet is heterogeneous, composed of three types of vehicles with varying capacities (weight and skid), labeled as 1 (largest), 2 (mid-sized), and 3 (smallest). Additionally, some customers have specific fleet requirements, while others may have no restrictions, denoted by 0.

The vehicles operate within a fixed time window, starting at 6:00 am and ending at the depot by 6:00 pm. The loading time at the depot is set to 60 minutes, with 20 minutes allocated for servicing each customer. The trucks are assumed to travel at an average speed of 0.7 km/m, equivalent to 42 km/h.

The objective function is to minimize the total travel distance. Additionally, it aims to ensure a fair distribution of workloads among the drivers and vehicles, as measured by load imbalance, and to maximize the use of smaller vehicles, which can potentially reduce both operational costs and CO_2 emissions.

Several performance metrics were used to evaluate the algorithm's practical applicability:

- Total traveling distance (km)
- Total traveling time (mins)
- Total CO2 emissions
- Number of trips per vehicle
- Number of total trips
- Number of vehicles
- Load imbalance: Measured by the standard deviation (SD) of (1) the total load delivered (TLD), (2) total travel time (TTT), and (3) route length (RL).
- Fleet composition (usage distribution across vehicle sizes)

The ACO algorithm was configured to achieve these objectives with 100 ants, 300 iterations, and a pheromone evaporation rate of 0.1. Due to the ACO process's inherent randomness, each instance is run five times, and the average results are reported to ensure reliability.

To validate the effectiveness of the proposed improvement mechanisms within the ACO framework, the experiments first compare these enhanced solutions with baseline solutions. Next, the algorithm's performance is compared with the actual industry solutions currently used by the company, providing a practical benchmark for evaluation.

5.2 Experimental Results

The experimental results, presented in Tables 1, 2, and 3 provide a comprehensive analysis of the performance of the CIACO, IACO, and Industry-standard approaches. Improvement computation was calculated as $\frac{\text{New Method}-\text{Baseline}}{\text{Baseline}} \times 100\%$. The main objective of the experiments was to assess improvements in total traveling distance, total traveling time, CO_2 emissions, and load balancing metrics across the different methods.

Table 1 provides a comparative analysis between CIACO, ACO, IACO, and Industry-standard approaches across key performance metrics. CIACO demonstrates notable improvements over ACO in several areas. Specifically, CIACO achieves a reduction in total traveling distance by approximately 4.08% relative to the ACO baseline. This decrease is accompanied by a corresponding 4.05% reduction in total travel time and a 4.46% decrease in CO_2 emissions, indicating that CIACO effectively optimizes route efficiency while minimizing environmental impact.

Additionally, CIACO lowers the number of trips per vehicle and the overall number of trips by 9.30% and 7.79%, respectively, highlighting an efficient use of fleet resources. However, CIACO employs a slightly higher number of vehicles than ACO, with an increase of 2.76%. This is due to the strategy of balancing the load more evenly across the fleet, thereby preventing any single vehicle from being overburdened.

When comparing CIACO to IACO and Industry-standard approaches, further improvements are evident. While CIACO results in a 1.50% increase in total traveling distance compared to IACO, it still provides a substantial reduction of 28.75% compared to industry standards. Regarding total travel time, CIACO is 2.49% longer than IACO but remains 23.24% shorter than Industry standards, highlighting efficiency despite the slight increase compared to IACO. Similarly, CIACO records CO_2 emissions that are 1.68% higher than IACO but demonstrate a significant 18.42% reduction compared to industry-standard practices. The number of trips per vehicle decreases by 6.67% compared to IACO and by 8.26% relative to industry standards, while the total number of trips drops by 1.99% compared to IACO and 20.19% compared to industry. Moreover, CIACO reduces the number of vehicles by 4.34% compared to IACO and by 34.68% compared to industry practices.

These results emphasize CIACO's effectiveness in achieving operational and environmental benefits while closely matching the efficiency of IACO and significantly outperforming industry-standard practices. Importantly, CIACO also demonstrates a notable improvement in load balancing across routes, which is further detailed in Table 3.

Table 2 provides insights into vehicle utilization, particularly focusing on the distribution of vehicle types across CIACO, IACO, and ACO. CIACO significantly outperforms in utilizing smaller vehicles (Vehicle Type 3) with a 3.87% higher utilization compared to IACO and an 8.63% increase relative to ACO. This result is consistent with the algorithm's design, which aims to maximize the use

Table 1: Comparative Experiment Results with Improvements from Baseline (ACO, IACO, and Industry)

	ACO[b]	CIACO[a]	IACO[c]	Industry[d]	Impv.[ab]	Impv.[ac]	Impv.[ad]
Total travelling distance (km)	1429.76	1371.51	1351.23	1924.89	-4.08%	1.50%	-28.75%
Total travelling time (mins)	1753.26	1682.15	1641.23	2191.58	-4.05%	2.49%	-23.24%
Total CO2 emissions	1658.79	1584.79	1558.66	1942.63	-4.46%	1.68%	-18.42%
No. of trips per vehicle	3.55	3.22	3.45	3.51	-9.30%	-6.67%	-8.26%
No. of total trips	33.62	31	31.63	38.84	-7.79%	-1.99%	-20.19%
No. of vehicles	9.43	9.69	10.13	14.83	2.76%	-4.34%	-34.68%

of smaller, more fuel-efficient vehicles where feasible, potentially reducing CO2 emissions. Conversely, CIACO shows a lower utilization of larger vehicles (Vehicle Type 1), with a decrease of 30.53% compared to IACO and 38.91% compared to ACO. This reallocation of vehicle types reflects CIACO's strategy to optimize fleet usage by deploying vehicles that match the demand requirements of each route rather than over-relying on larger, less efficient trucks.

Table 2: Average percentage differences in vehicle utilization

Vehicle Type	CIACO[a]	IACO[b]	ACO[c]	Impv.[ab]	Impv.[ac]
1	3.14	4.52	5.14	-30.53%	-38.91%
2	12.71	13.67	14.23	-7.02%	-10.68%
3	20.14	19.39	18.54	3.87%	8.63%

Table 3: Comparative Experiment Results: Load Imbalance

	CIACO[a]	IACO[b]	Indus.[c]	Impv.[ab]	Impv.[ac]
TLD SD diff.	13188.82 kg	15863.16 kg	12283.22 kg	-16.86%	7.37%
TTT SD diff.	33.69 mins	60.82 mins	57.58 mins	-44.60%	-41.49%
RL SD diff.	23.36 km	50.68 km	58.76 km	-53.88%	-60.25%

Load imbalance is a critical factor that impacts driver fairness and overall fleet efficiency. Table 3 analyzes three specific metrics: Total Load Delivered std diff, Total Travel Time std diff, and Route length std diff. These metrics were computed based on the standard deviation of load weights, travel times, and route lengths across different routes in the CIACO algorithm. The standard deviations provide an understanding of how balanced the workload is across the fleet.

CIACO significantly reduces load imbalances compared to both IACO and Industry standards. Specifically, CIACO achieves a 16.86% improvement in the Total Load Delivered std diff compared to IACO, although it is 7.37% less balanced than the Industry. This result highlights CIACO's ability to distribute load more evenly across vehicles, reducing the likelihood of any vehicle overloading. In terms of Total Travel Time std diff, CIACO shows a substantial 44.60% reduction compared to IACO and a 41.49% reduction relative to Industry. Finally, the Route length std diff is where CIACO's performance is particularly striking, with a 53.88% improvement over IACO and a 60.25% improvement over Industry. These results confirm that CIACO improves efficiency and contributes to a more balanced workload distribution, which is crucial for driver satisfaction and fleet management.

The experimental results show that CIACO offers considerable improvements over traditional ACO, IACO, and Industry-standard methods in total traveling distance, travel time, CO_2 emissions, and load balancing. CIACO optimizes vehicle utilization and distributes loads more evenly across the fleet while reducing environmental impact, making it a highly effective solution for modern logistics. These findings highlight CIACO's potential to enhance both operational efficiency and sustainability in real-world applications.

6 Conclusion

This paper addresses the practical challenges of the MTVRPHFTW problem in real-world logistics by proposing an enhanced ACO-based algorithm integrated with clustering techniques. The developed method includes additional mechanisms aimed at improving solution quality, such as more effective vehicle utilization and load balancing, which are essential for last-mile logistics operations. Using real industrial data, the results demonstrate that the proposed approach outperforms existing methods in both efficiency and performance.

Acknowledgments

This project was supported in part by collaborative research funding from the National Research Council of Canada's Artificial Intelligence for Logistics Program. The authors also would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for a part of financial support.

References

[1] Christian Blum and Andrea Roli. 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)* 35, 3 (2003), 268–308.

[2] Diego Cattaruzza, Nabil Absi, Dominique Feillet, and Thibaut Vidal. 2014. A memetic algorithm for the multi trip vehicle routing problem. *European Journal of Operational Research* 236, 3 (2014), 833–848.

[3] Jean-François Cordeau, Gilbert Laporte, Martin WP Savelsbergh, and Daniele Vigo. 2007. Vehicle routing. *Handbooks in operations research and management science* 14 (2007), 367–428.

[4] François Despaux and Sebastián Basterrech. 2014. A study of the multi-trip vehicle routing problem with time windows and heterogeneous fleet. In *2014 14th International Conference on Intelligent Systems Design and Applications*. IEEE, 7–12.

[5] François Despaux and Sebastián Basterrech. 2016. Multi-trip vehicle routing problem with time windows and heterogeneous fleet. *International Journal of Computer Information Systems and Industrial Management Applications* 8 (2016), 9–9.

[6] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. 1996. Ant system: optimization by a colony of cooperating agents. *IEEE transactions on systems, man, and cybernetics, part b (cybernetics)* 26, 1 (1996), 29–41.

[7] Ashima Gupta and Sanjay Saini. 2017. An enhanced ant colony optimization algorithm for vehicle routing problem with time windows. In *2017 ninth international conference on advanced computing (ICOAC)*. IEEE, 267–274.

[8] Jihee Han, Arash Mozhdehi, Yunli Wang, Sun Sun, and Xin Wang. 2022. Solving a multi-trip VRP with real heterogeneous fleet and time windows based on ant colony optimization: an industrial case study. In *Proceedings of the 15th ACM SIGSPATIAL International Workshop on Computational Transportation Science*. 1–4.

[9] John Hickman, Dieter Hassel, Robert Joumard, Zissis Samaras, and S Sorenson. 1999. Methodology for calculating transport emissions and energy consumption. (1999).

[10] LI Jingjing, Yaohuiqiong Fang, and Na Tang. 2022. A cluster-based optimization framework for vehicle routing problem with workload balance. *Computers & Industrial Engineering* 169 (2022), 108221.

[11] Yong-Ju Kwon, Young-Jae Choi, and Dong-Ho Lee. 2013. Heterogeneous fixed fleet vehicle routing considering carbon emission. *Transportation Research Part D: Transport and Environment* 23 (2013), 81–89.

[12] Phan Nguyen Ky Phuc and Nguyen Le Phuong Thao. 2021. Ant colony optimization for multiple pickup and multiple delivery vehicle routing problem with time window and heterogeneous fleets. *Logistics* 5, 2 (2021), 28.

[13] Silvia Mazzeo and Irene Loiseau. 2004. An ant colony algorithm for the capacitated vehicle routing. *Electronic Notes in Discrete Mathematics* 18 (2004), 181–186.

[14] Arash Mozhdehi, Mahdi Mohammadizadeh, and Xin Wang. 2024. Edge-DIRECT: A Deep Reinforcement Learning-based Method for Solving Heterogeneous Electric Vehicle Routing Problem with Time Window Constraints. *Proceedings of the Canadian Conference on Artificial Intelligence* (may 27 2024). https://caiac.pubpub.org/pub/vlg4rwhi.

[15] Arash Mozhdehi, Mahdi Mohammadizadeh, Yunli Wang, Sun Sun, and Xin Wang. 2024. EFECTIW-ROTER: Deep Reinforcement Learning Approach for Solving Heterogeneous Fleet and Demand Vehicle Routing Problem with Time-Window Constraints. In *Proceedings of the 32th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '24)* (Atlanta, GA, USA). https://doi.org/10.1145/3678717.3691208

[16] Binbin Pan, Zhenzhen Zhang, and Andrew Lim. 2021. Multi-trip time-dependent vehicle routing problem with time windows. *European Journal of Operational Research* 291, 1 (2021), 218–231.

[17] Martin WP Savelsbergh. 1985. Local search in routing problems with time windows. *Annals of Operations research* 4 (1985), 285–305.

[18] Hongguang Wu, Yuelin Gao, Wanting Wang, and Ziyu Zhang. 2021. A hybrid ant colony algorithm based on multiple strategies for the vehicle routing problem with time windows. *Complex & intelligent systems* (2021), 1–18.

[19] Bin Yu, Zhong-Zhen Yang, and Baozhen Yao. 2009. An improved ant colony optimization for vehicle routing problem. *European journal of operational research* 196, 1 (2009), 171–176.