Article

An Iterated Local Search Heuristic for the Multi-Trip Vehicle Routing Problem with Multiple Time Windows

Yinghui Wu *, Haoran Du and Huixin Song

School of Economics and Management, Jiangsu University of Science and Technology, Zhenjiang 212100, China; 221210405114@stu.just.edu.cn (H.D.); 211110401205@stu.just.edu.cn (H.S.)
* Correspondence: wyh@just.edu.cn

Abstract: This paper studies the multi-trip vehicle routing problem with multiple time windows, which extends the multi-trip vehicle routing problem by deciding not only the sequence of customers that each vehicle serves but also the service time window of each customer. It also requires that the delivery service time is within the selected time windows and that the total demand of the customers served by the vehicle on each trip does not exceed the maximum carrying capacity. For solving the studied problem, we develop a mixed integer linear programming model with the objective of minimizing the total travel distance of vehicles and design a tailored iterative local search heuristic. Within the framework of the iterative local search, an improved Solomon greedy insertion algorithm suitable for multiple time windows and multi-trip scenarios is designed to generate the initial solution, and local search operators such as Or-opt and Relocate, as well as Random Exchange perturbation operations, are also developed. The experiment results demonstrate the effectiveness of the proposed model and algorithm and confirm that by providing customers with multiple time windows option, carriers can flexibly plan vehicle routes and select appropriate service time windows, thereby reducing the number of vehicles used and the total distance travelled and improve delivery success.

Keywords: multi-trip vehicle routing problem; multiple time windows; mixed integer programming; iterated local search

MSC: 90-10

check for updates

Citation: Wu, Y.; Du, H.; Song, H. An Iterated Local Search Heuristic for the Multi-Trip Vehicle Routing Problem with Multiple Time Windows. *Mathematics* **2024**, *12*, 1712. https://doi.org/10.3390/math12111712

Academic Editors: David Canca and Gabriel Villa

Received: 6 May 2024
Revised: 25 May 2024
Accepted: 29 May 2024
Published: 30 May 2024

1. Introduction

According to statistics from the State Post Bureau of China, as of the end of October 2023, the cumulative volume of express parcels in China reached 105.17 billion, representing a year-on-year increase of 17.0%. This growth has led to a continuous rise in demand for last-mile delivery of express parcels [1]. Effective vehicle route planning is crucial for logistics companies to enhance delivery efficiency and user experience and to develop sustainable urban logistics. The vehicle routing problem (VRP) has been a hot topic of concern in the academic community since introduced by Dantzig and Ramser [2]. The VRP is typically defined as determining the optimal routes for a set of vehicles to visit customers, effectively meeting a series of customer demands while satisfying constraints such as service time windows and vehicle capacity. The objective of this problem is usually to minimize total travel costs, such as distance or time. Solutions to the VRP need to consider multiple aspects, including route planning, vehicle allocation, time window management, and delivery sequence. Over the decades, the VRP has generated numerous extensions and variations, including the VRP with time windows (VRPTW), the heterogeneous fleet VRPTW, the dynamic VRPTW, the simultaneous pickup and delivery VRPTW, and the electric VRPTW, among others [3–7].

In China, to relieve traffic congestion in urban central areas and reduce pollution, restrictions imposed by urban transportation departments limit the access of large freight vehicles to city center roads. Consequently, carriers often rely on small electric delivery trucks to complete the last-mile delivery. The traditional VRP and VRPTW and their variants assume that each vehicle can only execute one trip. In reality, due to the limited capacity of small delivery trucks, their utilization rate is low. To improve vehicle utilization and reduce operational costs, multiple return trips are often scheduled for deliveries when the fleet size is limited. Thus, the VRP and VRPTW have been extended to the multi-trip VRP (MTVRP) and MTVRPTW, respectively. In the context of last-mile delivery, the MTVRP and MTVRPTW provide higher practical value. As a result, the MTVRP and its variants have gradually attracted the attention of scholars in recent years [8].

As the concept of customer-centricity becomes increasingly ingrained in socioeconomic activities, logistics companies are placing greater emphasis on improving the quality and timeliness of last-mile delivery services. Many logistics companies have introduced time window services for customers, requiring carriers to provide services within specified time windows. The VRPTW and MTVRPTW assume that each customer has only one time window option. However, in practical last-mile delivery operations, the concentration of order demands often leads to a limited flexibility in carrier vehicle routing, resulting in carriers being unable to ensure timely deliveries to all customers within their designated time windows. The establishment of a single time window for each customer may actually lead to a decrease in customer satisfaction. To meet the demands of large-scale deliveries and increase flexibility in planning delivery routes, while avoiding the issue of excessively concentrated delivery requests within a single time window, carriers have introduced multiple time windows options. This allows each customer to select from multiple acceptable delivery time windows. Carriers optimize vehicle delivery routes and visit times for customers based on order demands, customer addresses, and customer time windows and then inform customers of the delivery times. This approach not only increases the flexibility of carrier vehicle routing planning but also enhances the success rate of deliveries, thereby reducing logistics transportation costs and improving customer satisfaction.

Therefore, in this paper, we consider the MTVRP with multiple time windows (MTVR PMTW), where the decision-making involves determining the sequence of serving customers for each vehicle, establishing service time windows for each customer, ensuring delivery times align with specified time windows, and meeting constraints such as the total demand served by each vehicle on each route, not exceeding the maximum vehicle capacity. By extending traditional VRPTW models to incorporate factors such as multi-trip, logistics companies can enhance the utilization of small delivery trucks and reduce operational costs. This optimization leads to fewer vehicle miles traveled, lower fuel consumption, and reduced emissions per delivery, thus mitigating the environmental impact of urban logistics operations. Moreover, the introduction of multiple time window options enables carriers to distribute delivery demands more evenly throughout the day, thereby alleviating peak-hour congestion and reducing the overall carbon footprint of last-mile delivery services. Additionally, by improving delivery efficiency and customer satisfaction, these optimizations contribute to the long-term viability and competitiveness of logistics businesses, fostering economic sustainability within the industry.

The main contributions of this paper are as follows: (1) formally defining the MTVRP MTW and formulating the problem as a mixed-integer linear programming (MILP) model to minimize the total vehicle travel distance; (2) designing a tailored iterated local search (ILS) heuristic for solving practical-scale instances of the MTVRPMTW, within which an improved Solomon greedy insertion algorithm is proposed to generate the initial solution suitable for multiple time windows and multiple route scenarios, along with the design of local search operators such as Or-opt and Relocate and Random Exchange perturbation operations; (3) conducting extensive computational experiments to demonstrate the effectiveness of the proposed model and algorithm.

The paper is organized as follows: Section 2 reviews related research on MTVRP and its variants. In Section 3, we define the MTVRPMTW and provide a MILP model for it. Section 4 describes the proposed ILS heuristic for solving the MTVRPMTW. In Section 5, we present and discuss the computational results. Section 6 concludes the paper and provides future research directions.

2. Literature Review

In recent years, there has been a growing body of research on variations of the MTVRP. For a comprehensive overview of the MTVRP and its variants, as well as the solving algorithms, it is recommended to refer to the study by Cattaruzza et al. [8].

François et al. [9] explore an MTVRPTW, whereby they design an integrated solution method when time windows are involved. Additionally, they examine two objective functions: total duration and travel times; minimizing the total travel time can lead to impractical increases in waiting time. Sethanan et al. [10] introduced an integer linear programming formulation and a hybrid differential evolution algorithm incorporating a genetic operator with a fuzzy logic controller to tackle the MTVRP with backhauls and a diverse fleet. Coelho et al. [11] also studied the MTVRP with backhauls and a diverse fleet, incorporating docking constraints where certain vehicles cannot serve specific customers. Their objective function reflects realistic costs, including fixed and distance-based vehicle costs, along with a cost per customer visited. They proposed a trajectory search heuristic named GILS-VND, and they proved that their method obtains competitive solutions and improves the company solutions leading to significant savings in transportation costs. Cattaruzza et al. [12] study the MTVRPTW, considering release dates (the release date represents the date when their requested merchandise becomes available at the depot). They proposed an approach that tackles this problem using a population-based algorithm, employing a giant tour representation for individuals. Zhen et al. [13] extended the MTVRPTW with release dates to the multi-depot version and formulated this as a MILP model. A hybrid particle swarm optimization algorithm and a hybrid genetic algorithm are developed to solve this problem. Pan et al. [14] studied the multi-trip time-dependent VRP with time windows (MT-TDVRPTW); they designed a hybrid meta-heuristic algorithm to solve the problem, leveraging the adaptive large neighborhood search for guided exploration and the variable neighborhood descend for intensive exploitation. Grangier et al. [15] studied the two-echelon MTVRP, incorporating constraints typical in city logistics such as time window constraints, synchronization constraints, and multiple trips at the secondary level. They proposed an adaptive large neighborhood search method to tackle this problem. Recently, Yang [16] proposed an exact price-cut-and-enumerate method (EPCEM). The EPCEM can be used to solve the capacitated MTVRPTW and its four variants: the CMTVRPTW with loading times, the CMTVRPTW with limited trip duration, the CMTVRPTW with release dates, and the drone routing problem. They show that the EPCEM significantly outperformed the state-of-the-art exact method through extensive numerical experiments. However, there is no literature focusing on the MTVRPMTW and its solution methods.

There have been several studies on the VRP considering multiple time windows (VRPMTW). For example, Ibaraki et al. [17] proposed local search algorithms for the VRPMTW, treating each customer's time window constraint as a penalty function, and efficiently minimizing total penalties through dynamic programming. Beheshti et al. [18] tackled the multi-objective VRP with multiple prioritized time windows, present a mathematical model for this problem, and proposed a cooperative coevolutionary multi-objective quantum genetic algorithm to address it, with a new local search strategy. Belhaiza et al. [19] introduced a hybrid variable neighborhood tabu search heuristic for the VRPMTW. Additionally, they presented a minimum backward time slack algorithm designed for a multiple time windows environment, which adjusts arrival and departure times based on recorded waiting time and delay. Schaap et al. [20] focused on the VRPMTW, aiming to optimize routes to ensure customer satisfaction within specified time windows. Their approach utilizes a large neighborhood search metaheuristic with dynamic programming to select optimal time windows for each customer, complemented by computationally efficient move descriptors. Although the works described above studied the VRP with the consideration of multiple time windows, they allowed the vehicles to perform only one trip during the planning horizon.

3. Problem Description and Formulation

In this section, we formally define the MTVRPMTW and formulate this problem as a MILP model.

3.1. Problem Description

The MTVRPMTW can be formally defined on a directed graph $G = (V, A)$, where $V = N \cup \{0, |N| + 1\}$ is the set of vertices, and $A = \{(i, j)|i, j \in N, i \neq j, i \neq |N| + 1, j \neq 0\}$ is the set of arcs. $N = \{1, 2, \ldots, n\}$ represent n customers, 0 represents the depot, and $|N| + 1$ represents a duplicate of the depot. Each customer $i \in N$ has a service time s_i and a known demand q_i. Unlike in many general VRPs, each customer $i \in N$ has m_i time windows represented by $TW_i = \{[e_i^1, l_i^1], \ldots, [e_i^{m_i}, l_i^{m_i}]\}$, where m_i denotes the number of time window of customer i. If the selected time window for serving customer i is $\left[e_i^p, l_i^p\right]$, the vehicle must wait until time e_i^p to serve customer i if it arrives at customer i earlier than e_i^p, resulting in a waiting time w_i^{kr}. The vehicle is not allowed to serve customer i after l_i^p. Each arc $(i, j) \in A$ is associated with a travel time $t_{i,j}$ and a travel distance $d_{i,j}$, both of which satisfy the triangle inequality. Let K denote a fleet of homogeneous vehicles with capacity Q. A trip is defined as a sequence of vertices visits that starts from the depot, visits a sequence of customers within any of time window, and returns to the depot. A tour is defined as a sequence of trips performed by the same vehicle chronologically. Let $R = \{1, 2, \ldots, r_{UB}\}$ represent the set of trips, where r_{UB} is an upper bound on the number of trips. Each vehicle is allowed to perform several trips that do not overlap in time (called a tour) within the period of schedule $[0, T]$, that is, the time window of the depot. To describe this case clearly, Figure 1 illustrates an example of an MTVRPMTW with 13 customers. Each customer has 2–3 time windows. Vehicles V1 and V2 each execute two trips. The trip for vehicle V1 is 0→1→8→3→7→0→9→11→4→0, and for vehicle V2, it is 0→2→13→10→0→6→12→5→0. The time windows for serving customers are indicated by bold black lines.

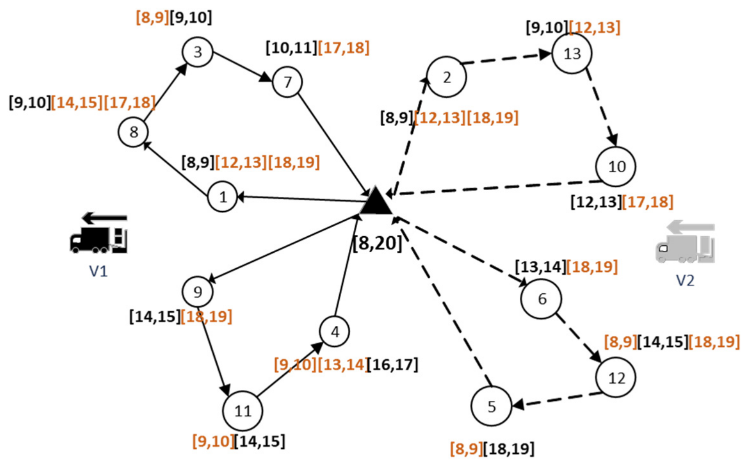

Figure 1. A schematic diagram of the MTVRPMTW.

The following assumptions for the MTVRPMTW are considered:

- Each customer is visited once by a vehicle in a trip;
- The total demand in each trip does not exceed the vehicle capacity Q;
- The times of different trips of the same vehicle do not overlap;
- The selected service time windows are accepted by the customers;
- Each trip of a vehicle starts from the depot, visits the required customer vertices according to the planned route, and returns to the depot;

- The end time of the last trip of each vehicle does not exceed time T.

3.2. Model Formulation

We first present the notations used in this paper in Table 1.

$$Min\ Z = \sum_{(i,j)\in A} d_{ij} \sum_{k\in K} \sum_{r\in R} x_{ij}^{kr} \tag{1}$$

$$\sum_{k\in K} \sum_{r\in R} y_i^{kr} = 1, \forall i \in N \tag{2}$$

$$\sum_{j\in N\backslash\{i\}} x_{ij}^{kr} = \sum_{j\in N\backslash\{i\}} x_{ji}^{kr} = y_i^{kr}, \forall i \in N \cup \{0\}, k \in K, r \in R \tag{3}$$

$$\sum_{i\in N} q_i y_i^{kr} \le Q, \forall k \in K, r \in R \tag{4}$$

$$a_i^{kr} + s_i + t_{ij} \le a_j^{kr} + M\left(1 - x_{ij}^{kr}\right), \forall i \in N \cup \{0\}, j \in N, k \in K, r \in R \tag{5}$$

$$a_i^{kr} + s_i + t_{i0} \le a_{N+1}^{kr} + M\left(1 - x_{i0}^{kr}\right), \forall i \in N, k \in K, r \in R \tag{6}$$

$$a_{N+1}^{kr} \le a_0^{kr+1}, \forall k \in K, r \in R\backslash\{n\} \tag{7}$$

$$a_{N+1}^{kr} < T, \forall k \in K, r \in R \tag{8}$$

$$\sum_{p\in\{1,2,\ldots,m_i\}} u_{ip}^{kr} = y_i^{kr}, \forall i \in N, k \in K, r \in R \tag{9}$$

$$e_i^p u_{ip}^{kr} \le a_i^{kr} + w_i^{kr}, \forall i \in N \cup \{0\}, p \in \{1,2,\ldots,m_i\}, k \in K, r \in R \tag{10}$$

$$a_i^{kr} + w_i^{kr} + s_i - T\left(1 - u_{ip}^{kr}\right) \le l_i^p u_{ip}^{kr}, \forall i \in N, p \in \{1,3,\ldots,m_i\}, k \in K, r \in R \tag{11}$$

$$x_{ij}^{kr} \in \{0,1\}, \forall k \in K, r \in R, i \in N \cup \{0\}, j \in N \cup \{0\} \tag{12}$$

$$y_i^{kr} \in \{0,1\}, \forall k \in K, r \in R, i \in N \cup \{0\} \tag{13}$$

$$a_i^{kr} \ge 0, \forall k \in K, r \in R, i \in N \cup \{0, |N|+1\} \tag{14}$$

$$u_{ip}^{kr} \in \{0,1\}, \forall k \in K, r \in R, i \in N, p \in \{1,2,\ldots,m_i\} \tag{15}$$

$$w_i^{kr} \ge 0, \forall k \in K, r \in R, i \in N \cup \{0\} \tag{16}$$

The objective function (1) minimizes the total distance of vehicles. Constraint (2) ensure that each customer is serviced by exactly one vehicle in one trip. Constraint (3) represents the flow balance constraint, stating that when a customer $i \in N$ is serviced by vehicle k in trip r, the number of arcs entering customer point i equals the number of arcs leaving it. When i represents the depot 0 or $|N|+1$, the number of arcs entering the depot equals the number of arcs leaving it. Constraint (4) ensures that the total demand of serviced customers in a single trip of each vehicle does not exceed the vehicle's capacity Q. Constraints (5)–(6) indicate that when vehicle k traverses i and j in trip r, i.e., $x_{ij}^{kr} = 1$, and also indicate the relationship of arrival time at j and i. Constraint (7) indicates that for any vehicle k, the arrival time at the depot at the end of the previous trip r must be less than or equal to the departure time from the depot at the beginning of the subsequent trip $r+1$. Constraint (8) ensures that the end time of each vehicle's single trip does not exceed T. Constraint (9) represents that when vehicle k services customer i in trip r, it must do so within only one of the available time windows. Constraints (10) and (11) ensure that vehicles must provide a service to customer i within one of its time windows $\left[e_i^p, l_i^p\right]$; if a vehicle arrives at customer i earlier than e_i^p, it must wait until e_i^p to begin service, with the waiting time denoted as w_i^{kr}; otherwise, the waiting time is 0, and the vehicle is not allowed to service customer i after time l_i^p. Constraints (12)–(16) define the feasible range of decision variables x_{ij}^{kr}, y_i^{kr}, a_i^{kr}, u_{ip}^{kr}, and w_i^{kr}.

Table 1. The notations used in this paper.

Sets and indexes	
$N \cup \{0, \|N\|+1\}$	The set of nodes. Node 0 represents the depot, $\|N\|+1$ represents a duplicate of the depot, $N = \{1, 2, \ldots, n\}$ represents the customers
R	The set of trips, $R = \{1, 2, \ldots, r_{UB}\}$ where r_{UB} is an upper bound on the number of trips (e.g., $r_{UB} = n$)
A	The set of arcs $(i,\ j)$, $A = \{(i,\ j) \| i, j \in N\}$
K	The set of vehicles, $K = \{1, 2, \ldots, K\}$
TW_i	The set of time windows of each customer $i \in N, TW_i = \left\{\left[e_i^1, l_i^1\right], \ldots, \left[e_i^{m_i}, l_i^{m_i}\right]\right\}$ where m_i is the number of time windows of customer i
Parameters	
q_i	Demand of each customer $i \in N$
t_{ij}	Travel time t_{ij} is associated with arc $(i,\ j)$
d_{ij}	Travel distance d_{ij} is associated with arc $(i,\ j)$
s_i	Service time of each customer $i \in N$
$\left[e_i^p, l_i^p\right]$	The pth time window of customer $i \in N$
m_i	Number of time windows of customer $i \in N$
Q	Capacity of each vehicle
$[0, T]$	Time window of the depot
M	A big positive number
Decision variables	
a_i^{kr}	Continuous variable, indicates the time at which trip r of vehicle k visits nodes $i \in N$, a_0^{kr}(resp. a_{N+1}^{kr}) is the time at which the route r starts (resp. ends) at the depot
x_{ij}^{kr}	Binary variable, if trip r of vehicle k travels through arc $(i,\ j)$, $x_{ij}^{kr} = 1$; otherwise, $x_{ij}^{kr} = 0$
y_i^{kr}	Binary variable, if trip r of vehicle k visits vertex i, $y_i^{kr} = 1$; otherwise, $y_i^{kr} = 0$
u_{ip}^{kr}	Binary variable, if trip r of vehicle k visits vertex i in pth time window $u_{ip}^{kr} = 1$; otherwise, $u_{ip}^{kr} = 0$
w_i^{kr}	The waiting time of vehicle k at customer i in trip r

4. ILS Heuristic for the MTVRPMTW

The MTVRPMTW is an extension of the MTVRPTW. Therefore, the MTVRPMTW studied in this paper is an NP-hard problem. For small-scale instances of this problem, optimal solutions can be obtained using optimization software such as CPLEX or Gurobi. However, for large-scale instances, CPLEX or Gurobi often struggle to find exact solutions. Hence, an ILS algorithm is proposed in this paper to address the MTVRPMTW. ILS is a simple, robust, and efficient metaheuristic algorithm [21]. Its basic idea is to start from an initialization and optimize it using local search methods. Subsequently, the optimized solution is perturbed to escape from the current local optimum and continue with the local search. This process is repeated multiple times, with each iteration using the solution obtained from the previous search as the new starting point, until a stopping condition is met. The ILS algorithm has been successfully applied to various combinatorial optimization problems, such as the traveling salesman problem [22–24] and the VRP and its variants [25–29].

Based on the ILS algorithm framework, this paper designs a tailored ILS heuristic to solve the MTVRPMTW, with the following specific steps:

Step 1: Improved Solomon greedy insertion algorithm to generate initialization.

Step 2: Before the termination criteria are met:

Local search: Perform local search on the current solution using Or-opt and Relocate operators. Or-opt optimizes the total travel distance of vehicles, producing a locally optimal solution, while Relocate reduces the number of vehicles based on the locally optimal solution.

Perturbation: When the current solution is trapped in a local optimum, perturb the solution using Random Exchange to escape the current local optimum.

Termination criteria: This paper adopts the number of iterations as the termination condition; that is, when the number of iterations exceeds a given threshold, the algorithm terminates and outputs the current best solution obtained.

4.1. Generating Initial Solution by Improved Solomon Greedy Insertion Algorithm

This section focuses on the characteristics of the MTVRPMTW and improves the Solomon insertion algorithm [30] for generating the initial solution to the MTVRPMTW. The steps of the improved Solomon greedy insertion algorithm are as follows:

Step 1: Initializing Customer Time Windows

Each customer has multiple time windows, so we default to use the first time window when constructing the initial solution.

Step 2: Initializing Vehicle Routes

In the initial solution generation phase, we first select a seed customer among all the customers as the first customer to be inserted into the path, the point corresponding to the seed customer is called the seed node, and the customer furthest away from the warehouse point is preferred when selecting the seed customer. This step initializes a vehicle route containing the depot, the delivery location of the seed customer, and a copy of the depot for the return trip.

Step 3: Route Planning

Starting from the seed node, construct vehicle routes using a greedy strategy. The specific steps are as follows:

Depart from the seed node and, while ensuring the vehicle load capacity is met, determine the optimal customer point and its corresponding time window to insert into the current route, based on the existing route and unvisited customers. This follows two rules:

Rule 1: Determine the optimal insertion positions $i(u)$ and $j(u)$ for each unvisited customer point u in the current route based on Equations (17)–(20) and select the best time window at the current insertion position. Equation (20) represents the time difference when the start of service time for the subsequent node is delayed after inserting point u, where b_j and b_{ju} denote the start of service time for the subsequent node before and after inserting point u, respectively. Equation (19) represents the increase in travel distance when inserting point u, where t_{ij} denotes the travel time between nodes i and j, and μ represents the weight coefficient for the distance between $i(u)$ and $j(u)$. Equation (18) is the weighted sum of the increase in travel time and the delay in service time, representing the time cost incurred when inserting customer point u, where α_1 and α_2 represent the weight coefficients for these two indicators. Equation (17) minimizes the time cost when inserting customer u, aiming to minimize travel distance. By sequentially attempting to insert between two nodes in the existing route, while ensuring the vehicle load capacity is met and no time window constraints are violated, determine the optimal positions $i(u)$ and $j(u)$ for inserting customer u to minimize the time cost, as well as the selected time window at the optimal position, where ρ represents the index of points in the existing route and m represents the number of nodes in the existing route (including the depot copy).

$$c_1(i(u), u, j(u)) = min\{c_1(i_{\rho-1}, u, j_\rho)\}, \rho = 1, 2, \ldots, m \tag{17}$$

$$c_1(i, u, j) = \alpha_1 c_{11}(i, u, j) + \alpha_2 c_{12}(i, u, j), \alpha_1 + \alpha_2 = 1; \alpha_1 \geq 0, \alpha_2 \geq 0 \tag{18}$$

$$c_{11}(i, u, j) = d_{iu} + d_{uj} - \mu d_{ij}, \mu \geq 0 \tag{19}$$

$$c_{12}(i, u, j) = b_{ju} - b_j \tag{20}$$

Rule 2: When inserting a customer point, in addition to considering the time cost incurred by Equation (18) in Rule 1, this rule also takes into account the waiting time generated by inserting customer point u. It selects a time window that is closest to the arrival time and does not violate the time window constraint (i.e., no lateness is allowed). Equation (22) incorporates both the waiting time generated by inserting customer point u and the time cost generated by inserting u in Rule 1. w_{ui} represents the waiting time of the vehicle at customer point u, and λ represents the weight coefficient for the waiting time, where ($\lambda > 0$). Based on Equations (21) and (22), the customer point u^* that generates the lowest total time cost among all uninserted u points is selected, i.e., the best customer point to insert in the current route, and the optimal insertion positions $i(u)$ and $j(u)$ are determined according to Rule 1.

$$c_2(i(u^*), u, j(u^*)) = max\{c_2(i(u), u, j(u))\} \tag{21}$$

$$c_2(i(u), u, j(u)) = -\lambda w_{ui} - c_1(i, u, j) \tag{22}$$

Based on the aforementioned rules, the optimal customer point is selected from the unvisited customer points, and it is inserted at its optimal insertion position. During the insertion process, the vehicle's load, arrival time, and the list of selected time windows are continuously updated to ensure compliance with vehicle load constraints and that no customer time windows are violated, thus constructing vehicle routes that meet the requirements. When there are no optimal customer points left for insertion, a virtual depot is inserted, initiating a new trip until no further trips can be added. At this point, a new vehicle is added, and the above process is repeated until all customers have been visited.

The coding of the solution representation of Figure 1 for the MTVRPMTW problem is shown in Figure 2. The first layer is represented as the order in which each vehicle visits the customer, and the second layer is represented as the time window in which the customer is visited.

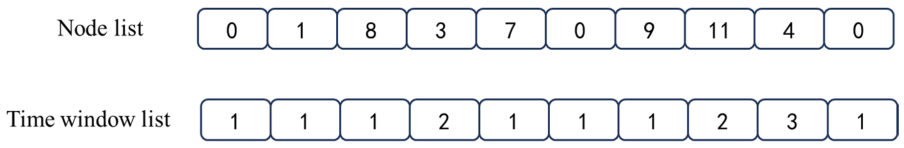

Figure 2. The coding of the solution representation.

4.2. Local Search and Perturbation Operators

According to the characteristics of the MTVRPMTW, this section designs two local search operators, Or-opt [31] and Relocate, and one perturbation operator, Random Exchange.

Or-opt Operator: This operator removes three arcs from the same route (including multiple trips), where the three initial nodes are sequentially connected to the end node of another route, forming new three arcs.

In Figure 3, for each r in the initialization, where r represents the route of vehicle v. Within route r, "△" denotes the depot, and "○" denotes the customer. Specifically, $i-1$, i, $i+1$, $i+2$, j, and $j+1$ represent the corresponding six customers, x_1, x_2, y_1, y_2, z_1, and z_2, in the route. In each iteration, we update the solution through the following steps. First, ensure that the number of customer points between y_1 and x_1 reaches a predetermined value. Second, ensure that z_1 is not located between x_1 and y_1. Then, we remove arcs (x_1, x_2), (y_1, y_2), and (z_1, z_2) and add arcs (x_1, y_2), (y_1, z_2), and (z_1, x_2), thus forming a new vehicle route. Next, based on the arrival time at customer points, we reselect the customer time windows and check whether the vehicle load and arrival time of the vehicle route meet the requirements, thereby determining whether a local candidate solution is generated. Finally, aiming to minimize the total travel distance of the vehicles, we select the best solution from the candidate solution set for the relocation operation.

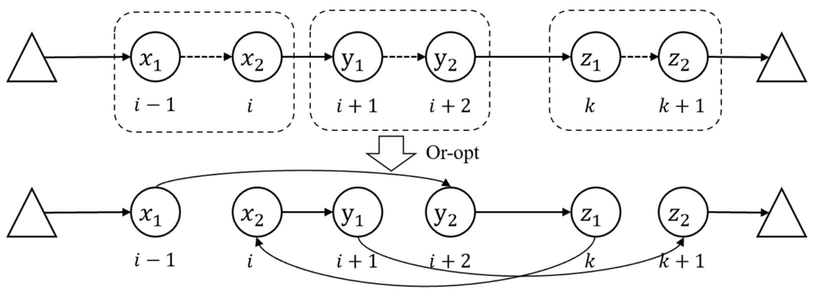

Figure 3. An example of Or-opt operator.

Relocate Operator: Under the premise of ensuring that vehicle load and time windows do not violate constraints, this operator sequentially embeds customer points from one path into another path.

As illustrated in Figure 4, paths R_1 and R_2 are paths from different vehicles.

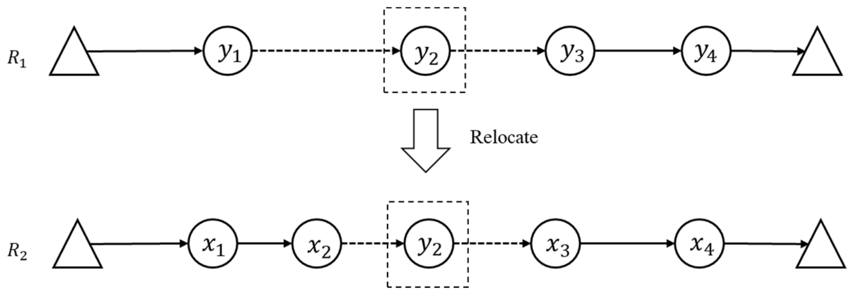

Figure 4. An example of Relocate Operator.

Within route R_1 and R_2, "△" denotes the depot, and "○" denotes the customer. Specifically, y_1, y_2, y_3, and y_4 represent the four customers served by route R_1. x_1, x_2, x_3, and x_4 represent the four customers served by route R_2. In each iteration, we update the solution through the following steps. Relocate involves that the customer point y_2 in R_1 and is then embedded between the customer points x_2 and x_3 of R_2. We remove each customer point in each vehicle route in turn, try to insert it into the vehicle path of another vehicle, record the reduction of the total vehicle travel time before and after the operation for each feasible solution, and perform the Relocate operation on the vehicle route with the lowest reduction after the vehicle routing operation for all vehicles. For the embedded customer points, the operator selects the time window closest to the deadline (i.e., with the shortest waiting time) based on the arrival time. It then checks whether the vehicle path satisfies the load and time window constraints, thereby determining whether a local candidate solution is generated. The purpose of this operator is to reduce the number of vehicles. When the number of vehicles decreases, the solution is moved; otherwise, it is discarded.

When the algorithm becomes trapped in a local optimum, it is necessary to escape from it by introducing perturbations. In this paper, we employ the Random Exchange method for perturbation.

Random Exchange Perturbation: This operation involves randomly swapping two customer points between two paths. First, two vehicle paths, R_1 for vehicle V_1 and R_2 for vehicle V_2, are randomly selected. Next, two customer points, u_1 from R_1 and u_2 from R_2, are randomly chosen for swapping. Subsequently, the algorithm checks whether the vehicle load and time windows are violated and determines whether a feasible solution is produced.

4.3. The ILS Heuristic Procedure

Algorithm 1 illustrates the whole procedure of the ILS heuristic for solving the MTVRP MTW. Initially, the algorithm generates an initial solution using the improved Solomon algorithm, as depicted in steps 1–3. The local search phase is outlined in steps 4–11, where the Or-opt operator is utilized to optimize vehicle routing paths, and the Relocate operator is employed to optimize vehicle usage. We perform the Or-opt operator on the vehicle path of each vehicle in turn, record the feasible solution of the total vehicle travel path reduction to form a candidate solution set, and then select the optimal solution in all candidate solution sets for the movement of Relocate. After the Relocate is executed on the customer point of each vehicle in the solution, choosing the move with the most improvement in the total travel time of the vehicle, we then find the local optimal solution. In cases where the local search becomes trapped in a local optimum, perturbation operations are executed, as described in steps 12–16. Subsequently, a subset of solutions with the minimum number of vehicles from the local optimal solution set is selected. The Or-opt operator is then applied again to optimize the routing paths of solutions within this subset. Finally, the best solution with the minimum total travel distance from the selected subset is returned, as demonstrated in steps 17–23.

Algorithm 1: The ILS heuristic for the MTVRPMTW

```
Input: Instance of the MTVRPMTW, Maximum iteration number $TT$, Maximum perturbation time $G$
Output: Customer time window selection, vehicle routes, Total vehicle travel distance
1:  Improved Solomon insertion algorithm generates initial solution $s_0$//see Section 4.1
2:  Local optimal solution set $S$, $S \leftarrow \{\}$
3:  The best solution $s^* \leftarrow s_0$, Iteration count $t \leftarrow 0$, Perturbation count $g \leftarrow 0$
4:  while $t < TT$ and $g < G$ do
5:      $t \leftarrow t+1$, $g \leftarrow g+1$
6:      Update $s^* \leftarrow$ Or-opt ($s_0$)//see Section 4.2
7:      Update $S \leftarrow S \cup s^*$
8:      $s' \leftarrow$ Relocate ($s^*$)//see Section 4.2
9:      if vehiclenumberof $s'$ < vehiclenumberof $s^*$ then
10:         $s^* \leftarrow s'$, initialize $S \leftarrow \{\}$, initialize $t \leftarrow 0$
11:         continue
12:     else
13:         Perturbation operation: Random exchange ($s'$)//see Section 4.2
14:         Accept all perturbation solution $s^* \leftarrow s'$
15:     end if
16: end while
17: $s_v \leftarrow$ The minimum number of vehicles from set $S$
18: $S^* \leftarrow \{s_i \in S \mid \text{vehiclenumberof}(s_i) = \min(\text{vehiclenumber})\, s_v\}$
19: for $s$ in $S^*$ do
20: Optimizing vehicle routing: Or-opt ($s$)//see Section 4.2
21: end for
22: $s^* \leftarrow$ solution in $S^*$ with shortest travel distance
23: Return $s^*$
```

5. Numerical Experiments

This section validates the effectiveness of the proposed MTVRPMTW model and ILS heuristic through case study experiments. Solomon "2" class datasets with "C", "R", and "RC" types of instances, characterized by dispersed customer distribution resembling real-world scenarios, are selected for this purpose. Since a wide time window would cover the entire working day, and the time windows may overlap, we generate the instances with non-overlapped time windows. Specifically, 14 case studies with non-overlapped time windows are chosen, including C201, C205-C208, R201, R205, R209, R211, RC201, and RC205–RC208. Additionally, 1–2 time windows, aligned with the characteristics of Solomon instances, are added to each customer in these cases, resulting in new two different scales of instances whose names starting with the letters "P" and "L", respectively. For example, instance PC201 is generated from Solomon instance C201 with the first 10 customers by adding additional time windows and changing other parameters, while LC201 has all 100 customers of C201.

For the generation of multiple time windows, we count the width of the time window of the original Solomons instances. If the width of the time window is greater than 1/3 of the planning period, the instance is discarded because the time windows will overlap if the time windows are generated. The time windows of the selected instances are divided into two categories, one has a fixed-width time window, and the other has a non-fixed-width time window. If the first category is based on the start time and end time of the original time window, we randomly generate a time window with the same width as the original time window to ensure that it does not overlap with the original time window and does not exceed the working day time. For the second category, the time window between the maximum and minimum widths of the time window for the instances is randomly generated.

The basic parameters of these new instances are shown in Table 2. All case study experiments were conducted using the Java programming language within the Eclipse 4.24.0 development environment and utilizing CPLEX version 12.10. Additionally, all experiments were performed on a personal computer equipped with a configuration of a 2.60 GHz CPU and 32 GB RAM.

Table 2. The basic parameters of these new instances.

Parameter	Value
n	10, 100
Q	100, 200
T	960, 1000, 3390
m_i	2, 3

5.1. Comparison with the MTVRPTW

We compare the results of solving small instances using the MTVRPMTW model with those using the MTVRPTW model to validate the effectiveness of considering multiple time windows. The model for the MTVRPTW is as follows:

$$Min\ Z = \sum_{(i,j)\in A} d_{ij} \sum_{k\in K} \sum_{r\in R} x_{ij}^{kr} \tag{23}$$

$$e_i y_i^{kr} \le a_i^{kr} + w_i^{kr}, \forall i \in N\{0\}, k \in K, r \in R \tag{24}$$

$$a_i^{kr} + w_i^{kr} + s_i - T\left(1 - y_i^{kr}\right) \le l_i y_i^{kr}, \forall i \in N, k \in K, r \in R \tag{25}$$

Constraints: (2)–(8), (12)–(14), (16).

Comparatively, in the MTVRPTW model, the variable u_{ip}^{kr} is not present, indicating the absence of considerations for selecting service time windows. In contrast to the constraints (9)–(11) in the MTVRPMTW model, the constraints (24)–(25) in the MTVRPTW model ensure that vehicles visit customers within their unique time windows.

Table 3 presents the results of solving small-scale instances using CPLEX under both MTVRPMTW and MTVRPTW models. It is worth noting that the instances solved under the MTVRPTW model have parameters identical to those solved under the MTVRPMTW model, except that each customer has only one time window. In Table 3, $Z_{MTVRPMTW}$ denotes the total vehicle travel distance for instances solved under the MTVRPMTW model, and $Z_{MTVRPTW}$ represents the total vehicle travel distance for instances solved under the MTVRPTW model. Δ_Z indicates the proportionate difference in total vehicle travel distance between the two models, calculated according to Equation (26). "AVG" represents the average total vehicle travel distance for the solved instances, as well as the average difference in total travel distance between the MTVRPMTW and MTVRPTW models.

$$\Delta_Z = \frac{Z_{MTVRPMTW} - Z_{MTVRPTW}}{Z_{MTVRPTW}} \times 100\% \tag{26}$$

Table 3. Computational results under the MTVRPMTW and MTVRPTW solved by CPLEX.

Instance	Q	n	m_i	K	$Z_{MTVRPMTW}$	$Z_{MTVRPTW}$	Δ_Z
PC201	100	10	3	1	151.8	195.7	−22.43%
PC205	100	10	3	1	151.8	185.7	−18.26%
PC206	100	10	3	1	151.8	185.7	−18.26%
PC207	100	10	3	1	151.8	185.7	−18.26%
PC208	100	10	3	1	151.8	185.7	−18.26%
AVG					151.8	187.7	−19.09%
PR201	100	10	3	1	194.2	254.3	−23.63%
PR205	100	10	2	1	194.2	222.7	−12.80%
PR209	100	10	2	1	194.2	202.3	−4.00%
PR211	100	10	2	1	194.2	194.2	0.00%
AVG					194.2	218.4	−10.11%
PRC201	200	10	3	1	165.9	218.5	−24.07%
PRC205	200	10	2	1	165.9	215.5	−23.02%
PRC206	200	10	2	1	165.9	211.6	−21.60%
PRC207	200	10	2	1	165.9	201.6	−17.71%
PRC208	200	10	2	1	165.9	165.9	0.00%
AVG					165.9	202.6	−17.28%

From Table 3, it can be observed that in the small instances comprising 14 sets of 10 customers, both the MTVRPMTW and MTVRPTW models yield exact solutions using the optimization solver CPLEX. In each instance, the number of vehicles used by thr MTVRPMTW and MTVRPTW are consistent. The total travel distance of the exact solution for the MTVRPMTW, considering multiple time windows, is consistently less than the total vehicle travel distance of the MTVRPTW, which considers only one time window. The travel distances for all instances within the given group are the same for the MTVRPMTW. This is because each group's instances have the same customers but with different time windows. Although the time windows for each customer have different widths, they satisfy the time window constraints for having the same optimal paths, resulting in their having the same travel distances. On average, for type "C" instances, this reduction amounts to 19.09% in total vehicle travel distance; the type "R" instances reduce the total vehicle travel distance by 10.11%; and the "RC" type instances reduce the total vehicle travel distance by 17.28%. Upon comparing the results of each instance set within the 14 groups, it is evident that the total vehicle travel distance can be reduced by up to 24.07%. In practical transportation operations, this translates directly to reduced transportation costs. Furthermore, it is observed that the results of solving identical types of instances in the MTVRPMTW are consistent. This consistency arises from the relaxation provided by the presence of multiple time windows, enabling better solutions when solving smaller-scale instances of the MTVRPMTW.

5.2. Performance of the ILS Heuristic

To validate the performance of the ILS heuristic in solving the MTVRPMTW problem, both small-scale and large-scale instances were addressed using the ILS heuristic. The large-scale instances involved 100 customers. Tables 4 and 5 present the results for two types of instances solved by the ILS heuristic for the MTVRPMTW problem. After several case experiments, we set TT and G for the study with 100 customers to be 2000 and 1000, respectively. In Table 4, Z_{ILS} and Z_{CPLEX}, respectively, denote the total vehicle travel distances for instances solved by the ILS heuristic and the optimization solver CPLEX, while t_{ILS} and t_{CPLEX} (in seconds) represent the time taken for solving the instances by the ILS heuristic and CPLEX, respectively. Δ_Z indicates the proportional difference in total vehicle travel distances between these two solutions, while Δ_t indicates the proportional difference in solution times between the two methods. The formulae for calculating Δ_Z and Δ_t are provided in Equations (27) and (28).

$$\Delta_Z = \frac{Z_{ILS} - Z_{CPLEX}}{Z_{CPLEX}} \times 100\% \tag{27}$$

$$\Delta_t = \frac{t_{ILS} - t_{CPLEX}}{t_{CPLEX}} \times 100\% \tag{28}$$

Table 4. Computational results comparing the ILS heuristic and CPLEX for solving the small-scale instances of MTVRPMTW.

Instance	***Q***	***n***	m_i	***K***	**CPLEX**		**ILS Heuristic**		Δ_Z	Δ_t
					Z_{CPLEX}	t_{CPLEX}	Z_{ILS}	t_{ILS}		
PC201	100	10	3	1	151.8	6.5	151.8	0.05	0.00%	−99.23%
PC205	100	10	3	1	151.8	13.3	151.8	0.08	0.00%	−99.40%
PC206	100	10	3	1	151.8	8.6	151.8	0.03	0.00%	−99.65%
PC207	100	10	3	1	151.8	6.2	152.2	0.05	0.26%	−99.19%
PC208	100	10	3	1	151.8	7.5	152.2	0.04	0.26%	−99.47%
PR201	100	10	3	1	194.2	0.6	194.2	0.03	0.00%	−95.00%
PR205	100	10	2	1	194.2	0.7	194.4	0.02	0.10%	−97.14%
PR209	100	10	2	1	194.2	0.4	194.4	0.02	0.10%	−95.00%
PR211	100	10	2	1	194.2	0.6	194.4	0.02	0.10%	−96.67%
PRC201	200	10	3	1	165.9	88.4	166.7	0.02	0.48%	−99.98%
PRC205	200	10	2	1	165.9	127.7	165.9	0.03	0.00%	−99.98%
PRC206	200	10	2	1	165.9	96.1	166.0	0.01	0.06%	−99.99%
PRC207	200	10	2	1	165.9	85.3	166.0	0.01	0.06%	−99.99%
PRC208	200	10	2	1	165.9	91.8	166.0	0.01	0.06%	−99.99%
AVG					169.0	38.1	176.0	0.04	0.11%	−98.62%

Table 5. Computational results on the large-scale instances of the MTVRPMTW solved by the ILS heuristic.

Instance	***Q***	***n***	m_i	**S-Insert**			**ILS-or**			**ILS-Relocate**			**ILS**		
				K	$Z_{\text{S-Insert}}$	$t_{\text{S-Insert}}$	K	$Z_{\text{ILS-or}}$	$t_{\text{ILS-or}}$	K	$Z_{\text{ILS-relocate}}$	$t_{\text{ILS-relocate}}$	K	Z_{ILS}	t_{ILS}
LC201	200	100	3	4	1966.5	19.1	4	1652.0	24.4	4	1600.3	10.2	4	1595.5	27.8
LC205	200	100	3	4	2187.9	30.4	4	1756.4	28.6	4	1784.9	13.0	4	1594.3	29.4
LC206	200	100	3	4	2022.2	27.5	4	1361.3	28.3	4	1539.3	14.7	4	1463.4	37.4
LC207	200	100	3	4	1928.0	28.6	4	1512.8	23.3	4	1489.9	19.1	3	1647.6	35.6
LC208	200	100	3	4	1810.1	25.3	4	1357.3	16.3	4	1385.6	18.1	4	1296.3	23.6
LR201	200	100	3	4	1937.8	31.6	4	1305.0	27.1	4	1519.4	21.2	3	1337.4	27.9
LR205	200	100	2	3	1398.7	21.5	3	1088.6	33.8	3	1180.6	26.1	3	1115.5	31.4
LR209	200	100	2	3	1417.6	13.7	3	1200.5	24.1	3	1208.1	12.7	3	1179.2	26.2
LR211	200	100	2	3	1326.6	22.6	3	1164.1	21.4	3	1195.3	16.5	3	1116.1	36.4
LRC201	200	100	3	4	1908.8	17.0	4	1449.7	28.4	4	1503.1	18.5	4	1445.3	29.6
LRC205	200	100	2	4	1564.9	28.4	4	1718.9	30.2	4	1902.2	16.3	4	1718.9	33.8
LRC206	200	100	2	3	1268.9	25.0	3	1285.1	40.1	3	1325.1	21.0	3	1235.4	38.1
LRC207	200	100	2	4	1231.6	25.7	4	1321.5	34.7	4	1380.5	15.1	3	1317.0	29.7
LRC208	200	100	2	4	1428.5	16.3	4	1433.8	25.4	4	1471.7	13.5	4	1479.5	36.1
AVG					1791.2	23.8		1400.5	27.6		1463.3	16.9		1395.8	31.6

From Table 4, it can be observed that the discrepancy between the total vehicle travel distances obtained by the ILS heuristic and those obtained by CPLEX is minimal. The ILS heuristic obtains the optimal solutions of five instances. The average gap between the total vehicle travel distances obtained by the ILS heuristic and the exact solutions obtained by CPLEX is 0.11%. The computing times for all instances using the ILS heuristic are much less than those using CPLEX. The ILS solution time for all three types of instances is relatively consistent, ranging from 0.01 to 0.08 s. CPLEX has the largest computing time 127.7 s. On average, the percentage of saving computing time is 98.62%. This suggests that the performance of the ILS heuristic in solving the MTVRPMTW problem is satisfactory.

When solving large-scale instances with 100 customers, as shown in Table 5, the ILS heuristic can find the solutions within 750 s. The average computing time of all instances is 448.01 s. This demonstrates the good performance of the ILS heuristic in solving the MTVRPMTW problem. To further explore the initial solution and the performance of each operator in the algorithm, on the same number of iterations, we carried out an improved Solomon insertion algorithm, local search only, with Or-opt based on the initial solution, and local search only with Relocate operator based on the initial solution. Table 5 shows the number of vehicles used, expressed as K, the total travel distance of vehicles expressed as $Z_{\text{S-Insert}}$, $Z_{\text{ILS-or}}$, $Z_{\text{ILS-relocate}}$, and Z_{ILS}, and the computing time of the solution under different solution conditions expressed as $t_{\text{S-Insert}}$, $t_{\text{ILS-or}}$, $t_{\text{ILS-relocate}}$, and t_{ILS}. The solution results show that the solution obtained by operator Or-opt is better than Relocate, and the solution obtained by ILS is better than the others. The results of LR201 and LRC207 also show that the number of vehicles obtained by the ILS is less than the others, so it can be concluded that the algorithm can effectively reduce the number of vehicles through local search and perturbation. The ILS heuristic can find the solutions within 60 s. The average computing time of all instances is 31.6 s. But the instance in Table 5 are large-scale problems, CPLEX cannot produce results in a reasonable time. This demonstrates the good performance of the ILS heuristic in solving the MTVRPMTW problem.

6. Conclusions

This paper, aimed at optimizing urban last-mile delivery logistics through adaptations of multiple trips and multiple time window options for vehicle routing, makes significant contributions to sustainable development goals by improving delivery efficiency and customer satisfaction, enhancing resource efficiency, and fostering economic viability within the logistics sector. Given the complexity of the MTVRPMTW, solving large-scale instances using optimization software such as CPLEX is challenging. To address this, a tailored ILS heuristic is developed to solve the problem. An improved Solomon greedy insertion algorithm, suitable for scenarios with multiple time windows and multiple routes, is proposed to generate the initial solution. Additionally, local search operators, including Or-opt and Relocate, as well as Random Exchange perturbation operations, are designed. Computational results demonstrate the effectiveness of the proposed model and algorithm. Compared to the MTVRPTW model, considering multiple time windows allows carriers to flexibly plan vehicle routes and select service time windows, thereby reducing the number of vehicles used and the total travel distance.

In conclusion, this paper delves into the more complex variants of the multi-trip vehicle routing problem (VRP) at a theoretical level, thereby enriching the theoretical and methodological research system in the field of the VRP. At a practical level, it provides useful insights for solving complex problems in logistics and distribution and offers decision-making support for future logistics planning and optimization.

To deepen the study of this problem, the following aspects can be developed in the future: (1) The multi-trip vehicle path problem for multiple delivery addresses with priority rankings can be studied. For multiple delivery addresses with different time periods, customers often have different priority rankings with different satisfaction levels, and the actual delivery should take the cost of delivery and customer satisfaction into account at the same time. (2) Multiple delivery options can be considered in the MTVRPMTW. It is possible to add consideration of the shared delivery location in the study of multi-trip vehicle paths with multiple delivery addresses, add the limitation of storage space in the shared delivery location, and set different delivery costs for door-to-door delivery and shared delivery addresses to carry out an in-depth study on the multi-trip vehicle path problem considering multiple delivery options. (3) More efficient ILS local search operators can also be designed to improve the solution quality based on this problem. (4) An exact algorithm can be developed for the multi-trip vehicle path problem considering flexible delivery options. More efficient exact algorithms can be studied, which can provide more reliable and high-quality solutions for large-scale problems in practical applications.

Author Contributions: Conceptualization, Y.W.; methodology, H.D.; software, H.D.; validation, Y.W. and H.S.; formal analysis, H.S.; investigation, Y.W.; writing—original draft preparation, H.S.; writing—original draft preparation, H.D.; writing—review and editing, Y.W. and H.S.; visualization, H.S.; supervision, Y.W.; project administration, Y.W.; funding acquisition, Y.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Humanities and Social Science Fund of Ministry of Education of China, grant number 22YJC630159, and the "Qinglan" Project Foundation of Jiangsu Province.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. Post Office of the People's Republic of China. 2023 Postal Industry Development Statistical Bulletin [EB/OL]. Available online: https://www.spb.gov.cn/gjyzj/c100015/c100016/202401/59eeb6e8b0e7404f8127aa2c7aebded6.shtml (accessed on 20 May 2024).
2. Dantzig, G.B.; Ramser, J.H. The truck dispatching problem. *Manag. Sci.* **1959**, *6*, 80–91. [CrossRef]
3. Golden, B.L.; Raghavan, S.; Wasil, E.A. *The Vehicle Routing Problem: Latest Advances and New Challenges*; Springer Science & Business Media: New York, NY, USA, 2008; pp. 3–589.
4. Toth, P.; Vigo, D. *Vehicle Routing: Problems, Methods, and Applications*, 2nd ed.; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2014; pp. 1–452.
5. Braekers, K.; Ramaekers, K.; Van Nieuwenhuyse, I. The vehicle routing problem: State of the art classification and review. *Comput. Ind. Eng.* **2016**, *99*, 300–313. [CrossRef]
6. Tan, K.; Liu, W.; Xu, F.; Li, C. Optimization model and algorithm of logistics vehicle routing problem under major emergency. *Mathematics* **2023**, *11*, 1274. [CrossRef]
7. Vidal, T.; Laporte, G.; Matl, P. A concise guide to existing and emerging vehicle routing problem variants. *Eur. J. Oper. Res.* **2020**, *286*, 401–416. [CrossRef]
8. Cattaruzza, D.; Absi, N.; Feillet, D. Vehicle routing problems with multiple trips. *4OR* **2016**, *14*, 223–259. [CrossRef]
9. François, V.; Arda, Y.; Crama, Y. Adaptive large neighborhood search for multitrip vehicle routing with time windows. *Transport. Sci.* **2019**, *53*, 1706–1730. [CrossRef]
10. Sethanan, K.; Jamrus, T. Hybrid differential evolution algorithm and genetic operator for multi-trip vehicle routing problem with backhauls and heterogeneous fleet in the beverage logistics industry. *Comput. Ind. Eng.* **2020**, *146*, 106571. [CrossRef]
11. Coelho, V.N.; Grasas, A.; Ramalhinho, H.; Coelho, I.M.; Souza, M.J.; Cruz, R.C. An ILS-based algorithm to solve a large-scale real heterogeneous fleet VRP with multi-trips and docking constraints. *Eur. J. Oper. Res.* **2016**, *250*, 367–376. [CrossRef]
12. Cattaruzza, D.; Absi, N.; Feillet, D. The multi-trip vehicle routing problem with time windows and release dates. *Transport. Sci.* **2016**, *50*, 676–693. [CrossRef]
13. Zhen, L.; Ma, C.; Wang, K.; Xiao, L.; Zhang, W. Multi-depot multi-trip vehicle routing problem with time windows and release dates. *Transport. Res. E-Log.* **2020**, *135*, 101866. [CrossRef]
14. Pan, B.; Zhang, Z.; Lim, A. Multi-trip time-dependent vehicle routing problem with time windows. *Eur. J. Oper. Res.* **2021**, *291*, 218–231. [CrossRef]
15. Grangier, P.; Gendreau, M.; Lehuédé, F.; Rousseau, L.M. An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. *Eur. J. Oper. Res.* **2016**, *254*, 80–91. [CrossRef]
16. Yang, Y. An exact price-cut-and-enumerate method for the capacitated multitrip vehicle routing problem with time windows. *Transport. Sci.* **2023**, *57*, 230–251. [CrossRef]
17. Ibaraki, T.; Imahori, S.; Kubo, M.; Masuda, T.; Uno, T.; Yagiura, M. Effective local search algorithms for routing and scheduling problems with general time-window constraints. *Transport. Sci.* **2005**, *39*, 206–232. [CrossRef]
18. Beheshti, A.K.; Hejazi, S.R.; Alinaghian, M. The vehicle routing problem with multiple prioritized time windows: A case study. *Comput. Ind. Eng.* **2015**, *90*, 402–413. [CrossRef]
19. Belhaiza, S.; Hansen, P.; Laporte, G. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. *Comput. Oper. Res.* **2014**, *52*, 269–281. [CrossRef]
20. Schaap, H.; Schiffer, M.; Schneider, M.; Walther, G. A large neighborhood search for the vehicle routing problem with multiple time windows. *Transport. Sci.* **2022**, *56*, 1369–1392. [CrossRef]
21. Lourenço, H.R.; Martin, O.C.; Stützle, T. Iterated local search: Framework and applications. *Handb. Metaheuristics* **2019**, *272*, 129–168.
22. Archetti, C.; Feillet, D.; Mor, A.; Speranza, M.G. An Iterated Local Search for the Traveling Salesman Problem with Release Dates and Completion Time Minimization. *Comput. Oper. Res.* **2018**, *98*, 24–37. [CrossRef]
23. Mardones, B.; Gatica, G.; Contreras-Bolton, C. A Metaheuristic for the Double Traveling Salesman Problem with Partial Last-in-First-out Loading Constraints. *Int. Trans. Oper. Res.* **2023**, *30*, 3904–3929. [CrossRef]

24. Dasari, K.V.; Singh, A. Two Heuristic Approaches for Clustered Traveling Salesman Problem with d-Relaxed Priority Rule. *Expert Syst. Appl.* **2023**, *224*, 120003. [CrossRef]
25. Schettini, T.; Gendreau, M.; Jabali, O.; Malucelli, F. An iterated local search metaheuristic for the capacitated demand-driven timetabling problem. *Transport. Sci.* **2023**, *57*, 1379–1401. [CrossRef]
26. Máximo, V.R.; Nascimento, M.C. A hybrid adaptive iterated local search with diversification control to the capacitated vehicle routing problem. *Eur. J. Oper. Res.* **2021**, *294*, 1108–1119. [CrossRef]
27. Penna, P.H.V.; Subramanian, A.; Ochi, L.S. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *J. Heuristics* **2013**, *19*, 201–232. [CrossRef]
28. Osorio-Mora, A.; Escobar, J.W.; Toth, P. An iterated local search algorithm for latency vehicle routing problems with multiple depots. *Comput. Oper. Res.* **2023**, *158*, 106293. [CrossRef]
29. Yahiaoui, A.E.; Afifi, S.; Allaoui, H. Enhanced iterated local search for the technician routing and scheduling problem. *Comput. Oper. Res.* **2023**, *160*, 106385. [CrossRef]
30. Solomon, M.M. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* **1987**, *35*, 254–265. [CrossRef]
31. Or, I. Traveling Salesman-Type Combinatorial Problems and Their Relation to the Logistics of Regional Blood Banking. Ph.D. Thesis, Northwestern University, Evanston, IL, USA, 1976.