systems

MDPI

Article

Multi-Trip Vehicle Routing Problem with Time Windows and Resource Synchronization on Heterogeneous Facilities

Rui Xu *, Shumin Li and Jiayan Wu

School of Business, Hohai University, Nanjing 211100, China
* Correspondence: rxu@hhu.edu.cn; Tel.: +86-25-6851-4618

Abstract: Inspired by long-distance road transport in industrial logistics in China, this paper studies a simultaneous loading scheduling and vehicle routing problem over a multi-workday planning horizon. Industrial cargo often requires specialized facilities, and these facilities vary in performance and quantity and are subject to available time constraints. Consequently, achieving coordinated optimization of vehicle routing and loading scheduling becomes a significant challenge in practice. We describe the studied problem as a multi-trip vehicle routing problem with time windows and resource synchronization on heterogeneous facilities. First, we develop a mixed integer programming model in a multi-workday setting to minimize the total travel distance and the number of vehicles. Moreover, a three-phase heuristic approach is developed. An initial solution is constructed using a sequential strategy in the first phase, and then an adaptive large neighbourhood search and a post-optimization procedure based on ejection chains are, respectively, designed to optimize the two hierarchical objective functions. Finally, extensive computational experiments are conducted to demonstrate the effectiveness of the proposed method. Specifically, the research results indicate that in long-distance road transport in industrial scenarios, expanding the planning horizon from a single workday to a multi-workday period could significantly reduce logistics operational costs and improve service quality.

Keywords: vehicle routing problem with time windows; multi-trip; heterogeneous facilities; hierarchical objectives; three-phase heuristic

Citation: Xu, R.; Li, S.; Wu, J. Multi-Trip Vehicle Routing Problem with Time Windows and Resource Synchronization on Heterogeneous Facilities. *Systems* **2023**, *11*, 412. https://doi.org/10.3390/systems11080412

Academic Editor: Brian Sauser

Received: 16 June 2023
Revised: 29 July 2023
Accepted: 7 August 2023
Published: 9 August 2023

1. Introduction

This paper is motivated by a long-distance logistics practice in a prominent industrial cigarette manufacturer in China. The manufacturer needs to arrange a fleet of vehicles to load cargo with the help of specialized facilities at the distribution center and then deliver them to customers with varying demands and service time windows. Given the wide geographical dispersion of customers, the manufacturer typically adopts a strategy of weekly scheduling, encompassing loading operations and distribution, to deliver cargo to customers. To reduce the number of vehicles used within a week, multiple trips may be assigned to a single vehicle, with each trip spanning 2–3 days on average. Furthermore, the scarcity of loading facilities, resulting from their limited number, significantly impacts the subsequent distribution process. During peak logistics periods, vehicles have to wait when all facilities are occupied. The maximum number of vehicles that can be loaded simultaneously is restricted by the number of available facilities, described as resource synchronization [1]. Moreover, due to the differences in the performance of various facilities, the facilities deployed in the distribution center are heterogeneous, indicating they have different loading speeds. This problem can be defined as a multi-trip vehicle routing problem with time windows and resource synchronization on heterogeneous facilities (MTVRPTWRS).

Multi-trip vehicle routing permits the assignment of multiple trips to the same vehicle [2]. However, most existing studies ignore the time involved in cargo handling

associated with each trip. In addition, this problem also addresses a critical yet often overlooked aspect of the multi-trip vehicle routing problem (MTVRP): the scarcity of loading and unloading facilities. The above two factors should be taken into account in industrial logistics because the loading and unloading operations are time-consuming and rely on scarce facilities. As a result, any scheduling plan that disregards resource synchronization is inherently suboptimal or infeasible [3]. Acknowledging the importance of these resource constraints is essential in formulating a feasible and optimal scheduling plan.

The MTVRPTWRS problem is NP-hard as it presents a variant problem of the traditional vehicle routing problem with resource synchronization. Introducing resource synchronization constraints implies a complex scheduling problem embedded within the vehicle routing problem [4]. In the MTVRPTWRS problem, we need to assign multiple trips to a single vehicle and arrange the sequences for the loading operations mapped by each trip. Constrained by the time windows, the vehicle must depart from the depot within a specific time interval, thereby necessitating the completion of the loading operation within the designated timeframe. To guarantee a feasible schedule, a comprehensive check is required to evaluate the assignment of available vehicles, the arrangement of loading operations, and the planning of each vehicle trip. These characteristics pose challenges to building efficient mathematical models and designing tailored algorithms.

In this study, we consider the hierarchical objectives that prioritize minimizing the total travel distance (TTD), followed by minimizing the number of vehicles (NV). The manufacturer's primary focus is on minimizing TTD since the distribution cost highly depends on TTD, which has a critical impact on improving enterprise efficiency. When logistics costs are minimized, the manufacturer then aims to minimize the number of dispatched vehicles. This consideration stems from the fact that each vehicle is typically operated by a regular driver. The driver's familiarity with the entire process, particularly the loading operations, contributes to increased operational efficiency and higher-quality delivery service.

The main contributions of this paper are threefold: (1) We introduce a novel variant of VRP in the industrial logistics environment called MTVRPTWRS with hierarchical objectives, which considers the impact of the availability of loading facilities on the subsequent routing plan. By incorporating realistic factors, the established model can significantly enhance the feasibility and quality of the scheduling plan. (2) We formulate this problem as a mixed-integer linear programming (MILP) model, with the objectives of minimizing TTT as the primary objective and minimizing NV as the secondary objective. Subsequently, we devise a three-phase heuristic to solve the problem and use IRACE to adjust the parameters of the proposed heuristic. (3) We conduct computational experiments on a new set of test instances for the MTVRPTWRS problem, which is designed based on the classical VRPTW test instances. The experimental results indicate that the proposed three-phase heuristic efficiently solves the MTVRPTWRS problem and performs well on the test instances.

The remainder of this paper is organized as follows. A brief review of the relevant literature is provided in Section 2. In Section 3, we present the problem description and establish a corresponding MILP model. In Section 4, we develop a three-phase heuristic to solve the MTVRPTWRS problem, and the performance is evaluated through computational experiments in Section 5. We conclude this paper and provide some promising directions for future research in Section 6.

2. Literature Review

Some recent studies have focused on the MTVRP and the vehicle routing problem with resource synchronization, respectively [5,6]. However, there is a scarcity of research that jointly addresses the features of "multi-trip" and "resource synchronization" in vehicle routing problems. To the best of our knowledge, only Huang et al. [7] studied a variant of MTVRP with resource synchronization related to urban waste collection. In the following subsections, we comprehensively review relevant studies on MTVRP and the vehicle routing problem with resource synchronization.

2.1. Related Works on MTVRP

In the MTVRP problem, each vehicle is allowed to perform multiple trips without exceeding the maximum working duration. Fleischmann [2] initially proposed a two-phase framework to tackle the MTVRP. This approach began by constructing multiple trips and then employed a bin-packing problem heuristic to distribute these trips among a fleet of vehicles. In the allocation phase, vehicles with predetermined maximum working durations were treated as capacity-constrained boxes, while trips with varying travel times were regarded as items with distinct weights. Since its inception, the two-phase framework has influenced subsequent research on solving the MTVRP. Taillard, Laporte and Gendreau [6] introduced a two-phase heuristic that utilized a tabu search algorithm with adaptive memory and developed MTVRP benchmark instances for evaluation purposes. Prins [5] developed a multi-phase heuristic algorithm aimed explicitly at minimizing overtime. Mingozzi et al. [7] employed an exact algorithm based on set partitioning to solve the MTVRP, successfully providing optimal solutions for 42 instances of the problem. Cattaruzza et al. [8] devised combined operators by integrating VRP neighbourhood operators such as relocate and swap and operators capable of swapping trips between different vehicles. These operators were embedded in a hybrid genetic algorithm for local search, improving solutions for specific MTVRP instances.

The introduction of time windows and time-dependent travel time in the MTVRP significantly amplifies its complexity. Battarra et al. [9] conducted a study on MTVRP with time windows, highlighting the requirement for each trip to start within a specific interval when assigning trips to vehicles. Failure to start a trip within the designated time interval led to time windows violations for customers. In another study, Pan et al. [10] focused on MTVRP with time windows and time-dependent travel time. They devised a component to evaluate the feasibility of a trip within a short time frame, which enhanced the algorithm's search efficiency. This approach contributed to improved performance and streamlined the search process.

In the aforementioned studies, the treatment of loading and unloading operation times is typically simplified. Most of these studies primarily focus on the loading and unloading time at the customer locations, often treating it as a fixed duration [9]. Pan, Zhang and Lim [10] assumed that the loading time at the depot is proportionate to the total service time required for the trip. This assumption acknowledged the impact of the routing plan on loading time but failed to account for potential waiting time caused by limited loading and unloading facilities.

2.2. Related Works on Vehicle Routing Problem with Resource Synchronization

Resource synchronization was defined as follows: "At any point in time, the total utilization or consumption of a specified resource by all vehicles must be less than or equal to a specified limit" [1]. The vehicle routing problem with resource synchronization generally emerged from the logistics domain, where specific scarce facility resources played a significant role in the routing process. For instance, in industrial logistics, the loading and unloading operations heavily rely on specialized facilities such as docking stations [3], pumps [11–15], and loaders [16,17].

In the vehicle routing problem with resource synchronization, there exist two crucial operations: constructing optimal routes for vehicles, assigning facilities and determining sequences for their loading and unloading operations. While cargo loading and unloading operations may occur at the depot and each customer node visited by vehicles, the allocation of facilities and sequences of the loading and unloading operations need to be taken into consideration only when the number of vehicles visiting a particular node exceeds the capacity of available facilities. Based on the distribution of facilities across nodes in the road network, two types of deployment modes can be distinguished: multi-point single-facility, where each node is equipped with a single facility, and single-point multi-facility, where a single node accommodates multiple facilities.

In a multi-point single-facility scenario, facilities are dispersed among various nodes in the network, with each node assigned a unique facility. In other words, although multiple vehicles are allowed to access the same node, the facility can only accommodate a single vehicle at any given time. Thus, designing a proper sequence for vehicles to access the facilities at a specific node is critical. In a dynamic airport AGV cargo system with multi-resource constraints, Ebben, van der Heijden and van Harten [3] designed a serial scheduling algorithm to maximize the delivery rate. Asbach, Dorndorf and Pesch [11] presented a novel local search approach to scheduling the routes of concrete mixer vehicles throughout a workday, ensuring efficient transportation from concrete-producing depots to concrete-demanding customers. Schmid, Doerner, Hartl and Salazar-González [13] proposed a hybrid algorithm that combines integer multi-commodity network flow (MCNF) and variable neighbourhood search (VNS) techniques to address the ready-mix concrete routing problem, which involved scheduling the transportation of concrete between asphalt concrete plants and construction sites and routing the fleet of vehicles to fulfil these transportation requests. Grimault, Lehuédé and Bostel [15] developed a two-phase framework to solve the ready-mix concrete routing problem by dividing the order first and then routing second. The algorithm adopted in the routing phase is based on Ebben, van der Heijden and van Harten [3]. Following that, Grimault, Bostel and Lehuédé [12] further proposed an adaptive large neighbourhood search algorithm to address the routing problem, where the removals in this algorithm were designed considering both routing and scheduling and the repair framework employed aligned with the aforementioned serial scheduling algorithm. El Hachemi, Gendreau, and Rousseau [17] studied a real-life log truck scheduling problem that exhibited similar characteristics to the concrete delivery problem, e.g., full truckload and resource synchronization constraints.

In a single-point multi-facility scenario, multiple facilities are deployed at a node that serves all vehicles passing through it, such as the depot or intermediate site. In the urban garbage collection process, garbage collected by the vehicle needs to be unloaded and processed on limited disposal. Huang et al. [18] developed a set partition model with two sets of mutually exclusive constraints and proposed a branch-pricing-cutting algorithm to solve the urban garbage collection problem, which shares the same features as the problem investigated in our study. Building upon the concept of "route first assemble second" [19], Grangier et al. [4] studied the vehicle routing problem with cross-docking and incorporated a resource constraint examination procedure during the route generation phase to enhance solution quality. The results proved that the scheduling heuristic has an excellent performance.

Existing research addresses real-life transportation problems with industry-specific characteristics, leading to varying assumptions and constraints. Specifically, studies focus on routing design and loading/unloading scheduling within two deployment modes. The most complex scenario involves the loading or unloading scheduling in a multi-facility mode, which requires a specified loading sequence and facility assignment. In our study, the MTVRPTWRS problem considers all loading facilities located at a unique depot, which is classified as the single-point multi-facility scenario. However, previous research oversimplifies practical scenarios by assuming homogeneous facilities and primarily focusing on a single objective, disregarding hierarchical objectives and diverse management requirements. Therefore, there is a need to explore approaches that consider hierarchical objectives to accommodate enterprises' diverse needs effectively.

3. Problem Description and Formulation

3.1. Problem Description

The sets, parameters, and variables for the problem description and mathematical formulation of the MTVRPTWRS problem are defined in Table 1.

Table 1. List of notations.

	Sets
H	The set of workdays, $H = \{1, \ldots, \|H\|\}$
V	The set of vertices, $V = \{0, n+1\} \cup V_c$
V_c	The set of customers, $V_c = \{1, \ldots, n\}$
A	The set of arcs, $A = \{(i,j) \| i, j \in V, i \neq j\}$
K	The set of vehicles, $K = \{1, \ldots, \|K\|\}$
M	The set of facilities, $M = \{1, \ldots, \|M\|\}$
VM	The set of virtual replicas of loading facilities, where each facility with multiple available periods is treated as a distinct entity with a single available period, $VM = \{1, \ldots, \|M\|\|H\|\}$
R	The set of trips, $R = \{1, \ldots, \|R\|\}$, the number of trips in R does not exceed the number of customers $\|V_c\|$
J	The set of the loading operations mapped by the set of trips, $J = \left\{J_1, \ldots, J_{\|R\|}\right\}$
MS	The set of virtual trips whose mapped loading operations could be used as the first virtual loading operation for different facilities on different workdays, $MS = \{\|R\|+1, \ldots, \|R\|+\|M\|\|H\|\}$
ME	The set of virtual trips whose mapped loading operations could be used as the last virtual loading operation for different facilities on different workdays, $ME = \{\|R\|+\|M\|\|H\|+1, \ldots, \|R\|+2\|M\|\|H\|\}$
KS	The set of virtual trips, where each trip can be used as the first virtual trip for a vehicle, $KS = \{\|R\|+2\|M\|\|H\|, \ldots, \|R\|+2\|M\|\|H\|+\|K\|\}$
KE	The set of virtual trips, where each trip can be used as the last virtual trip for a vehicle $KE = \{\|R\|+2\|M\|\|H\|+\|K\|+1, \ldots, \|R\|+2\|M\|\|H\|+2\|K\|\}$
	Parameters
q_i	The demand of customer i
$[e_i, l_i]$	The service time window of customer i
s_i	The service duration at customer i
t_{ij}	The time travelled from node i to node j
c_{ij}	The distances travelled from node i to node j, $t_{ij} = c_{ij}$ numerically
T_H	The working duration of a workday
T_M	The available duration of the facilities on a workday
Q	The maximum capacity of a vehicle
$\mathcal{M}$	A large number
	Decision variables
x_{ijr}	A binary variable is equal to 1 if arc $(i,j) \in A$ is travelled during trip r and 0 otherwise
y_{rumh}	A binary variable is equal to 1 if the mapped loading operation of trip r and u are processed sequentially on the facility m on the h^{th} workday and 0 otherwise
z_{irmh}	A binary variable is equal to 1 if node i is visited by trip r, and the mapped loading operation is executed on the facility m on the h^{th} workday and 0 otherwise
g_{ruk}	A binary variable is equal to 1 if the trip r and u are executed sequentially by vehicle k
D_k	A binary variable is equal to 1 if at least one non-virtual trip is assigned to vehicle k
b_r	A continuous variable represents the beginning time of the loading operation mapped by trip r
w_{ir}	A continuous variable represents the service beginning time at customer i in trip r
	Auxiliary variable
U_r	The set of customers visited by trip r
ϕ_r	The quantity of cargo loaded by a vehicle during trip r
L_{mh}	The working duration of facility m on the h^{th} workday
P_{rm}	The processing time of the loading operation J_r on facility m
$[a_r, d_r]$	The processing time window for the loading operation J_r
VP_r	The trip duration in which trip r occupies the assigned vehicle

The planning horizon of the MTVRPTWRS problem consists of multiple workdays, which is denoted as H. The working duration of a workday is T_H and the duration of

the planning horizon is $|H| * T_H$. The network is defined on a complete direct graph $G = (V, A)$. The graph consists of the node set $V = \{0, n+1\} \cup V_c$ and the arc set A. Node 0 represents the start of a trip, and node $n+1$ represents the end of a trip. The node set V_c represents the customer to be served. Each node $i \in V$ is associated with a service time s_i, a demand q_i, and a time window $[e_i, l_i]$. Note that $q_0 = q_{n+1} = 0$, $s_0 = s_{n+1} = 0$, $e_0 = e_{n+1} = 0$ and $l_0 = l_{n+1} = |H| * T_H$. Each arc $(i, \mathrm{j}) \in A$ associates with a travel time t_{ij} and travel distance c_{ij}. Assuming that the vehicle travels at a constant speed, the two values can be assumed to be numerically equal, i.e., $t_{ij} = c_{ij}$. A fleet of homogeneous vehicles with a capacity of Q is deployed to carry out the deliveries. A group of facilities is responsible for handling the loading operations, with each facility having a loading speed of δ_m. The facility is available for a duration of T_M on each workday, which is shorter than the working duration of a typical workday, i.e., $T_M < \mathrm{T_H}$. The available period of each loading facility on the $\mathrm{h^{th}}$ workday can be expressed as $[(h-1) * T_H, (h-1) * T_H + T_M]$, $h \in H$.

The following assumptions are introduced in the MTVRPTWRS problem:

The processing time for a loading operation can be acquired by dividing the quantity of loaded cargo by the loading speed of the assigned facility.
The number of vehicles loaded simultaneously cannot exceed the number of facilities $|M|$. Vehicles need additional waiting time when all the loading facilities are occupied by other vehicles.
Once a loading operation is initiated, it must be completed without interruptions or switching to alternative facilities.
Each loading operation must start and finish within the available period of loading facilities in a particular workday.
The time window is a hard constraint, meaning that vehicles are only allowed to begin service within the interval defined by the time window.
Each customer can only be visited once by a single vehicle.

Each vehicle has the flexibility to enter and exit the depot at any time during the planning horizon but has to return to the depot before the end of the final workday.

The MTVRPTWRS problem entails developing a weekly scheduling plan that involves identifying multiple trips for vehicles and scheduling the associated loading operations. To enhance understanding and facilitate a comprehensive analysis of the problem, we decompose it into three distinct levels of subproblems: *routing design* (lowest level), *loading scheduling* (middle level), and *trip scheduling* (highest level).

At the lowest level, the *routing design* subproblem focuses on generating a routing plan that includes multiple trips to meet customer requirements. The trips in the plan are represented by the set R. Each trip, denoted as r, visits a set of customers, U_r, and the amount of cargo loaded during trip r is ϕ_r, calculated as the sum of demand for all customers in U_r. Before carrying out a trip, the corresponding loading operation must be completed first. As a result, each trip is associated with a loading operation. Analyzing the trip attributes enables us to derive two attributes of the mapped loading operation: the processing time and the processing time window, which play a critical role in the *loading scheduling* subproblem. Noted that mapped loading operations refer to the correspondence between a vehicle's trip and the corresponding loading operation performed before the trip. All loading operations are stored in the set J. The processing time of a loading operation J_r on facility m, denoted as P_{rm}, depends on the loading speed and cargo quantity: $P_{rm} = \phi_r / \delta_m$. The processing time window is vital in connecting the *routing design* and *loading scheduling* subproblems. It serves as an expected timeframe for the completion of the loading operation. In other words, the processing time window establishes a specific timeframe within which a vehicle departs from the depot, ensuring trip feasibility and minimizing waiting time at all customer points. A trip is treated as feasible if the vehicle can depart from the depot at time zero and reach subsequent customers within their specified time windows. Departing earlier would lead to increased vehicle waiting time, while departing later would violate the time window constraints for visiting some customers.

Consequently, the loading operation is expected to be completed within this period. The processing time window is precisely defined by Definition 1, and the detailed computation rules can be found in Appendix A.

Definition 1. *(Processing time window). The processing time window of the loading operation J_r mapped by trip r is represented by $[a_r, d_r]$, where a_r denotes the earliest expected finish time of J_r and d_r represents the latest completion time, also referred to as the due date.*

In the middle level, the *loading scheduling* subproblem aims to assign loading operations associated with the trip set R to different facilities. For convenience, we consider the same facility on different working days as distinct virtual facilities and introduce a set of virtual replicas of loading facilities. Each facility with multiple available periods is treated as a distinct entity with a single availability period, denoted as $VM = \{1, \ldots, |M||H|\}$.

By integrating the routing plan and loading scheme, we determine two crucial time points for each trip: the beginning time of the loading operation and the time of return to the depot. We define the vehicle utilization period VP_r for trip r as the timeframe spanning from the start of the loading operation to the vehicle's arrival back at the depot. If trip r is assigned to vehicle k, it will occupy the vehicle during VP_r, while the remaining periods can be scheduled for other trips. At the highest level, the *trip scheduling* subproblem involves generating a trip scheme that assigns and arranges trips on different vehicles, ensuring non-overlapping vehicle utilization periods for multiple trips assigned to the same vehicle.

3.2. Mathematical Formulation

The MTVRPTWRS model comprises two objective functions and a set of constraints. To align with the three subproblems that form the MTVRPTWRS problem, we categorize the constraints into three distinct parts: constraints (3)–(12) correspond to the routing design subproblem, constraints (13)–(24) pertain to the loading scheduling subproblem, and constraints (25)–(30) relate to the trip scheduling subproblem.

1. Objective functions

$$minTTD = \sum_{r \in R} \sum_{i \in V} \sum_{j \in V} x_{ijr} c_{ij} \tag{1}$$

$$min\ NV = \sum_{k \in K} D_k \tag{2}$$

Equation (1) is the primary objective, which is to minimize TTD, and Equation (2) is the secondary objective, which is to minimize NV.

2. Constraints related to the *routing design* subproblem

$$\sum_{r \in R} \sum_{j \in V} x_{ijr} = 1 (\forall i \in V_c) \tag{3}$$

$$\sum_{j \in V} \sum_{i \in V} x_{ijr} q_i \leq Q \ \ (\forall r \in R) \tag{4}$$

$$\sum_{j \in V} x_{ijr} = \sum_{j \in V} x_{jir} \ \ (\forall i \in V_c, r \in R) \tag{5}$$

$$\sum_{i \in V} x_{0ir} = \sum_{i \in V} x_{i(n+1)r} \leq 1 \ \ (\forall r \in R) \tag{6}$$

$$\sum_{i \in V} x_{i0r} = \sum_{i \in V} x_{(n+1)ir} = 0 \ \ (\forall r \in R) \tag{7}$$

$$\sum_{i \in V} \sum_{j \in V} x_{ijr} = 0 \ \ (\forall r \in MS \cup \ ME \cup \ KS \cup \ KE) \tag{8}$$

$$\sum_{j\in V_s}\sum_{i\in V_s} x_{ijr} \le |V_s| - 1 \ \ (\forall r \in R, V_s \subseteq V_c, \ |V_s| \ge 2) \tag{9}$$

$$w_{jr} - (w_{ir} + s_i + t_{ij}) \ge \overline{\mathcal{M}}(x_{ijr} - 1) \ \ (\forall i, j \in V, \ r \in R) \tag{10}$$

$$w_{ir} + \overline{\mathcal{M}}(\sum_{j\in V} x_{ijr} - 1) \le l_i \ \ (\forall i \in V, \ r \in R) \tag{11}$$

$$w_{ir} + \overline{\mathcal{M}}(1 - \sum_{j\in V} x_{ijr}) \ge e_i \ \ (\forall i \in V, \ r \in R) \tag{12}$$

Constraint (3) mandates that each customer is visited exactly once. Constraint (4) specifies the capacity of the vehicle. Constraint (5) enforces flow conservation across all customer nodes. Constraints (6) and (7) ensure that each trip begins at node 0 and ends at node $n + 1$. Constraint (8) guarantees that virtual trips have no access to customers. Constraint (9) represents the classical sub-tour elimination constraint. Constraint (10) considers the time continuity of adjacent arcs within a trip. Constraints (11) and (12) impose hard time window constraints for customers and the depot.

3. Constraints related to the *loading scheduling* subproblem

$$\sum_{j\in V} x_{ijr} = \sum_{h\in H}\sum_{m\in M} z_{irmh} \ \ (\forall i \in V_c, r \in R) \tag{13}$$

$$\sum_{h\in H}\sum_{m\in M}\sum_{u\in R\cup\ ME} y_{rumh} \le 1 \ \ (\forall r \in R) \tag{14}$$

$$\sum_{i\in V}\sum_{j\in V} x_{ijr} \le \overline{\mathcal{M}} \sum_{h\in H}\sum_{m\in M}\sum_{u\in R\cup\ ME} y_{rumh} \ \ (\forall r \in R) \tag{15}$$

$$\sum_{i\in V} z_{irmh} \le \overline{\mathcal{M}} \sum_{u\in R\cup\ ME} y_{rumh} \ \ (\forall r \in R, m \in M, h \in H) \tag{16}$$

$$\sum_{i\in V_c} z_{irmh} = 0 \ \ (\forall r \in MS \cup\ ME, m \in M, h \in H) \tag{17}$$

$$\sum_{u\in R\cup\ ME} y_{rumh} = \sum_{u\in R\cup\ MS} y_{urmh} \ \ (\forall r \in R, m \in M, h \in H) \tag{18}$$

$$\sum_{r\in MS}\sum_{u\in R\cup\ ME} y_{rumh} = \sum_{r\in ME}\sum_{u\in R\cup\ MS} y_{urmh} = 1 \ \ (\forall m \in M, h \in H) \tag{19}$$

$$\sum_{r\in ME}\sum_{u\in R\cup\ MS} y_{rumh} = \sum_{r\in MS}\sum_{u\in R\cup\ ME} y_{urmh} = 0 \ \ (\forall m \in M, h \in H) \tag{20}$$

$$w_{0r} \ge b_r + \sum_{i\in V} q_i z_{irmh}/\delta_m + \overline{\mathcal{M}}(\sum_{u\in R\cup\ ME} y_{rumh} - 1) \ \ (\forall r \in R, m \in M, h \in H) \tag{21}$$

$$b_u - b_r + \overline{\mathcal{M}}(1 - y_{rumh}) \ge \sum_{i\in V} q_i z_{irmh}/\delta_m \ \ (\forall r \in R \cup\ MS, u \in R \cup\ ME, m \in M) \tag{22}$$

$$b_r \ge \overline{\mathcal{M}}(\sum_{u\in R\cup\ ME} y_{rumh} - 1) + (h-1) * T_H \ \ (\forall r \in R \cup\ MS, m \in M, h \in H) \tag{23}$$

$$b_r + \sum_{i\in V} q_i z_{irmh}/\delta_m \le \overline{\mathcal{M}}(1 - \sum_{u\in R\cup\ ME} y_{rumh}) + (h-1) * T_H + T_M \ \ (\forall r \in R, m \in M, h \in H) \tag{24}$$

Constraints (13)–(16) ensure that a mapped trip is assigned to a specific facility on a given workday only if it involves delivery, and Constraints (16) and (17) establish the relationship between decision variables x_{ijr} and y_{rumh}, as well as z_{irmh} and y_{rumh} by in-

troducing a very large positive value $\overline{\mathcal{M}}$. Constraint (17) prohibits virtual trips from accessing any customers. Constraint (18) mandates that non-virtual loading operations must have both a predecessor and a successor, ensuring proper sequencing and connectivity. Constraints (19) and (20) specify that the loading sequence at each facility should start with a virtual loading operation from the set MS and end with another virtual loading operation from the set ME. Constraint (21) enforces that delivery can only begin after the completion of loading operations. Constraint (22) ensures that subsequent loading operations at any facility commence no earlier than the completion of the preceding loading operation. Constraints (23) and (24) guarantee that each loading operation starts and ends within the available period of facilities on a given workday.

4. Constraints related to the *trip scheduling* subproblem

$$\sum_{i\in V}\sum_{j\in V} x_{ijr} \le \overline{\mathcal{M}} \sum_{k\in K}\sum_{u\in R\cup\ KE} g_{ruk} \ \ (\forall r \in R) \tag{25}$$

$$\sum_{r\in R\cup\ KS}\sum_{u\in R\cup\ KE} g_{ruk} \le \overline{\mathcal{M}} D_k \ \ (\forall k \in K) \tag{26}$$

$$\sum_{u\in R\cup\ KS} g_{urk} = \sum_{u\in R\cup\ KE} g_{ruk} \ \ (\forall r \in R, k \in K) \tag{27}$$

$$\sum_{r\in KS}\sum_{u\in R\cup\ KE} g_{ruk} = \sum_{r\in KE}\sum_{u\in R\cup\ KS} g_{urk} = 1 (\forall k \in K) \tag{28}$$

$$\sum_{r\in R\cup\ KE}\sum_{u\in KS} g_{ruk} = \sum_{r\in R\cup\ KS}\sum_{u\in KE} g_{urk} = 0 (\forall k \in K) \tag{29}$$

$$w_{(n+1)r} \le b_u + \overline{\mathcal{M}}(1 - g_{ruk}) \ \ \forall r, u \in R, k \in K \tag{30}$$

Constraints (25) and (26) require that each non-empty trip be assigned to a single vehicle. Constraint (27) ensures continuity between consecutive trips within the same vehicle. Constraints (28) and (29) dictate that each vehicle begins with a virtual trip from the set KS and ends with another virtual trip from the set KE. Constraint (30) guarantees that the vehicle cannot be loaded with the cargo required for the next trip until it returns to the depot from the preceding trip.

3.3. Illustration of an Example

In order to enhance understanding of the MTVRPTWRS problem, we illustrate a small example.

Example 1: This example involves serving 11 customers within a planning horizon spanning two workdays. The customer service information can be found in Table 2. Note that nodes 0 and 12 correspond to the beginning and end of a trip, respectively. Each workday has a working duration T_H of 50 and an available duration T_M of 30. Two facilities are available, with loading speeds of 2 and 1, respectively.

Table 2. Information on customer requests.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
q_i	4	4	2	8	3	4	3	2	6	5	3
s_i	5	4	2	8	3	4	3	2	6	5	3
e_i	50	60	10	20	20	20	30	10	10	20	40
l_i	90	90	30	40	60	80	70	20	40	90	90

The solution for the MTVRPTWRS problem comprises a routing plan, a loading scheme, and a trip scheme. The optimal routing plan is presented in Table 3 and Figure 1. Table 4 details the attributes of the mapped loading operations, including the processing

time and the processing time window. Furthermore, Figure 2 visualizes the loading and trip schemes through Gantt charts, offering a clear overview. For more detailed information, Tables 5 and 6 present comprehensive data on the loading and trip schemes in tabular form.

Table 3. The results of the routing plan (TTD = 104).

Trip	Travel Time τ_r	Quantity of Cargo ϕ_r	Trip Sequence
r_1	20	4	$\langle 0,1,2,12\rangle$
r_2	27	10	$\langle 0,3,4,12\rangle$
r_3	31	10	$\langle 0,5,6,7,12\rangle$
r_4	29	8	$\langle 0,8,9,12\rangle$
r_5	18	8	$\langle 0,10,11,12\rangle$

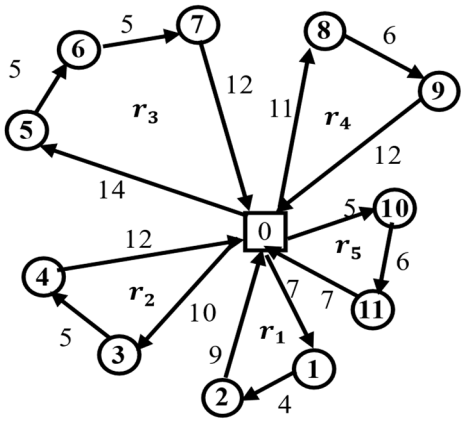

Figure 1. The visualization of the routing plan (TTD = 104).

Table 4. The attributes of the loading operations.

Loading Operation	Processing Time on Different Facilities		Processing Time Window	
	P_{r1}	P_{r2}	a_r	d_r
J_1	8	4	43	71
J_2	10	5	2	20
J_3	10	5	0	39
J_4	8	4	0	9
J_5	8	4	24	79

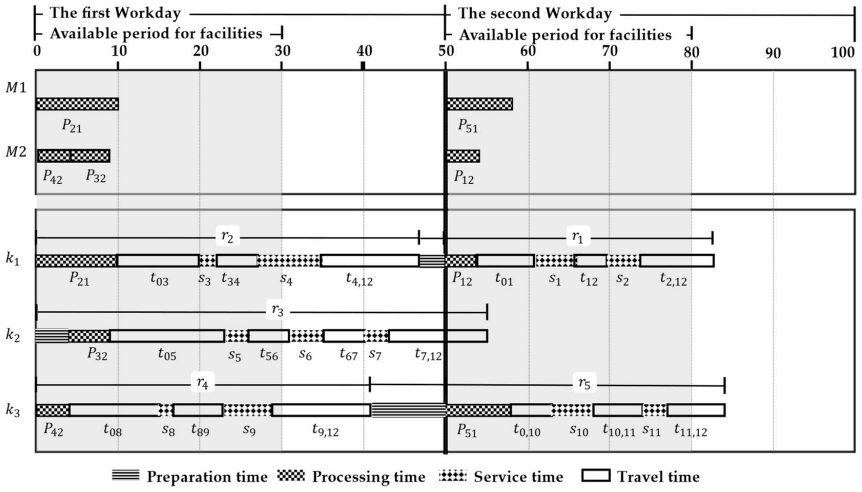

Figure 2. Gantt chart of loading scheme and trip scheduling (NV = 3).

Table 5. The results of the loading scheme (NV = 3).

Facility	The First Workday		The Second Workday	
	Sequence	Working Time P_m	Sequence	Working Time P_m
$M1$	(J_2)	10	(J_5)	8
$M2$	(J_4, J_3)	9	(J_1)	4

Table 6. The results of the trip scheme (NV = 3).

Vehicle	Trip Sequence
k_1	$\langle r_2, r_1 \rangle$
k_2	$\langle r_3 \rangle$
k_3	$\langle r_4, r_5 \rangle$

The routing plan involves visiting 11 customers distributed across five trips. As an example, the sequence of the trip r_1 is <0,1,2,12>. The TTD is the total travel time for all trips, totalling 104. Based on the routing plan, we derive a set of loading operations with distinct attributes, illustrated in Table 4.

Figure 2 illustrates a Gantt chart displaying the loading scheme and trip scheme. On the first workday, the loading operation J_2 is processed at facility $M1$, while J_4 and J_3 are sequentially handled at facility $M2$. On the second workday, J_5 is processed at $M1$, and J_1 is processed at $M2$. Examining Table 4, we observe that the processing time window of J_4 is $0, 9$, indicating that it must be completed between 0 and 9 to serve customers 8 and 9 within their requested time windows. In the current loading scheme, the loading operation J_4 is completed at time 4, aligning with the required processing time window and enabling the subsequent trip to be feasibly executed. In the trip scheme, three vehicles are assigned to five trips: vehicle k_1 executes trips r_2 and r_1 sequentially, vehicle k_3 executes trips r_4 and r_5 sequentially, and vehicle k_2 exclusively handles trip r_3. The routing plan depicted in Figure 1 and the loading and trip schemes showcased in Figure 2 comprises a solution featuring a TTD of 104 and an NV of 3.

4. Solution Approach

The MTVRPTWRS problem is NP-hard and the issue studied in this paper arises from real-world scenarios, often characterized by large-scale instances that require obtaining feasible solutions within a limited solving time. When using exact methods to solve the mathematical formulation presented in Section 3, only very small problem instances can be tackled. Additionally, these exact methods typically rely on state-of-the-art commercial MIP solvers, which may not be suitable for practitioners. Instead, heuristic algorithms and simulation methods have become the primary approach for solving vehicle routing problems in practical scenarios [20–22]. Therefore, our solution approach is based on a three-stage heuristic framework, allowing us to efficiently handle real-world-sized instances and achieve high-quality solutions in a reasonable amount of time without the need for an additional solver.

Our solution approach comprises three phases. In the first phase, we utilize a sequential strategy to address the *routing design* and *loading scheduling* subproblems and construct the initial routing plan and loading scheme. In the second phase, we employ the Adaptive Large Neighborhood Search (ALNS) algorithm to minimize the TTD and acquire an optimized routing plan alongside a feasible loading scheme. To enhance the exploration capabilities of ALNS, we integrate a strategy enabling it to search in infeasible regions, complemented by an augmented cost function with penalty terms. Notably, the *trip scheduling* subproblem is not considered during the first and second phases, and trips are individually assigned to vehicles by default.

The feasible routing plan and loading scheme obtained in the second phase become the input for the third phase. In this final phase, we adopt a post-optimization procedure based on ejection chains to construct a trip scheme and further optimize the loading scheme

correspondingly to improve the objective NV. To streamline the third phase, we incorporate the processing time window to swiftly evaluate the feasibility of each trip within different loading and trip schemes.

For a comprehensive overview of our proposed three-phase heuristic approach, please refer to Figure 3.

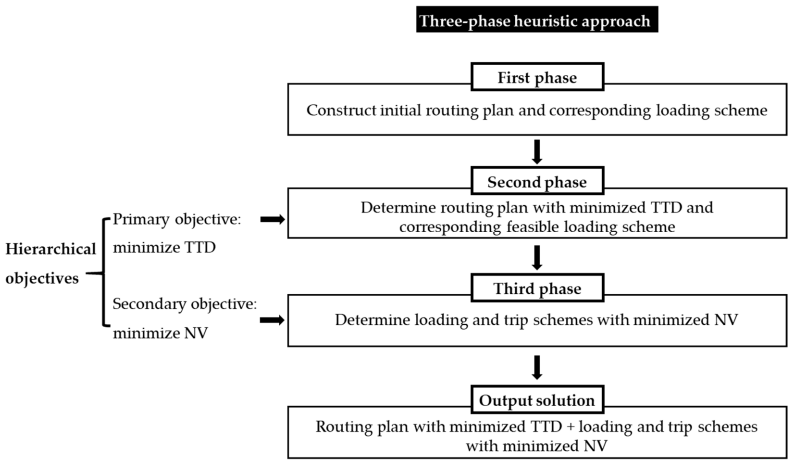

Figure 3. The overview of the three-phase heuristic approach.

4.1. First Phase: Initial Solution Construction

In the initial phase, we adopt a sequential strategy inspired by the "route first assemble second" method proposed by Froger, Jabali, Mendoza and Laporte [19]. This strategy entails two key steps: firstly, constructing a set of trips R using a basic greedy heuristic, and secondly, assigning the corresponding loading operations to different facilities and workdays based on scheduling rules. The scheduling rule, derived from the parallel machine scheduling rule introduced by Dogramaci and Surkis [23], aims to minimize total tardiness. The rule prioritizes unscheduled loading operations with the earliest due date and assigns them to facilities with minimum slack. In this context, minimum slack represents the difference between the due date and the completion time of a loading operation on different facilities.

Sequential Strategy:

Step 1.	Employ a basic greedy heuristic to generate a set of trips, denoted as R. This heuristic starts with an empty route and systematically inserts customers by considering the minimized insertion costs. A new empty trip is created if no customers can be feasibly inserted. This iterative process continues until all customers are successfully inserted. The customer insertion cost is determined by a linearly weighted combination of the increased distance and the delayed arrival time of the successor node after inserting the candidate customer.
Step 2.	Generate the set of mapped loading operations J from the set of trips R obtained in Step 1. Compute the processing time P_{rm} and processing time window $[a_r, d_r]$ of each mapped loading operation based on the attributes of the trip r.
Step 3.	Sort the loading operations in non-decreasing order according to their due dates and obtain a list of unscheduled loading operations, denoted as CJ.
Step 4.	Assign the loading operations in CJ to the different facilities with minimum slack time in turn. The slack time SL^j_{mh} of the loading operation J_r on the facility m and the h^{th} workday is defined as $SL^j_{mh} = b_r + P_{rm} - d_r$. A negative value of slack time indicates that the loading operation is completed later than the due date. If the slack time on all the facilities is negative, assign the loading operation to the facility and order with the largest value of slack time; otherwise, assign it to the facility with the smallest positive slack time.

4.2. Second Phase: TTD Minimization

In the second phase, we employ ALNS to improve the initial routing plan and loading scheme obtained in the first phase, to minimize TTD. We aim to identify the routing plan with minimal TTD and a corresponding feasible loading scheme. To enhance the search performance of the algorithm, we enable exploration within the infeasible solution space and introduce an associated augmented cost function incorporating penalty terms.

ALNS is a powerful metaheuristic algorithm that has gained widespread popularity for its efficiency and effectiveness in solving various variants of vehicle routing problems. It starts with an initial solution and iteratively operates based on the principle of destroying and repairing solutions to find better-quality solutions. The algorithm's adaptive nature allows it to dynamically adjust its search strategy during the optimization process based on the encountered solution quality and problem characteristics, achieving a balance between exploration and exploitation to escape local optima and improve solution quality. Meanwhile, we also incorporate the Metropolis criterion from Simulated Annealing to enhance the algorithm's exploration ability.

4.2.1. An Augmented Cost Function with Penalty Terms

In the second phase, both feasible and infeasible solutions are permitted, and a solution is evaluated using an augmented cost function:

$$f(S) = TTD + \alpha \sum_{i \in V_c} \sum_{r \in R} \max\{w_{ir} - l_i, 0\} + \beta \sum_{h \in H} \sum_{m \in M} (L_{mh} - T_M), \tag{31}$$

In the above equation, the first term represents the primary objective TTD, the second term is the penalty cost function for violating the customer time window constraints, and the third is the penalty cost function for violating the available duration of facilities. L_{hm} denotes the total working duration of facility m on the h^{th} workday, which is the sum of the processing times of all loading operations assigned to facility m on that workday. The parameters α and β correspond to different penalization factors.

An adaptive learning mechanism is employed to effectively guide the search process and generate a high-quality feasible solution by the end of the second phase. The penalization parameters α and β are initially set to α_0 and β_0, respectively, and their values are constrained within the intervals (a_{min}, α_{max}) and $(\beta_{min}, \beta_{max})$. The search process is divided into segments consisting of 100 iterations each, with the penalization coefficients remaining constant throughout each segment. At the end of every segment, the value of α is updated based on the following criteria: if the number of infeasible solutions obtained within the segment exceeds the threshold ρ_1, α is multiplied by $1 + \lambda$; if the number falls below the threshold ρ_2, α is divided by $1 + \lambda$. The adjustment of parameter β follows the same rules as α. For the setting of the above parameter values, please refer to the experimental part of Section 5.

4.2.2. Removal and Repair Operators

In each iteration of ALNS, the removal operators remove n_q customers and put them into requests bank $\mathcal{B}$ temporarily. Subsequently, the repair operators select customers from requests bank B and reinsert them into the current partial solution. The value of n_q is randomly determined within the intervals of $(\varepsilon_{min} * |V_c|, \varepsilon_{max} * |V_c|)$, where ε_{min} and ε_{max} represent the minimum and maximum percentages of customers to be removed, respectively.

For the removal operator, we employ the following five operators.

1. Random customer removal: This operator randomly removes n_q customers.
2. Random route removal: This operator bears similarities to the random customer removal operator; however, it removes all the customers in the selected trip simultaneously rather than individual customers. The process involves randomly selecting a trip. If removing all customers along the trip exceeds n_q, a random subset of nodes within the trip is selected for removal. Conversely, if the removal of all customers in

the trip is within the limit, all customers along the trip are removed. This process is repeated until the desired number of customers n_q, is reached.

3. Related customer removal: This operator attempts to improve the solution by removing a set of customers with high relatedness and rearranging their insertion positions. The relatedness $RL(i,j)$ between the customer i and j consists of distance and time terms, defined as follows $RL(i,j) = c_{ij} + \left|w_i - w_j\right|$, where w_i and w_j denote the start time of service at customer i and j. The process begins by selecting a customer at random. Subsequently, a customer is randomly chosen from the request bank for comparison. The remaining unremoved customers are then sorted in non-decreasing order based on the relatedness with the compared customer, giving higher preference to the candidate with lower relatedness. The process is repeated until the desired number of nodes n_q have been removed.
4. Worst removal: This operator aims to enhance the solution by identifying a new insertion position for the customer with the high customer insertion cost. The customer insertion cost of customer i, denoted as $cost(i,S)^-$, is calculated as $cost(i,S)^- = f(S) - f_{-i}(S)$, where $f(S)$ and $f_{-i}(S)$ denote the augmented cost function before and after removing customer i from the current solution S, respectively. The remaining unremoved customer are then sorted in non-increasing order according to customer insertion cost, giving higher priority to the candidate customer with the higher customer insertion cost. The process is repeated until the desired number of nodes n_q have been removed.
5. Facility tardiness removal: This operator aims to decrease the overtime value of a facility by removing customers in the trip associated with the loading operation assigned to the facility. If the total processing time of assigned loading operations on a facility surpasses the available duration of the facility for a given workday, it will be considered overtime. The process is repeated until either the number of removed customers reaches n_q or all facilities across all workdays are within the designated available period. Generally, there is a higher possibility of removal for customers with greater demand.

For the repair operator, we propose the following two operators.

1. Greedy insertion: this operator always selects customer i from requests bank $\mathcal{B}$ with the smallest insertion cost $Cost(i,s)^+$, and inserts it into the corresponding trip and position. The process is repeated until all customers stored in requests bank $\mathcal{B}$ are successfully inserted into the trip. The value of the change in the augmented cost function $f(S)$, represented by $\Delta f_{i,r}$, indicates the impact of inserting customer i into trip r. If inserting customer i into trip r would violate the vehicle capacity constraint, then $\Delta f_{i,r}$ is set to $+\infty$. The insertion cost $Cost(i,s)^+$ of customer i is defined as the minimum $\Delta f_{i,r}$ among all possible trips, denoted as $Cost(i,s)^+ = \underset{r \in R}{min} \Delta f_{i,r}$.
2. Two-regret insertion: The operator prioritizes customer i with the highest regret value RV_i. The process is repeated until all customers stored are reinserted into the solution. Firstly, compute the value $\Delta f_{i,r}$ for inserting customer i in all candidate trips and sort them in non-decreasing order. If the index of $\Delta f_{i,r}$ among all candidate trips is j, denote it as $\Delta f_{i,r_j}$. Then the regret value RV_i represents the difference between the insertion cost of the best trip $\Delta f_{i,r_1}$, and the second-best route $\Delta f_{i,r_2}$, calculated as $RV_i = \Delta f_{i,r_2} - \Delta f_{i,r_1}$.

4.2.3. Adaptive Weight Mechanism

In each iteration of the ALNS algorithm, we employ a roulette wheel mechanism to select a pair of removal and repair operators from the candidate operators and recreate the solution. Each operator is assigned a weight, with higher weights indicating a greater possibility of selection. As operators' performance can differ depending on the problem structure, we follow the practice proposed by Ropke and Pisinger [24] and utilize an adaptive weight mechanism to enhance the selection of operators better suited for

the MTVRPTWRS problem. This approach dynamically updates the weights of operators based on their historical performance, increasing the possibility of selecting more effective operators.

Considering 100 iterations as a segment, the weight of each operator remains constant within the segment and is uniformly updated at the segment's end. Let $w_{j,p}$ represent the weight of operator j in segment p. The weight of operator j in segment $p+1$ is then calculated as follows:

$$w_{j,p+1} = (1-\gamma)w_{j,p} + \gamma\frac{\varphi_{j,p}}{\max\{1,\sigma_{j,p}\}}, \tag{32}$$

where $\varphi_{j,p}$ denotes the score accumulated by operator j in segment p, $\sigma_{j,p}$ represents the cumulative times that operator j is selected in segment p, and $\gamma \in (0,1)$ controls the speed of weight adjustment. Following the approach presented by Ropke and Pisinger [24], we set $\gamma = 0.1$.

Within the segment p, if the operator i is selected in a given iteration, the cumulative times $\theta_{j,p}$ of its selection is incremented by one, and the cumulative score $\varphi_{j,p}$ is updated based on the following three cases:

$$\varphi_{j,p} = \begin{cases} \varphi_{j,p} + \xi_1, & if\ f(S_{new}) < f(S_{best}), \\ \varphi_{j,p} + \xi_2, & if\ f(S_{best}) < f(S_{new}) < f(S), \\ \varphi_{j,p} + \xi_3, & if\ f(S) < f(S_{new})\ but\ be\ accepted, \end{cases} \tag{33}$$

where S_{best} indicates the globally best solution found so far, S represents the current solution being compared, S_{new} denotes the newly recreated solution. In the first case, when a new globally best solution is generated, then the scores of the removal and repair operator involved are increased by ξ_1. In the second case, when a new solution better than the current solution is found, then the scores are increased by ξ_2. In the third case, a new solution inferior to the current solution is constructed but is accepted with the probability $e^{-(f(S_{new})-f(S))/TP}$, then the scores are increased by ξ_3. The third solution acceptance criterion is derived from the simulated annealing algorithm, known as the Metropolis criterion. This criterion prevents the algorithm from getting stuck in local optima and enhances its exportation. The temperature parameter, TP, is initially set to $TP_{initial}$ and gradually decreases as $TP = TP * cooling$, where $0 < cooling < 1$. We set the $TP_{initial}$ to a value that ensures a 50% probability of accepting a solution 5% worse than the current solution, as described in Ropke and Pisinger [24].

The pseudocode of the ALNS algorithm for the second phase is described as shown in Algorithm 1.

Algorithm 1 ALNS algorithm

$S_{best} = S_0$; $S = S_0$; $TP = TP_{initial}$ //S_0 denote the initial solution
while termination conditions are not met **do**:
    select removal and repair operators adaptively
    $S_{new} = RemovalAndRepair(S)$
    if $f(S_{new}) < f(S_{best})$ **then:**
        $S_{best} = S_{new}$; $S = S_{new}$;
    end if
    if $f(S_{new}) < f(S)$ **then:**
        $S = S_{new}$
    else if $f(S_{new}) > f(S)$ **then**:
        accepted $S = S_{new}$ with probability $e^{-(f(S_{new})-f(S))/TP}$
    end if
    update the scores of operators and penalty coefficient, $TP = TP * cooling$
end while
return S_{best}

4.3. Third Phase: NV Minimization

In the second phase, we obtain a routing plan with a minimized TTD and a corresponding feasible scheduling scheme. Note that in the second phase, each vehicle performs a separate trip by default, and the number of trips is the number of vehicles dispatched. In the third phase, we further attempt to assign multiple trips to the same vehicle to reduce the number of vehicles while maintaining the original routing plan, which involves optimizing the load*ing* and *trip scheduling* subproblems. Under different loading and trip schemes, vehicles leave the depot at different times, which subsequently affects the arrival time of different customers within the trip. If the arrival time of a customer does not fall within the required time window, the loading and the trip scheme are viewed as infeasible. Considering that the trip sequence is fixed, we utilize the processing time window to examine whether the customer time window constraint is violated under different loading schemes and trip schemes in a short time.

The post-optimization procedure is designed on the ejection chain. The ejection chain was first proposed by Glover [25] and has shown good performance in solving VRPTW problems to minimize the number of vehicles [26,27]. In the post-optimization procedure, we prioritize the removal of vehicles with the lowest unitization and place the trips assigned to that vehicle into the ejection pool, then sequentially find feasible insertion positions for the removed trip in the current loading and trip schemes. If a feasible insertion position cannot be obtained for a removed trip, the removal operation for that vehicle is backtracked, and another attempt is made to remove the vehicle with the second lowest utilization. If all the removed trips are successfully inserted into the remaining vehicles, continue to select the remaining removed vehicles based on the utilization of vehicles and repeat the process until the removal operation fails for all vehicles. Vehicle utilization is the ratio of the total duration of the vehicle utilization period for all trips assigned to that vehicle to the planning horizon.

Post-optimization procedure

Step 1.	Sequence the vehicles in the no-decreasing order based on their vehicle utilization, and get a candidate list of removing vehicles CK.
Step 2.	If the candidate list CK is empty, skip to Step 12; otherwise, remove the first vehicle k in CK, remove all the trips assigned to vehicle k, and put them into the ejection pool EP.
Step 3.	If the EP is empty, skip to Step 1; otherwise, select the trip r with the earliest due date d_r in EP.
Step 4.	Compute the deviation time of the loading operation J_r mapped by trip r in different feasible candidate insertion positions and get a candidate insertion list ML. An insertion position of a loading operation includes the assigned workday h, facility m, insertion index j, and beginning time b_r. The deviation time serves as merit to assess the different insertion positions, with smaller values indicating better insertion positions. For a given facility m and h^{th} workday, let $A_{hm} = \left\{J_1, \ldots, J_{\lvert A_{hm}\rvert}\right\}$ denotes the loading sequence. To determine the deviation time, insert J_r into the j^{th} position of the sequence A_{hm} and record the deviation time according to the following two cases: (1) if the loading operation can be completed within the processing time window, then the deviation time is recorded as 0; (2) if the completing time is earlier than the earliest desired start time, the deviation time is $a_r - b_r - p_{km}$. Other cases are deemed infeasible and are not recorded. Each feasible insertion position is stored in the form of $[h, m, j, b_r, deviation\ time]$ in ML.
Step 5.	If the list ML is empty, then the insertion is regarded as a fail and skip to Step 11; otherwise, insert J_r into the position with the shortest deviation time.
Step 6.	Compute the vehicle utilization period for trip r.

Step 7.	Compute the conflict time of trip r at different insertion positions and obtain the list of candidate insertion position KL. The insertion position of a trip consists of the inserted vehicle k and the inserted position j, and the conflict time is an indicator to evaluate different insertion positions. The smaller the conflict time corresponds to a more favourable position. The order of the executed trips of vehicle k is denoted by B_k. Insert the trip r into the j^{th} position of the sequence B_k. If the vehicle use period of trip r does not overlap with the preceding line in the use period, then the overlapping period with the succeeding trip is recorded as the conflict time. Each insertion position is stored in the form of $[k, j, conflict\ time]$ in KL.
Step 8.	If the list KL is empty, the insertion fails and skip to Step 11; otherwise, select the insertion position with the shortest conflict time.
Step 9.	If the conflict time of the insert position is zero, insert the trip into that position directly, skip to Step 3, and initialize the number of ejections to 0; otherwise, eject the succeeding trip, which conflicts with trip r after inserted.
Step 10.	If the number of ejections reaches the maximum number, skip to Step 11; otherwise, the current number of ejections increases by one, and the ejected trip is used as the trip r to be inserted, and skip to Step 4.
Step 11.	Restore the solution to the solution when vehicle k is not removed, remove vehicle k from the sequence CK, and skip to Step 2.
Step 12.	Output the final solution.

5. Computational Experiments

The addressed MTVRPTWRS problem originates from a tobacco manufacturing company in China, and the proposed solution is applicable to this industrial enterprise. However, due to confidentiality restrictions, we could not use the actual Tobacco Manufacturing Instances. Thus, to assess and analyze the performance of the proposed three-phase algorithm, we design test instances for the MTVRPTWRS based on Solomon's VRPTW benchmark and employ IRACE to tune the parameters of the algorithm. We conduct three distinct sets of experiments, outlined as follows: (1) The first set of experiments aims to analyze the contributions of different components to the objective. (2) The second set of experiments compares the detailed performance across various test instances. (3) The third set of experiments evaluates the benefits of the weekly scheduling plan compared to those of the daily scheduling plan, particularly in long-distance logistics scenarios.

The proposed algorithm was implemented in python and executed on a powerful desktop PC equipped with a 2.3 gigahertz Pentium processor and 512 gigabytes of RAM. The test instances were adapted from the classical Solomon instances, which are detailed in Section 5.1. We run the algorithm ten times for each instance to ensure reliable results. In the case of the three-phase heuristic, the second phase concludes either after a maximum runtime of 3600 s or if no improved solutions are obtained for 200 consecutive iterations. After conducting preliminary testing, we established specific values for the penalization coefficient bounds, thresholds, and renewal factor in our experiments. The lower bound for the penalization coefficient was set at $\alpha_{min} = \beta_{min} = 1$, and the upper bound was set at $\alpha_{max} = \beta_{max} = 20$. Furthermore, we assigned $\rho_1 = 30$ and $\rho_2 = 60$. as the respective thresholds. Finally, we adopted a renewal factor of $\lambda = 2$.

5.1. Test Instances of the MTVRPTWRS Problem

Due to the absence of standard test instances, we made modifications to the benchmark instances from the existing literature for our experiments. The MTVRPTWRS test instances were derived from the VRPTW Solomon instances, with a specific focus on the R2, C2, and RC2 instances that have wide time windows. Each test instance includes 100 customers. These three types are chosen because other instances with narrow time windows limit the number of possible trips a vehicle can make. The customer distribution varies across the three types: R2 has random distribution, C2 has clustered distribution, and RC2 has a mixed distribution.

The original VRPTW test instances already include customer and vehicle information, so only parameters related to the planning horizon and facilities need to be added. In all three types of instances, the following settings are applied uniformly: the planning horizon consists of 5 workdays, i.e., $|H| = 5$, there are 2 machines, i.e., $|M| = 2$, and the values (δ_1, δ_2) are set to $(8, 10)$. However, the working duration T_H and available periods of facilities T_M differ across instances due to variations in the time windows of the depot in different types. Detailed information can be found in Table 7.

Table 7. The parameters of T_H and T_M on different types.

	C2	R2	RC2
T_H	700	200	200
T_M	560	160	160

5.2. Automatic Configuration

The IRACE package is a common automatic algorithm configuration tool which implements the Iterated Race method for tuning the most appropriate parameter settings [10,28]. In this experiment, the maximum number of runs and concurrent threads are set to 5000 and 1024, respectively, to improve the tuning speed. Three critical parameters need to be set in this study: (1) the minimum and maximum percentage $[\varepsilon_{min}, \varepsilon_{max}]$ of customers to be removed, (2) the *cooling* coefficient used for controlling the temperature, and (3) the operation reward *score* configuration.

Here the tuning range of ε_{min} and ε_{max} are set to $[0.05, 0.10, 0.15]$ and $[0.30, 0.35, 0.40, 0.45, 0.50, 0.55, 0.60]$, respectively. The *cooling* coefficient is varied from 0.90 to 1.0. Noted there are three different *score* configurations employed from the previous related study. Score #1 refers to the score setting $[\xi_1, \xi_2, \xi_3] = [1, 0, 5]$ obtained from Demir, et al. [29]; Score #2 denotes the setting $[\xi_1, \xi_2, \xi_3] = [33, 9, 13]$ proposed by Ropke and Pisinger [25]; Score #3 is $[\xi_1, \xi_2, \xi_3] = [33, 20, 13]$ employed from Masson, et al. [30].

The three best superior configurations obtained from IRACE are listed in Table 8. In subsequent experiments, we use the best configuration [0.15; 0.3; 0.91; #3] as the default parameter setting.

Table 8. Three best configurations obtained from IRACE.

Parameter	Configuration 1	Configuration 2	Configuration 3
ε_{min}	0.15	0.10	0.05
ε_{max}	0.3	0.3	0.3
cooling	0.91	0.9	0.91
score	#3	#2	#3

5.3. Contribution of Different Phases of the Heuristic

The initial solution, constructed in the first phase, undergoes successive optimization through the ALNS in the second phase and the post-optimization procedure in the third phase. To assess the effectiveness of the heuristic algorithm, we present the objective functions of the solutions obtained at different phases, as detailed in Table 9. The columns TTD and NV display the average values of TTD and NV, respectively, at each phase. The column Gap(TTD) indicates the difference in average TTD between the first and second phases, while the column Gap(NV) represents the difference in average NV between the second and third phases.

Table 9. The results of different phases.

	First Phase		Second Phase			Third Phase		
	TTD	**NV**	**TTD**	**NV**	**Gap (TTD)**	**TTD**	**NV**	**Gap (NV)**
C2	1741.50	13.5	1253.73	13.43	28.01%	1253.73	9.62	28.37%
R2	1745.08	8.05	990.22	8.07	43.26%	990.22	7.65	5.22%
RC2	2114.00	9	1197.04	9.72	43.38%	1197.04	9.34	3.86%
Average	1834.05	9.95	1125.23	10.19	38.65%	1125.23	8.69	14.68%

As observed from Table 9, the ALNS in the second phase effectively reduces TTD, while the post-optimization procedure effectively reduces the number of vehicles. However, the performance varies significantly across different instances. Upon optimization by ALNS, the TTD objective exhibits a substantial improvement, averaging 38.65%. Notably, the extent of improvement varies widely among different instances, ranging from a minimum improvement of 28.01% for the cluster-distributed C2 instances to over 43% for the R2 and RC2 instances. Subsequently, the post-optimization procedure further reduces the number of vehicles used, achieving an average reduction of 14.68%. Specifically, in the C2 instances, the average reduction in the number of vehicles amounts to 28.37%. However, the reduction in the number of vehicles is not significant in the R2 and RC2 instances.

5.4. Analysis of Computational Results on Different Instances

In Section 5.3., we observed significant variations in the optimized performance of the post-optimization procedure across different types of instances, and we will further analyze the underlying reasons by examining the data metrics in this section. Table 10 presents the metrics related to vehicles and trips. The column TTP indicates the average travel time proportion to each trip's planning horizon, while the column PTP represents the average proportion of processing time to the planning horizon. The VPP column signifies the average ratio of trip duration to the planning horizon. It is important to note that vehicles may spend some time in the depot area after completing loading operations, so the value of VPP may slightly exceed the sum of the TTP and the PTP. The column number of trips denotes the average number of trips performed by each vehicle, while the column vehicle utilization indicates the average ratio of the trip duration of multiple trips to the planning horizon for each vehicle.

Table 10. The indicators about vehicles and trips on different instances.

	Vehicle		Trip		
	Number of Trips	**Vehicle Utilization**	**TTP**	**PTP**	**VPP**
C2	1.42	62.36%	41.91%	0.46%	44.79%
R2	1.06	54.15%	46.26%	2.06%	51.23%
RC2	1.04	51.77%	44.18%	1.99%	49.76%
Average	1.17	56.20%	44.35%	1.53%	48.81%

As observed in Table 10, the C2 instance exhibits an average of 1.42 trips executed by vehicles, with a vehicle utilization rate of 62.35%, notably higher than the R2 and RC2 instances. On average, each trip in the R2 and RC2 instances occupies 51.23% and 49.76% of the planning horizon, respectively. Furthermore, the time window constraint requires vehicles to initiate each trip within a specific time interval, which poses challenges in assigning multiple trips to the same vehicle when individual trips are time-consuming. However, in the C2 instance, the proportion of time intervals within the planning horizon is slightly lower than in the R2 and RC2 instances, facilitating the allocation of multiple trips to the same vehicle and improving vehicle utilization.

Lower vehicle utilization is highly undesirable. For vehicles from a third-party logistics company, lower utilization necessitates additional third-party vehicles, leading to

increased management complexity. For company-owned vehicles, lower vehicle utilization means that the company needs to acquire more vehicles, resulting in increased driver salaries and maintenance expenses. Ultimately, these higher costs are passed on to the customers' purchasing costs, highlighting the importance of business managers developing differentiated purchasing strategies and charging varying costs based on customers' delivery time requirements. Our work can assist companies in developing differentiated pricing strategies to balance service quality and cost.

5.5. Comparison between the Weekly Scheduling Plan and the Daily Scheduling Plan

The weekly scheduling plan employed in this study differs from the traditional daily scheduling plan primarily in terms of the planning horizon. The former focuses on planning routes and scheduling loading operations for customers over a week, while the latter encompasses route planning for all customers within a single workday. Generally, the weekly scheduling plan offers greater flexibility and enables a more cost-effective routing plan and loading scheme. This section compares these two scheduling plans to validate this proposition.

Under the daily scheduling plan, the routing plan and loading scheme must be designed based on the specific customer demand for each workday. The total distance travelled under this plan is the cumulative distance covered by vehicles across different working days, and the number of vehicles used is determined by merging trips that do not conflict with the planning horizon.

As evident from Table 11, adopting the weekly scheduling plan instead of the daily scheduling plan yields an average reduction of 34.22% in the total distance travelled and a decrease of 33.52% in the number of vehicles used. The weekly scheduling plan offers the advantage of enabling vehicles to visit geographically close customers on different workdays during the same trip. This flexibility contributes to the construction of a more efficient routing plan. Conversely, such a capability is unattainable within the constraints of the daily scheduling plan.

Table 11. The Comparison between the weekly scheduling plan and the daily scheduling plan.

	Daily Scheduling Plan		**Weekly Scheduling Plan**		**Gap (TTD)**	**Gap (NV)**
	TTD	**NV**	**TTD**	**NV**		
C2	1607.10	14.43	1253.73	9.62	21.99%	33.33%
R2	1605.63	11.52	990.22	7.65	38.33%	33.57%
RC2	2134.34	14.00	1197.04	9.34	43.92%	33.26%
Average	1710.52	13.08	1125.23	8.69	34.22%	33.52%

6. Conclusions

This paper investigates a new variant of VRP in industrial logistics that considers time windows, resource synchronization on heterogeneous facilities, multi-trip, and hierarchical objectives consisting of minimizing TTD and minimizing the NV. To address this problem, we decompose it into three subproblems: *routing design*, *loading scheduling*, and *trip scheduling*. We analyze the interrelation between these subproblems and adopt hierarchical objectives, followed by proposing a MILP model and a three-phase heuristic to solve the problem. Experimental results demonstrate the ALNS and post-optimization effectiveness within the three-phase heuristic algorithm, which can effectively optimize TTD and NV objectives. Furthermore, our findings reveal that the weekly scheduling plan surpasses the traditional daily scheduling plan in long-distance logistics scenarios. It achieves significant reductions in both TTD and NV objectives, with an average decrease of 34.22% in TTD and 33.52% in NV. These results highlight the importance of incorporating the weekly scheduling plan to improve efficiency in industrial logistics.

In industrial logistics scenarios, the scarcity of loading and unloading resources profoundly impacts the subsequent routing process, underscoring the need for our research to provide feasible and high-quality scheduling plans for enterprises operating under tight

loading and unloading resource. Moreover, it is imperative to consider additional realistic factors to enhance the practical applications of our approach. For instance, introducing limitations on the maximum working duration for a single trip and the minimum interval time for multiple trips can mitigate the risk of driver fatigue and enhance transportation safety. Furthermore, future research should emphasize the MTVRPTWRS problem in dynamic environments, enabling the acceptance of emergency insertion orders from customers and updating the unexecuted weekly scheduling plan. This enhancement would significantly improve the applicability of the model in real-life scenarios.

Author Contributions: Conceptualization, R.X.; supervision, R.X.; methodology, R.X. and J.W.; validation: J.W.; software: J.W.; writing—original draft preparation, J.W.; formal analysis, S.L.; resources, S.L.; data curation, S.L.; writing—review and editing, R.X., S.L. and J.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Guangdong Provincial Key Laboratory (Grant No.2020B121201001), the National Natural Science Foundation of China (Grant No.62106098/62272210), and the Stable Support Plan Program of Shenzhen Natural Science Fund (Grant No. 20200925154942002).

Data Availability Statement: The data presented in this study are available in [article].

Acknowledgments: The authors would like to acknowledge the support and inspiration provided by the Business Data Laboratory at the School of Business, Hohai University. The authors are grateful to the anonymous reviewers for greatly improving the quality of this paper.

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

Appendix A

Inspired by the work of Battarra, Monaci and Vigo [9] in solving MTVRP with time windows, we define the processing time window. The processing time window of the loading operation J_r consists of a earliest desired completion time a_r and a due date d_r, which can be obtained by the following derivation rules of the processing time window.

Derivation rules of processing time window

Step 1.	It is assumed that the visiting sequence of trip r is $\sigma_r = \left\langle i_0, i_1, \ldots, i_{\lvert U_r \rvert}, i_{n+1} \right\rangle$. Then the earliest service start time $a_{r,i_{\lvert U_r \rvert}}$ and the latest service start time $d_{r,i_{\lvert U_r \rvert}}$ at customer node $i_{\lvert U_r \rvert}$ could be obtained from Equations (A1) and (A2), respectively.

$$a_{r,i_{|U_r|}} = e_{i_{|U_r|}}, \tag{A1}$$

$$d_{r,i_{|C_r|}} = \min\left(l_{i_{|U_r|}}, l_{i_{|U_r+1|}} - t_{i_{|U_r|},i_{|U_r+1|}} - s_{i_{|U_r|}}\right). \tag{A2}$$

Step 2.	Initialize $j = \lvert U_r \rvert - 1$
Step 3.	Compute the earliest service start time a_{r,i_j} and the latest service start time d_{r,i_j} at i_j according to Equations (A3) and (A4), respectively.

$$a_{r,i_j} = \max\left(e_{i_j}, a_{k,i_{+1}} - t_{i_{(j+1)},i_j} - s_{i_j}\right), \tag{A3}$$

$$d_{k,i_j} = \min\left(l_{i_j}, d_{k,i_{(j+1)}} - t_{i_j,i_{(j+1)}} - s_{i_j}\right). \tag{A4}$$

Step 4.	If $j > 0$, let $j = j - 1$ and skip to Step3; otherwise, output the results as follows: $a_r = a_{r,i_0}$,$d_r = d_{r,i_0}$.

References

1. Drexl, M. Synchronization in Vehicle Routing—A Survey of VRPs with Multiple Synchronization Constraints. *Transp. Sci.* **2012**, *46*, 297–316. [CrossRef]
2. Fleischmann, B. *The Vehicle Routing Problem with Multiple Use of Vehicles*; Fachbereich Wirtschaftswissenschaften, Universität Hamburg: Hamburg, Germany, 1990.
3. Ebben, M.J.R.; van der Heijden, M.C.; van Harten, A. Dynamic transport scheduling under multiple resource constraints. *Eur. J. Oper. Res.* **2005**, *167*, 320–335. [CrossRef]

4. Grangier, P.; Gendreau, M.; Lehuédé, F.; Rousseau, L.-M. The vehicle routing problem with cross-docking and resource constraints. *J. Heuristics* **2019**, *27*, 31–61. [CrossRef]
5. Prins, C. A simple and effective evolutionary algorithm for the vehicle routing problem. *Comput. Oper. Res.* **2004**, *31*, 1985–2002. [CrossRef]
6. Taillard, É.D.; Laporte, G.; Gendreau, M. Vehicle Routeing with Multiple Use of Vehicles. *J. Oper. Res. Soc.* **1996**, *47*, 1065–1070. [CrossRef]
7. Mingozzi, A.; Roberti, R.; Toth, P. An Exact Algorithm for the Multitrip Vehicle Routing Problem. *INFORMS J. Comput.* **2013**, *25*, 193–207. [CrossRef]
8. Cattaruzza, D.; Absi, N.; Feillet, D.; Vidal, T. A memetic algorithm for the Multi Trip Vehicle Routing Problem. *Eur. J. Oper. Res.* **2014**, *236*, 833–848. [CrossRef]
9. Battarra, M.; Monaci, M.; Vigo, D. An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. *Comput. Oper. Res.* **2009**, *36*, 3041–3050. [CrossRef]
10. 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]
11. Asbach, L.; Dorndorf, U.; Pesch, E. Analysis, modeling and solution of the concrete delivery problem. *Eur. J. Oper. Res.* **2009**, *193*, 820–835. [CrossRef]
12. Grimault, A.; Bostel, N.; Lehuédé, F. An adaptive large neighborhood search for the full truckload pickup and delivery problem with resource synchronization. *Comput. Oper. Res.* **2017**, *88*, 1–14. [CrossRef]
13. Schmid, V.; Doerner, K.F.; Hartl, R.F.; Salazar-González, J.-J. Hybridization of very large neighborhood search for ready-mixed concrete delivery problems. *Comput. Oper. Res.* **2010**, *37*, 559–574. [CrossRef]
14. Schmid, V.; Doerner, K.F.; Hartl, R.F.; Savelsbergh, M.W.P.; Stoecher, W. A Hybrid Solution Approach for Ready-Mixed Concrete Delivery. *Transp. Sci.* **2009**, *43*, 70–85. [CrossRef]
15. Grimault, A.; Lehuédé, F.; Bostel, N. A two-phase heuristic for full truckload routing and scheduling with split delivery and resource synchronization in public works. In Proceedings of the 2014 International Conference on Logistics Operations Management, Rabat, Morocco, 5–7 June 2014; pp. 57–61.
16. El Hachemi, N.; El Hallaoui, I.; Gendreau, M.; Rousseau, L.-M. Flow-based integer linear programs to solve the weekly log-truck scheduling problem. *Ann. Oper. Res.* **2015**, *232*, 87–97. [CrossRef]
17. El Hachemi, N.; Gendreau, M.; Rousseau, L.-M. A heuristic to solve the synchronized log-truck scheduling problem. *Comput. Oper. Res.* **2013**, *40*, 666–673. [CrossRef]
18. Huang, N.; Li, J.; Zhu, W.; Qin, H. The multi-trip vehicle routing problem with time windows and unloading queue at depot. *Transp. Res. Part E Logist. Transp. Rev.* **2021**, *152*, 102370. [CrossRef]
19. Froger, A.; Jabali, O.; Mendoza, J.E.; Laporte, G. The electric vehicle routing problem with capacitated charging stations. *Transp. Sci.* **2022**, *56*, 460–482. [CrossRef]
20. He, P.; Li, J. Vehicle routing problem with partly simultaneous pickup and delivery for the cluster of small and medium enterprises. *Arch. Transp.* **2018**, *45*, 35–42. [CrossRef]
21. Kłodawski, M.; Jacyna, M.; Vasek, R.; Klimek, P.; Jachimowski, R.; Szczepański, E.; Lewczuk, K. Route Planning with Dynamic Information from the EPLOS System. *Teh. Glas.* **2020**, *14*, 332–337. [CrossRef]
22. Peng, Y.; Mo, Z.; Liu, S. Passenger's routes planning in stochastic common-lines' multi-modal transportation network through integrating Genetic Algorithm and Monte Carlo simulation. *Arch. Transp.* **2021**, *59*, 73–92. [CrossRef]
23. Dogramaci, A.; Surkis, J. Evaluation of a Heuristic for Scheduling Independent Jobs on Parallel Identical Processors. *Manag. Sci.* **1979**, *25*, 1208–1216. [CrossRef]
24. Ropke, S.; Pisinger, D. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transp. Sci.* **2006**, *40*, 455–472. [CrossRef]
25. Glover, F. New ejection chain and alternating path methods for traveling salesman problems. In *Computer Science and Operations Research*; Balci, O., Sharda, R., Zenios, S.A., Eds.; Pergamon: Amsterdam, The Netherlands, 1992; pp. 491–509.
26. Lim, A.; Zhang, X. A Two-Stage Heuristic with Ejection Pools and Generalized Ejection Chains for the Vehicle Routing Problem with Time Windows. *INFORMS J. Comput.* **2007**, *19*, 443–457. [CrossRef]
27. Nagata, Y.; Bräysy, O. A powerful route minimization heuristic for the vehicle routing problem with time windows. *Oper. Res. Lett.* **2009**, *37*, 333–338. [CrossRef]
28. López-Ibáñez, M.; Dubois-Lacoste, J.; Pérez Cáceres, L.; Birattari, M.; Stützle, T. The irace package: Iterated racing for automatic algorithm configuration. *Oper. Res. Perspect.* **2016**, *3*, 43–58. [CrossRef]
29. Demir, E.; Bektaş, T.; Laporte, G. An adaptive large neighborhood search heuristic for the Pollution-Routing Problem. *Eur. J. Oper. Res.* **2012**, *223*, 346–359. [CrossRef]
30. Masson, R.; Lehuédé, F.; Péton, O. The Dial-A-Ride Problem with Transfers. *Comput. Oper. Res.* **2014**, *41*, 12–23. [CrossRef]