A hybrid urban delivery system with robots

Shaohua Yu, Ann Melissa Campbell, Jan Fabian Ehmke, Jakob Puchinger

A Hybrid Urban Delivery System With Robots

Shaohua Yu*[a], Ann Melissa Campbell[b], Jan Fabian Ehmke[c], Jakob Puchinger[d,e]

[a]*School of Intelligent Manufacturing, Nanjing University of Science and Technology, Nanjing 210094, China*
[b]*Business Analytics Department, Tippie College of Business, University of Iowa, Iowa City, Iowa, 52242 USA*
[c]*Business Analytics Group, Universität Wien, Vienna, Austria*
[d]*EM Normandie Business School, Métis Lab, 92110, Clichy, France*
[e]*Université Paris-Saclay, CentraleSupélec, Laboratoire Génie Industriel, 91190, Gif-sur-Yvette, France*

Abstract

Cities are now restricting access for conventional delivery technologies in some areas, requiring businesses to adopt more flexible distribution systems to complete their deliveries. We present a two-echelon hybrid truck-based robot delivery system for last-mile logistics operations. Robots can navigate through truck no-go areas such as pedestrian zones and college campuses, while trucks can travel through less restricted areas. The hybrid delivery model allows the distribution system to automatically select the better distribution strategy, thereby improving distribution efficiency.

We present a mixed-integer linear program to model the proposed system. We also offer valid inequalities to strengthen the formulation and a large neighborhood search-based algorithm with innovative adaptive methods and multiple operators to solve medium and large-scale instances efficiently. Computational experiments are conducted to evaluate how our proposed model performs. Sensitivity analysis experiments considering truck no-go areas of different sizes and area access time windows are performed and reveal managerial insights. We suggest setting up appropriate time windows for some truck no-go areas to reduce the burden of logistics companies in increasing distribution routes due to access restrictions.

Keywords: Logistics, Innovative last mile distribution, Hybrid truck-based robot delivery, Access restrictions

1. Introduction

To reduce the number of conventional vehicles entering the inner city, several cities have implemented congestion pricing schemes. However, recent research has suggested that these schemes may actually increase emissions in city centers, and higher congestion fees may not be correlated to lowering emissions (Zhang et al. 2019). Alternatively, some cities have tried to ease traffic pressure in parts of inner cities by setting up truck no-go areas (where trucks cannot access). For example, the city of Cambridge has received permission to ban all commercially-plated trucks over 2.5 tons gross vehicle weight on several streets 24 hours a day (Joseph 2010). The emergence of truck no-go areas makes it important to consider new ways to serve the customers in these areas.

The rapid growth of sensors, the Internet of Things, and artificial intelligence have promoted technological progress and applications of robot urban delivery (Sonneberg et al. 2019), which provides a potential solution for deliveries in truck no-go areas. More and more robots are used in applications in urban logistics for delivering goods. For example, JD.COM, Starship Technologies, and TwinswHeel have conducted robot distribution tests in specific areas (Chevallier 2017, Li 2017, Gu 2018, Andrew 2019). Also, the popularity and demand for robot deliveries have increased due to the COVID-19 pandemic and the need for contactless delivery due to social distancing guidelines. Therefore, consumers, businesses, and governments have shifted from being cautious beta testers to enthusiastic early adopters of this innovative technology (Pani et al. 2020). However, due to the limitations of technology and safety, it is difficult for robots to complete the full delivery from distribution centers to customers alone.

Considering the advantages of truck and robot deliveries, robots have been used in urban logistics in the context of two-echelon distribution for the second echelon (last-mile delivery). Some robots are dispatched from transfer hubs in the city center or from neighborhood hubs near residential areas. Others use robots for distribution through trucks carrying robots to various locations in a city. According to Yu et al. (2022), the two-echelon delivery model and truck-based robot delivery model have their advantages. Which model is better depends on the fixed costs of trucks, robots, satellites, and other factors. When the fixed costs of robots, trucks, and satellites are lower, the two-echelon routing model is more efficient than the truck-based robot delivery model in a hybrid pickup and delivery setting.

In this paper, we want to combine the two-echelon and the truck-based robot delivery model by building a hybrid truck-based robot delivery model (HTRD). The HTRD model lets the distribution system optimize the distribution

Email addresses: shaohua.yu@njust.edu.cn (Shaohua Yu*), ann-campbell@uiowa.edu (Ann Melissa Campbell), jan.ehmke@univie.ac.at (Jan Fabian Ehmke), jpuchinger@em-normandie.fr (Jakob Puchinger)

strategy (using hubs or parking nodes) to improve distribution efficiency while fulfilling complex restrictions given by municipalities. We also want to look at the advantages and disadvantages of the combined HTRD model using either the two-echelon delivery model or the truck-based robot delivery model. To this end, we study an HTRD system, where trucks or trucks carrying robots move along the first-level routes, serving truck customers and visiting parking nodes to drop off/pick up their robots or visit hubs to unload the freight. Robots start from parking nodes or hubs to address customer services along the second-level routes.

This paper makes the following contributions: (1) We propose an HTRD model, combining the two-echelon delivery and truck-based robot delivery system. From a methodological perspective, compared to the classic truck-based robot delivery model, this model requires a new decision: whether to visit a hub. Visiting the hub may increase the travel distance, but it could also enhance delivery efficiency, presenting a trade-off. (2) We present a mixed-integer linear program (MILP) for the HTRD model and then propose some valid inequalities to speed up MILP solving. (3) We propose new adaptive methods for the ALNS-based algorithm for the newly-introduced problem, which takes the joint effect of the destroy and repair operators more into account, and considers the difficulty and benefits of obtaining a better solution in the weight adjustment process. (4) We compare our HTRD model with the separate two-echelon delivery model and the truck-based robot delivery model to evaluate how our proposed model performs, and we find that the adaptability of the HTRD model is higher than the other two models. (5) We conduct sensitivity analysis experiments to assess the impact of truck no-go areas, including their size and time windows, and introduce the concept of heterogeneous time windows for customers. Sensitivity analysis experiments show that as the truck no-go area shrinks, the cost of the distribution system is significantly reduced. Besides, as the length of the truck no-go area access time window increases, the impact gradually decreases.

The HTRD problem is described and modeled in Section 2, and the related literature is in Section 3. Section 4 deals with the heuristic approaches. Computational experiments are presented in Sections 5. We conclude in Section 6.

2. Problem Description and Formulation

This section first introduces the HTRD problem. Then, we formulate the HTRD mathematically and propose some valid inequalities.

2.1. Problem Statement

We consider an HTRD system that includes two-echelon and truck-based robot delivery modes. In the HTRD system, trucks or trucks carrying robots go along first-level routes, provide services to consumers, visit parking nodes to drop off or pick up their robot, or visit hubs to unload goods. Robots start from parking nodes or hubs to handle deliveries along 2nd-level routes.

Our assumptions and settings for the HTRD system are as follows. We consider two types of customers: *truck customers* and *robot-only customers*. The former can be visited by either the truck or the robot, and the latter can only be visited by robots. Robot-only customers are located in areas where trucks cannot go.

We also assume each customer must be visited by exactly one truck (including a truck that carries its robot) or robot once. Note that each customer has a service time and time window. We assume parking nodes are used for the truck to drop off/pick up and replenish its onboard robot, and hubs are used to store robots and goods. Each hub can only be visited once by a truck. Each hub contains a maximum number of robots, and it can also act as a parking node. We assume that the depot has a sufficient number of trucks and that parking nodes can be visited multiple times.

We refer to the robots that are transported by trucks as *onboard robots*, and to the robots that are stationed at hubs as *hub robots*. These two kinds of robots belong to the same type but serve two distinct purposes. Each robot can visit multiple customers during a trip. Robots have a maximum battery capacity, which determines the distance they can travel, but there is no maximum travel distance restriction for trucks. There is a maximum capacity for trucks and robots. It is assumed that the combined weight of the truck and its robot must not exceed the truck's maximum capacity. Each truck has a designated onboard robot that matches with it, and no other truck can carry that robot. Onboard robots must be dropped off and picked up by the same truck. Onboard robots do not need to return to the parking node where they left their truck, but they can be picked up at any parking node or hub as long as the pickup is performed by their original truck. Onboard robots cannot serve customers directly from the depot and cannot return to the depot independently. However, onboard robots can be used multiple times, they can be replenished, and their batteries can be swapped by the corresponding truck at parking nodes or hubs. For hub robots, hub robots depart from hubs to serve customers and then must return to the hub where they left initially.

For recharging, we suppose that the robot's battery is changed every time a truck picks it up. We assume that a hub robot can be restocked with goods at hubs and that an onboard robot can be restocked with goods from its corresponding truck at parking nodes. Note that hubs can be replenished with goods from trucks, and trucks' freight must be loaded at the depot. We do not consider the possibility of a truck resupplying another truck or a robot resupplying another robot. We also assume that the goods from different trucks cannot be assigned to the same hub robot. We assume the time required for tasks such as dropping off, picking up, replenishing, and switching out a battery are fixed values and are represented as a single service time in our model.

Our objective is the minimization of the total cost, consisting of travel costs and labor costs. Travel costs include the travel costs of trucks and robots. Labor costs (also called time consumption costs) include driving costs and waiting costs. Waiting costs refer to the cost of drivers (trucks) waiting at nodes, corresponding to the waiting time multiplied by the waiting cost rate. Driving costs represent the cost for drivers (trucks) driving on the route, equal to the driving time multiplied by the driving cost rate. Note that we assume the waiting cost rate equals the driving cost rate. Hence we can use duration time multiplied by the waiting/driving cost rate to represent the labor costs.

Figure 1 shows an example of the problem. The pentagons stand for hubs, the square represents the depot, the triangles represent parking nodes, and the circles represent customer nodes. Solid lines show truck routes, while dotted lines show robot routes. Note that customer nodes in truck no-go areas can only be served by robots.

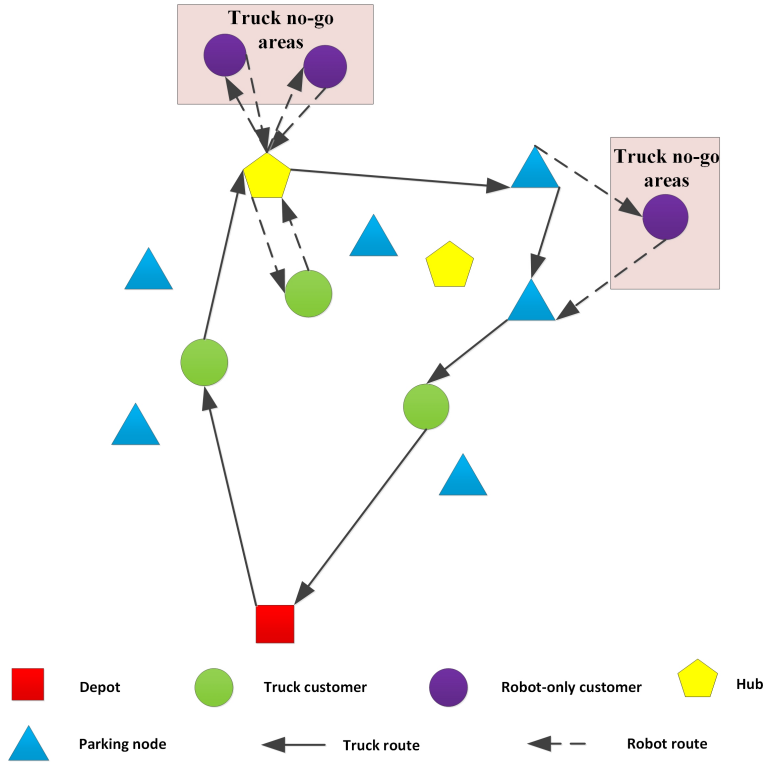


Figure 1: HTRD Example

2.2. *Variable and Parameter Definitions*

The hybrid truck-based robot delivery problem is defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$. The depot $\mathcal{V}_0$ is modeled by two nodes 0 and $0'$. Truck routes start at node 0 and end at node $0'$. The set $\mathcal{V}_s$ represents the hubs where robots are based. The set $\mathcal{V}_p$ corresponds to the parking nodes where trucks can perform drop off, pick up, replenishment, and robot battery swapping operations. Each parking node might be visited more than one time, we therefore introduce dummy nodes permitting to model multiple visits at parking nodes. The set $\mathcal{V}_r = \mathcal{V}_p \cup \mathcal{V}_s$ represents parking nodes and hubs.

The set $\mathcal{V}_c$ is the overall customer node set, while $\mathcal{V}_{c1}$ contains the truck customer nodes; note that $\mathcal{V}_{c1} \subseteq \mathcal{V}_c$. Let $\mathcal{V}_{sc} = \mathcal{V}_s \cup \mathcal{V}_c$ be the set of hubs and customers, $\mathcal{V}_{rc1} = \mathcal{V}_r \cup \mathcal{V}_{c1}$ denotes the set of parking nodes, hubs and truck customer nodes, and $\mathcal{V}^0_{rc1} = \mathcal{V}_r \cup \mathcal{V}_{c1} \cup \mathcal{V}_0$ represents the set of depot 0, parking nodes, hubs and truck customer nodes. $\mathcal{V}^0_r = \mathcal{V}_r \cup \mathcal{V}_0$, and $\mathcal{V}^0_c = \mathcal{V}_c \cup \mathcal{V}_0$. The truck routes correspond to the arc set $\mathcal{A}_1 = \{(0, j) \mid j \in \mathcal{V}_{rc1}\} \cup \{(i, j) \mid i, j \in \mathcal{V}_{rc1}, i \neq j\} \cup \{(i, 0') \mid i \in \mathcal{V}_{rc1}\}$ and $\mathcal{A}_2 = \{(i, j) \mid i \in \mathcal{V}_r; j \in \mathcal{V}_c\} \cup \{(i, j) \mid i \in \mathcal{V}_c; j \in \mathcal{V}_c, i \neq j\} \cup \{(i, j) \mid i \in \mathcal{V}_c; j \in \mathcal{V}_r\}$ is the arc set representing the robot routes. The arc set $\mathcal{A}_3 = \mathcal{A}_1 \cup \mathcal{A}_2$ corresponds to all possible routes and $\mathcal{A}_4 = \mathcal{A}_3 \setminus \mathcal{A}_1$ corresponds to routes not reachable by a truck. The arc set $\mathcal{A}_5 = \{(i, j) \mid i \in \mathcal{V}_s, j \in \mathcal{V}_c\} \cup \{(i, j) \mid i \in \mathcal{V}_c, j \in \mathcal{V}_c, i \neq j\} \cup \{(i, j) \mid i \in \mathcal{V}_c, j \in \mathcal{V}_s\}$ represents the potential routes of hub robots.

K is the number of trucks, and $k \in \mathcal{K} = 1 \ldots K$ stands for the *kth* truck. Given the one-to-one correspondence between trucks and onboard robots, $\mathcal{K}$ also correlates to the set of onboard robots. We also let $\mathcal{L} = \{1, .., l, .., L\}$ be the subset of hub robots belonging to a truck, where L is a maximum number of hub robots assignable to a truck.

The cost function is composed of the unit travel costs of trucks c_1 and robots c_2; c_t denotes the time consumption cost rate, respectively. The travel speed for trucks is v_1 and the speed for robots is v_2. Each edge has a corresponding distance d_{ij}, thereby $c_1 d_{ij}$ and $c_2 d_{ij}$ are the respective travel costs for trucks an robots. This holds similarly for travel times. The goods stored in the depot $\{0\}$ are to be delivered to customers i with corresponding demands q_i and a service time s_i. Customer time windows are represented by $[a_i, b_i]$ for each customer $i \in \mathcal{V}_c$, designating the time interval where

service at customer i must begin. The earliest possible departure time from the depot a_0 and the latest possible arrival time at the depot $b_{0'}$ define the overall time horizon of the problem at at hand. All considered time windows are hard.

The characteristics of trucks and robots are given by their respective load capacities C_v and C_r. For simplicity, we assume that the total load of the truck and its onboard robots are not allowed to exceed C_v if the robot is in the truck. The robot battery capacity is Q while the battery consumption rate is given by g. N_h is the maximum number of robots in a hub.

We define the decision variables required for the HTRD model as follows. Variables x_{ijk} are set to one if arc (i, j) in $\mathcal{A}_1$ is used by the k^{th} truck. Variables y_{ijk} are set to one if arc (i, j) in $\mathcal{A}_3$ is used by the k^{th} onboard robot. Variables z_{ijk} are set to one if arc (i, j) in $\mathcal{A}_1$ is used by the k^{th} truck with its robot on board. Variables y_{ij}^{kl} are set to one if arc (i, j) in $\mathcal{A}_5$ is used by the l^{th} hub-robot belonging to the k^{th} truck. Variables W_i^k and w_i^k represent arrival times for the k^{th} truck and k^{th} onboard robot at node i. The variables w_i^{kl} represent arrival times for the l^{th} hub robot of the k^{th} truck at node i. Variables e_i^k correspond to the remaining battery charge level of the onboard robot at its arrival at node i. Variables F_{ijk} correspond to the freight flow of the truck traveling along arc (i, j) in $\mathcal{A}_1$. If a robot is carried by the truck, F_{ijk} corresponds to the freight flow of the truck including its robot. Variables f_{ijk} represent the freight flow in arc (i, j) in $\mathcal{A}_3$ of the onboard robot belonging to k^{th} truck. Variables f_{ij}^{kl} represent the freight flow in arc (i, j) in $\mathcal{A}_5$ of the l^{th} hub robot belonging to k^{th} truck.

The variable and parameter definitions used in this paper are also shown in Table 1.

2.3. Hybrid Truck-based Robot Delivery Model

The hybrid truck-based robot delivery problem is modeled as the following Mixed Integer Program (MIP):

(1) Objective function

$$\min\Big(\sum_{k\in\mathcal{K}}\sum_{(i,j)\in\mathcal{A}_1} c_1 d_{ij} x_{ijk} + \sum_{k\in\mathcal{K}}\sum_{(i,j)\in\mathcal{A}_3} c_2 d_{ij} y_{ijk} - \sum_{k\in\mathcal{K}}\sum_{(i,j)\in\mathcal{A}_1} c_2 d_{ij} z_{ijk} + \sum_{k\in\mathcal{K},l\in\mathcal{L}}\sum_{(i,j)\in\mathcal{A}_5} c_2 d_{ij} y_{ij}^{kl} + \sum_{k\in\mathcal{K}} c_t W_{0'}^k\Big) \tag{1}$$

The whole cost is minimized by the objective function (1). It is equivalent to the travel costs (travel costs of trucks and robots) plus the labor costs (driving costs and waiting costs). Note that in the calculation of travel cost, we double calculate the cost when robots are on a truck, so we use a minus sign in the objective function to subtract this distance.

(2) Arc constraints

$$\sum_{(i,j)\in\mathcal{A}_1} x_{ijk} \le 1, \qquad \forall j \in \mathcal{V}_{rc1}, k \in \mathcal{K} \tag{2}$$

$$\sum_{(i,j)\in\mathcal{A}_1} x_{ijk} - \sum_{(j,i)\in\mathcal{A}_1} x_{jik} = 0, \qquad \forall j \in \mathcal{V}_{rc1}, k \in \mathcal{K} \tag{3}$$

$$\sum_{(i,j)\in\mathcal{A}_3} y_{ijk} \le 1, \qquad \forall j \in \mathcal{V}_{rc}, k \in \mathcal{K} \tag{4}$$

$$\sum_{(i,j)\in\mathcal{A}_3} y_{ijk} - \sum_{(j,i)\in\mathcal{A}_3} y_{jik} = 0, \qquad \forall j \in \mathcal{V}_{rc}, k \in \mathcal{K} \tag{5}$$

$$\sum_{(i,j)\in\mathcal{A}_5} y_{ij}^{kl} \le 1, \qquad \forall j \in \mathcal{V}_{sc}, k \in \mathcal{K}, l \in \mathcal{L} \tag{6}$$

Table 1: Variables and Parameters

$\mathcal{V}_0$	The depot is modeled using a start depot 0 and an end depot $0'$.
$\mathcal{V}_p$	Parking nodes
$\mathcal{V}_s$	Robot hubs
$\mathcal{V}_r$	Set of parking nodes and hubs, $\mathcal{V}_r = \mathcal{V}_p \cup \mathcal{V}_s$
$\mathcal{V}_c$	Customers
$\mathcal{V}_{sc}$	Hubs and customers
$\mathcal{V}_{c1}$	Truck customers, $\mathcal{V}_{c1} \in \mathcal{V}_c$
$\mathcal{V}_{rc1}$	Parking nodes, robot hubs and truck customers, $\mathcal{V}_{rc1} = \mathcal{V}_r \cup \mathcal{V}_{c1}$
$\mathcal{V}^0_{rc1}$	Set of depot 0, parking nodes, hubs and truck customer nodes, $\mathcal{V}^0_{rc1} = \mathcal{V}_{rc1} \cup \mathcal{V}_0$
$\mathcal{V}^0_r$	Set of depot 0, parking nodes, and hubs, $\mathcal{V}^0_r = \mathcal{V}_r \cup \mathcal{V}_0$
$\mathcal{V}^0_c$	Set of depot 0 and customer nodes, $\mathcal{V}^0_c = \mathcal{V}_c \cup \mathcal{V}_0$
$\mathcal{A}_1$	1^{st}-level route (truck routes), $\mathcal{A}_1 = \{(0, j) \mid j \in \mathcal{V}_{rc1}\} \cup \{(i, j) \mid i, j \in \mathcal{V}_{rc1}, i \neq j\} \cup \{(i, 0') \mid i \in \mathcal{V}_{rc1}\}$
$\mathcal{A}_2$	2^{nd}-level route (robot routes), $\mathcal{A}_2 = \{(i, j) \mid i \in \mathcal{V}_r; j \in \mathcal{V}_c\} \cup \{(i, j) \mid i \in \mathcal{V}_c; j \in \mathcal{V}_c, i \neq j\} \cup \{(i, j) \mid i \in \mathcal{V}_c; j \in \mathcal{V}_r\}$
$\mathcal{A}_3$	All possible routes, $\mathcal{A}_3 = \mathcal{A}_1 \cup \mathcal{A}_2$
$\mathcal{A}_4$	Set of arcs corresponding to unreachable routes for trucks, $\mathcal{A}_4 = \mathcal{A}_3 \setminus \mathcal{A}_1$
$\mathcal{A}_5$	Arc set corresponding to routes that the hub robot in hubs can travel, $\mathcal{A}_5 = \{(i, j) \mid i \in \mathcal{V}_s, j \in \mathcal{V}_c\} \cup \{(i, j) \mid i \in \mathcal{V}_c, j \in \mathcal{V}_c, i \neq j\} \cup \{(i, j) \mid i \in \mathcal{V}_c, j \in \mathcal{V}_s\}$
$\mathcal{K}$	The trucks, $\mathcal{K} = \{1, 2, .., k, .., K\}$, with K being the number of trucks, and $k \in \mathcal{K}$ representing the k^{th} truck. Since there is a one-to-one correspondence between trucks and carried robots, this set effectively represents the onboard robots.
$\mathcal{L}$	Subset of hub robots belonging to a truck, $\mathcal{L} = \{1, .., l, .., L\}$, where L is a maximum number of hub robots assignable to a truck
d_{ij}	Distance between nodes i and j
g	Robot battery consumption rate
c_1	Truck travel cost
c_2	Robot travel cost
c_t	Time consumption cost rate
Q	Robot battery capacity
C_v	Truck cargo capacity
C_r	Robot cargo capacity
N_h	Maximum number of robots in a hub
v_1	Truck speed
v_2	Robot speed
q_i	Demand at node i
s_i	Service time at node i
$[a_i, b_i]$	Customer time window at node i, a_0 represents the earliest possible departure time from the depot 0, and $b_{0'}$ represents the corresponding latest possible arrival time at $0'$
x_{ijk}	Variable set to one if arc (i, j) in $\mathcal{A}_1$ is used by the k^{th} truck, 0 otherwise
y_{ijk}	Variable set to one if arc (i, j) in $\mathcal{A}_3$ is used by the k^{th} onboard robot, 0 otherwise
z_{ijk}	Variable set to one if arc (i, j) in $\mathcal{A}_1$ is used by the k^{th} truck with its robot on board, 0 otherwise
y^{kl}_{ij}	Variable set to one if arc (i, j) in $\mathcal{A}_5$ is used by the l^{th} hub robot belonging to k^{th} truck, 0 otherwise
W^k_i	Arrival time for the k^{th} truck at node i
w^k_i	Arrival time for the k^{th} onboard robot at node i
w^{kl}_i	Arrival time for the l^{th} hub robot belonging to k^{th} truck at node i
e^k_i	Remaining battery charge of the onboard robot of k^{th} truck on arrival at node i
F_{ijk}	Freight flow of the truck in arc (i, j) in $\mathcal{A}_1$. If there is a robot on board the truck, F_{ijk} represents the freight flow of the truck and its onboard robot.
f_{ijk}	Freight flow in arc (i, j) in $\mathcal{A}_3$ of the onboard robot belonging to k^{th} truck
f^{kl}_{ij}	Freight flow in arc (i, j) in $\mathcal{A}_5$ of the l^{th} hub-robot belonging to k^{th} truck

$$\sum_{(i,j)\in\mathcal{A}_5} y_{ij}^{kl} - \sum_{(j,i)\in\mathcal{A}_5} y_{ji}^{kl} = 0, \qquad \forall j \in \mathcal{V}_{sc}, k \in \mathcal{K}, l \in \mathcal{L} \tag{7}$$

$$\sum_{l\in\mathcal{L}} \sum_{(i,j)\in\mathcal{A}_5} y_{ij}^{kl} \leq L \sum_{(i,j)\in\mathcal{A}_1} x_{ijk}, \qquad \forall i \in \mathcal{V}_s, k \in \mathcal{K} \tag{8}$$

$$\sum_{i\in\mathcal{V}_{rc1}} x_{i0'k} = \sum_{j\in\mathcal{V}_{rc1}} x_{0jk} = \sum_{i\in\mathcal{V}_{rc1}} y_{i0'k} = \sum_{j\in\mathcal{V}_{rc1}} y_{0jk} \leq 1, \qquad \forall k \in \mathcal{K} \tag{9}$$

$$\sum_{k\in\mathcal{K}} \left(\sum_{(i,j)\in\mathcal{A}_2} (y_{ijk} - z_{ijk}) + \sum_{(i,j)\in\mathcal{A}_1} x_{ijk} \right) + \sum_{k\in\mathcal{K}, l\in\mathcal{L}} \sum_{(i,j)\in\mathcal{A}_5} y_{ij}^{kl} = 1, \qquad \forall j \in \mathcal{V}_c \tag{10}$$

$$\sum_{(i,j)\in\mathcal{A}_3} y_{ijk} + \sum_{(i,j)\in\mathcal{A}_1} x_{ijk} - \sum_{(i,j)\in\mathcal{A}_1} z_{ijk} \leq 1, \qquad \forall i \in \mathcal{V}_{c1}^0, k \in \mathcal{K} \tag{11}$$

$$\sum_{(i,j)\in\mathcal{A}_3} y_{ijk} + \sum_{(i,j)\in\mathcal{A}_1} x_{ijk} - \sum_{(i,j)\in\mathcal{A}_1} z_{ijk} \leq 1, \qquad \forall j \in \mathcal{V}_{c1}^{'}, k \in \mathcal{K} \tag{12}$$

$$\sum_{(i,j)\in\mathcal{A}_3} y_{ijk} \leq \sum_{(i,j)\in\mathcal{A}_1} x_{ijk}, \qquad \forall i \in \mathcal{V}_r, k \in \mathcal{K} \tag{13}$$

$$2z_{ijk} \leq x_{ijk} + y_{ijk} \leq z_{ijk} + 1, \qquad \forall (i,j) \in \mathcal{A}_1, k \in \mathcal{K} \tag{14}$$

$$x_{ijk}, y_{ijk}, z_{ijk} \in \{0,1\}, \qquad \forall (i,j) \in \mathcal{A}_3, k \in \mathcal{K} \tag{15}$$

$$y_{ij}^{kl} \in \{0,1\}, \qquad \forall (i,j) \in \mathcal{A}_5, k \in \mathcal{K}, l \in \mathcal{L} \tag{16}$$

$$x_{ijk} = 0, z_{ijk} = 0, \qquad \forall (i,j) \in \mathcal{A}_4, k \in \mathcal{K} \tag{17}$$

The constraints (2)-(17) are general arc constraints for our problem. Constraints (2)-(5) make sure that every node is visited no more than once and that every vehicle that enters a node leaves it as well. Constraints (6)-(8) are arc constraints for hub-robot routes departing from hubs. Constraints (6)-(7) state that each customer node and hub are visited no more than once and that every vehicle that enters a node also leaves it. Constraints (8) state that the hub robots can start to serve customers only once a truck arrives at the hub. Constraints (9) make sure that the one-to-one relationship between trucks and robots holds by forcing their indices to be equal when they leave and return to the depot. Constraints (10) make sure that every customer node is visited exactly once by either a truck or a robot. Constraints (11)-(12) require for customer nodes and the depot that they can be either visited by truck or its onboard robot but not by both, except if the robot is onboard the truck. Also, they cannot leave the customer node separately. Constraints (13) make sure that robots cannot visit a parking node except if their corresponding truck visits it as well. Constraints (14) set z_{ijk} to one if both x_{ijk} and y_{ijk} are equal to one, meaning that arc (i, j) in $\mathcal{A}_1$ is used by the k^{th} truck together with its onboard robot. The decision variables are defined in (15)-(17).

(3) Time constraints

$$W_i^k + s_i + d_{ij}/v_1 \times x_{ijk} - W_j^k \leq (b_0^{'} + s_i)(1 - x_{ijk}), \qquad \forall \{i \in \mathcal{V}_{c1} | (i,j) \in \mathcal{A}_1\}, k \in \mathcal{K} \tag{18}$$

$$w_i^k + s_i + d_{ij}/v_2 \times (y_{ijk} - x_{ijk}) - w_j^k \leq (b_0^{'} + s_i)(1 - y_{ijk} + x_{ijk}), \qquad \forall \{i \in \mathcal{V}_c | (i,j) \in \mathcal{A}_3\}, k \in \mathcal{K} \tag{19}$$

$$max(w_i^k, W_i^k) + s_i + d_{ij}/v_1 \times x_{ijk} - W_j^k \leq (b_0^{'} + s_i)(1 - x_{ijk}), \qquad \forall \{i \in \mathcal{V}_r^0 | (i,j) \in \mathcal{A}_1\}, k \in \mathcal{K} \tag{20}$$

$$max(W_i^k, w_i^k) + s_i + d_{ij}/v_2 \times (y_{ijk} - x_{ijk}) - w_j^k \leq (b_0^{'} + s_i)(1 - y_{ijk} + x_{ijk}), \qquad \forall \{i \in \mathcal{V}_r^0 | (i,j) \in \mathcal{A}_3\}, k \in \mathcal{K} \tag{21}$$

$$-b_0^{'} \left(1 - \sum_{(i,j)\in\mathcal{A}_1} z_{ijk}\right) \leq W_j^k - w_j^k \leq b_0^{'} \left(1 - \sum_{(i,j)\in\mathcal{A}_1} z_{ijk}\right), \qquad \forall j \in \mathcal{V}_{rc1}^{'}, k \in \mathcal{K} \tag{22}$$

$$W_i^k + s_i + d_{ij}/v_2 \times y_{ij}^{kl} - w_j^{kl} \leq (b_0^{'} + s_i)(1 - y_{ij}^{kl}), \qquad \forall i \in \mathcal{V}_s, j \in \mathcal{V}_c, k \in \mathcal{K}, l \in \mathcal{L} \tag{23}$$

$$w_i^{kl} + s_i + d_{ij}/v_2 \times y_{ij}^{kl} - w_j^{kl} \leq (b_0^{'} + s_i)(1 - y_{ij}^{kl}), \qquad \forall i \in \mathcal{V}_c, j \in \mathcal{V}_c, i \neq j, k \in \mathcal{K}, l \in \mathcal{L} \tag{24}$$

$$a_i \leq W_i^k, w_i^k, w_i^{kl} \leq b_i, \qquad \forall i \in \mathcal{V}_c^0, k \in \mathcal{K}, l \in \mathcal{L} \tag{25}$$

The constraints (18)-(24) are time-flow conservation constraints used to calculate truck and robot arrival times at each node, the duration involved in the objective function, and for eliminating subtours in truck and robot routes. Constraints (18)-(19) represent the time-flow related to the truck and onboard robot departures from customers. Constraints (18) are used for modeling the time flow of truck departures from truck customers. Constraints (19) represent the time-flow of onboard robot departures from customers. Constraints (20)-(21) are modeling the truck and robot departures from parking nodes, hubs or the depot. Constraints (20) represent truck route timings, and (21) robot route timings. Note that constraints (20)-(21) enforce the time synchronization of trucks and their corresponding onboard robots. Constraints (22) force arrival times of trucks to be equal to that of their corresponding robots if they are onboard. Constraints (23)-(24) ensure the time-flow of the hub-robot routes. Constraints (25) enforce time windows. Note constraints (22) is implemented by linearizing the absolute value constraints $|W_j^k - w_j^k| \leq b_0^{'} \left(1 - \sum_{(i,j)\in\mathcal{A}_1} z_{ijk}\right), \forall j \in \mathcal{V}_{rc1}^{'}, k \in \mathcal{K}$.

(4) Freight constraints

$$\sum_{(i,j)\in\mathcal{A}_1} F_{ijk} - \sum_{(j,i)\in\mathcal{A}_1} F_{jik} = q_j \sum_{(i,j)\in\mathcal{A}_1} x_{ijk}, \qquad \forall j \in \mathcal{V}_{c1}, k \in \mathcal{K} \tag{26}$$

$$\sum_{(i,j)\in\mathcal{A}_2} f_{ijk} - \sum_{(j,i)\in\mathcal{A}_2} f_{jik} = q_j \sum_{(i,j)\in\mathcal{A}_2} (y_{ijk} - z_{ijk}), \qquad \forall j \in \mathcal{V}_c, k \in \mathcal{K} \tag{27}$$

$$\sum_{(i,j)\in\mathcal{A}_5} f^{kl}_{ij} - \sum_{(j,i)\in\mathcal{A}_5} f^{kl}_{ji} = q_j \sum_{(i,j)\in\mathcal{A}_5} y^{kl}_{ij}, \qquad \forall j \in \mathcal{V}_c, k \in \mathcal{K}, l \in \mathcal{L} \tag{28}$$

$$\sum_{(i,j)\in\mathcal{A}_1} F_{ijk} - \sum_{(j,i)\in\mathcal{A}_1} F_{jik} = \sum_{(j,i)\in\mathcal{A}_3} f_{jik} - \sum_{(i,j)\in\mathcal{A}_3} f_{ijk}, \qquad \forall j \in \mathcal{V}_p, k \in \mathcal{K} \tag{29}$$

$$\sum_{k\in\mathcal{K}} \left(\sum_{(i,j)\in\mathcal{A}_1} F_{ijk} - \sum_{(i,j)\in\mathcal{A}_1} F_{jik} \right) = \sum_{k\in\mathcal{K}} \left(\sum_{(j,i)\in\mathcal{A}_3} f_{jik} - \sum_{(i,j)\in\mathcal{A}_3} f_{ijk} \right) + \sum_{k\in\mathcal{K}, l\in\mathcal{L}} \sum_{(j,i)\in\mathcal{A}_5} f^{kl}_{ji}, \qquad \forall j \in \mathcal{V}_s \tag{30}$$

$$\sum_{k\in\mathcal{K}, l\in\mathcal{L}} \sum_{j\in\mathcal{V}_c} y^{kl}_{ij} \le N_h, \qquad \forall i \in \mathcal{V}_s \tag{31}$$

$$0 \le F_{ijk} \le C_v x_{ijk}, \qquad \forall (i,j) \in \mathcal{A}_1, k \in \mathcal{K} \tag{32}$$

$$0 \le f_{ijk} \le C_r(1 - z_{ijk}), \qquad \forall (i,j) \in \mathcal{A}_1, k \in \mathcal{K} \tag{33}$$

$$0 \le f_{ijk} \le C_r(y_{ijk} - z_{ijk}), \qquad \forall (i,j) \in \mathcal{A}_2, k \in \mathcal{K} \tag{34}$$

$$0 \le f^{kl}_{ij} \le C_r y^{kl}_{ij}, \qquad \forall (i,j) \in \mathcal{A}_5, k \in \mathcal{K}, l \in \mathcal{L} \tag{35}$$

Constraints (26)-(30) model freight flows. Constraints (26) represent the conservation of freight flows of trucks leaving truck customer nodes. Constraints (27) represent the freight flow conservation for an onboard robot departing from a customer node. Constraints (28) represent the freight flow conservation for a hub robot departing from a customer node. Constraints (29) enforce freight flow conservation at parking nodes. Constraints (30) ensure freight flow conservation at hubs. Constraints (31) enforce for each hub that its capacity is not exceeded. Variables are defined in (32)-(35).

(5) Energy constraints

$$e^k_i + d_{ij} g (y_{ijk} - x_{ijk}) - e^k_j \le Q(1 - y_{ijk} + x_{ijk}), \qquad \forall \{i \in \mathcal{V}_{rc} | (i,j) \in \mathcal{A}_2\}, k \in \mathcal{K} \tag{36}$$

$$\sum_{(i,j)\in\mathcal{A}_5} d_{ij} g y^{kl}_{ij} \le Q, \qquad \forall k \in \mathcal{K}, l \in \mathcal{L} \tag{37}$$

$$0 \le e^k_i, e^{kl}_i \le Q, \qquad \forall i \in \mathcal{V}_{rc}, k \in \mathcal{K}, l \in \mathcal{L} \tag{38}$$

The constraints (36)-(38) model energy utilization. Constraints (36) refer to the energy of onboard robots, while constraints (37) ensure that the energy consumption of hub robots does not exceed their energy capacity. Energy variables are defined in constraints (38).

2.4. Valid Inequalities

We also propose symmetry-breaking constraints and time-based lower bounds as valid inequalities, following Tamke and Buscher (2021), Roberti and Wen (2016). These are meant to strengthen our MIP formulation.

(1) Symmetry breaking constraints

$$\sum_{(i,j)\in\mathcal{A}_1} x_{ij(k+1)} \le \sum_{(i,j)\in\mathcal{A}_1} x_{ijk}, \qquad \forall k \in \{1, 2, ..., K-1\} \tag{39}$$

$$\sum_{(i,j)\in\mathcal{A}_5} y^{k(l+1)}_{ij} \le \sum_{(i,j)\in\mathcal{A}_5} y^{kl}_{ij}, \qquad \forall k \in \mathcal{K}, l \in \{1, 2, ..., L-1\} \tag{40}$$

$$x_{ijk} + x_{jik} \le 1, \qquad \forall \{i \in \mathcal{V}_{rc1}, j \in \mathcal{V}_{rc1} | (i,j) \in \mathcal{A}_1\}, k \in \mathcal{K} \tag{41}$$

$$y_{ijk} + y_{jik} \le 1, \qquad \forall \{i \in \mathcal{V}_{rc}, j \in \mathcal{V}_{rc} | (i,j) \in \mathcal{A}_3\}, k \in \mathcal{K} \tag{42}$$

Symmetry-breaking constraints eliminate symmetries between variables and reduce the search space size. Constraints (39)-(40) sort by the number of access nodes to break symmetric with identical values for objectives but with different tasks to vehicle assignments. Constraints (41)-(42) can be added to speed up solving the model in a general-purpose MILP solver with trivial cuts (Roberti and Wen 2016).

(2) Lower bounds based on time

$$\sum_{(i,j)\in A_1} (d_{ij}/v_1 + s_i)x_{ijk} \le W^k_{0'}, \qquad \forall k \in \mathcal{K} \tag{43}$$

$$\sum_{(i,j)\in A_1} (d_{ij}/v_1 + s_i)z_{ijk} + \sum_{(i,j)\in A_2} (d_{ij}/v_2 + s_i)(y_{ijk} - z_{ijk}) \le W^k_{0'}, \qquad \forall k \in \mathcal{K} \tag{44}$$

$$(a_i + s_i + d_{i0'}/v_1)x_{i0'k} \le W^k_{0'}, \qquad \forall i \in V_{c1}, k \in \mathcal{K} \tag{45}$$

Constraints (43)-(45) are lower bounds on time. Constraints (43) ensure that each truck's arrival time at the depot is equal to or greater than the amount of time required for it to go along its route, including the amount of time needed to serve all truck customers (Schermer et al. 2019). Similarly, constraints (44) state the time lower bound for an onboard robot. Constraints (45) ensure that each truck's arrival time at the depot is equal to or greater than the sum of the start time window for any of the customers on the visit plus the service time and the time the truck travels from that customer to the depot.

3. Related literature

In the literature review, we first present the related work on the two-echelon vehicle routing problem. We then address the related papers on the truck-based robot delivery model. Finally, we discuss the vehicle routing problem with truck no-go areas.

3.1. Two-Echelon Vehicle Routing Problem

The two-echelon vehicle routing problem (2E-VRP) is a version of the vehicle routing problem (VRP) that involves finding the best routes for a distribution system with two echelons. In the first echelon, goods are delivered from a depot to intermediary hubs. In the second echelon, goods are transported from the hubs to their final destination. To ensure cost-effectiveness, large trucks are used to complete first-echelon deliveries, while smaller vehicles are used for the second-echelon distribution (Sluijk et al. 2023). Compared with the 2E-VRP, in our HTRD model, trucks are not only used for transporting goods but also for transporting robots. The HTRD model has parking nodes, which can be used to release and retrieve robots through trucks, making it more flexible than the 2E-VRP model.

Some scholars focus on different types of the 2E-VRP. For example, Li et al. (2020, 2021) study the 2E-VRP under a variety of hub synchronization restrictions. Liu et al. (2021) further study the 2E-VRP that involves goods transshipment and multi-depot. In this model, trucks and delivery robots are assigned to serve the two distinct echelon. Wang et al. (2021) investigate the collaborative multi-depot multi-period based 2E-VRP. Dellaert et al. (2019) investigate the 2E-VRPTW that arises in city logistics, their model mainly deals with the coordination of goods and time between trucks and the second-level vehicles at hubs, and they propose an exact method based on the branch-and-price framework. Dellaert et al. (2021) then study the multi-commodity 2E-VRPTW, considering customer-specific origins to destinations, and non-substitutable demands. Fontaine et al. (2021) enhance a two-echelon city logistics system by incorporating inbound and outward requirements, various modes of transportation, and mass transit vehicles with traditional, road-based carriers.

Existing research also considers using robots for second-echelon delivery. Bakach et al. (2021) propose a two-echelon delivery model. At the first level, a truck transports packages from a large depot to small local robot hubs that maintain a predetermined number of robots. At the second level, these robots deliver items to customers who may or may not have time windows using pendulum tours. Their goal is to reduce the number of hubs and robot mileage required to serve all customers, considering the robot's limited range due to the battery size. Alfandari et al. (2022) address an operational problem of determining an optimal route for a single vehicle carrying customer parcels from a central depot to a set of hubs. Autonomous vehicles such as robots are launched from these hubs to perform last-mile distribution. The objectives of min-max tardiness, minimal total tardiness, and the minimum number of late deliveries are considered, and a branch-and-benders-cut method is proposed for the three objective functions.

3.2. Truck-Based Robot Delivery Problem

Several intelligent delivery concepts have recently been proposed in order to reduce the negative impacts of freight transport in urban areas. Among those, truck-based robot delivery has attracted some interest. In the truck-based robot delivery problem, the truck performs as a mobile platform for carrying, releasing, and retrieving robots, increasing the flexibility of robot delivery by overcoming range limitations. Some relevant surveys have already discussed this type of problem in detail. For example, Srinivas et al. (2022) classifies the truck-based robot delivery problem into three models: dispatch–wait–collect mode: The truck waits at the same drop-off site for the robots to return; dispatch–move–collect mode: The truck continues its route and later picks up the deployed robots; dispatch–leave mode: The truck continues its route without returning to collect the robots. Alverhed et al. (2024) categorize the problem into four models: direct truck-based robots, flexible truck-based robots, cyclic truck-based robots, and depot-assisted truck-based robots. In

contrast, this paper takes a different approach, classifying truck-based robot delivery problems into two categories based on whether or not the truck itself participates in the delivery tasks within the distribution system.

In the first variant, trucks do not serve customers directly, called separate truck-based robot deliveries. The concept of separate truck-based robot deliveries involves using robots to make the final delivery to customers while utilizing trucks as a "mothership" to transport and release the robots. This model allows for a reduction in truck mileage and an increase in driver productivity, making it an attractive option for both cost and environmental impact. Yu et al. (2020) study a separate truck-based robot delivery model with trucks that transport the robots from the depot to parking nodes, where they are dropped off and sent to make deliveries. When the robots have finished their deliveries, they return to the parking nodes, where they can be picked up by the truck and either loaded for transport back to the depot or released to make further deliveries. Boysen et al. (2018) and Ostermeier et al. (2022) studied a model called separate truck-based robot delivery with robot depot. It is based on a network of robot depots and parking nodes. In both location types, robots can be released for delivery. The difference is that parking nodes only release robots, while robot depots allow robots to be released and new ones collected for later deliveries.

In the second variant, trucks can serve customers directly, called mixed truck-based robot deliveries. There are a variety of circumstances in which a person must make the final delivery rather than using a robot. This may be because specific customers, such as elderly individuals, are unwilling to interact with a robot to retrieve their goods. In addition, some individual orders may be too large for the robot compartment, such as delivering home appliances like televisions and refrigerators. Researchers have proposed a delivery system that utilizes humans and robots to deliver goods more efficiently. By employing both humans and robots for delivery, mixed truck-based robot deliveries enhance the flexibility and adaptability of the delivery system. However, this approach also introduces higher complexity to the routing process. Several studies have investigated this problem, such as Simoni et al. (2020), Chen et al. (2021), Heimfarth et al. (2022), Yu et al. (2024), and Yu et al. (2022). Note Simoni et al. (2020) and Chen et al. (2021) investigate the truck-based robot delivery model without parking nodes. Simoni et al. (2020) examine the version where one truck carries one robot without time window constraints, and Chen et al. (2021) explore the version where each truck carries several robots that can be deployed to serve multiple customers at their parking nodes while the truck is serving other customers. However, each robot can only visit one customer per trip, and the truck has to wait until all robots have been collected.

This research particularly extends the work of Yu et al. (2024) and Yu et al. (2022) by adding the hub attribute, incorporating the logistics network of 2E-VRP that is widely used in the real world, and by presenting an ALNS algorithm with a novel adaptive mechanism to solve the new problem. This research is also different from the previous studies of Boysen et al. (2018), Ostermeier et al. (2022), and Heimfarth et al. (2022) on the truck-based robot delivery model with a robot depot. In their model, the robot depot can store robots but not goods; the truck releases and leaves robots but does not collect them; a single truck carries multiple robots, and the main concern is whether the vehicle can reach customers. Boysen et al. (2018) first introduced the concept of separate truck-based robot delivery with a robot depot, where a truck loads goods and robots from a depot and drives to the city center. The truck lets the loaded robots go to serve single customers. The robots deliver the goods to their assigned customers and return to the robot depots in the city center. The truck can reload the robots at these robot depots and let more of them go until all customers are served. Based on Boysen et al. (2018), Ostermeier et al. (2022) add the constraints of a limited robot fleet to their separate truck-based robot delivery with a robot depot model. Ostermeier et al. (2023) expand their previous model to a multi-truck form. Heimfarth et al. (2022) assume that the truck can also serve customers directly in their mixed truck-based robot delivery with robot depot model.

It should also be noted that truck-robot collaborative distribution has similarities with truck-drone collaborative distribution (Wang et al. 2024, Madani et al. 2024), but the main difference lies in the calculation of delivery distance. For drones, the delivery distance is usually measured by the Euclidean distance, while for robots, the delivery distance is usually measured by the Manhattan distance or real distance. Another difference is the speed of the delivery agents. Drones generally have higher speed than trucks, while robots generally have lower speed than trucks.

3.3. Vehicle Routing Problem with Truck No-Go Areas

Scholars have begun to study the impact of truck no-go areas on the efficiency of the distribution system. In the classical 2E-VRP setting, all customers are located in the truck no-go areas. However, in many cases, trucks are not banned in the whole city but only in specific parts, such as campuses, pedestrian zones, etc. Besides, some truck no-go areas are not forbidden the entire time but only for some periods. Some cities directly ban trucks from entering urban areas such as narrow streets etc.

Yu et al. (2022) study a truck-based robot last-mile pickup and delivery problem and conduct a case study and a sensitivity analysis on truck no-go areas. Results show that the truck no-go restriction can lead to a 15% objective function reduction in their experimental setting. Anderluh et al. (2021) consider a two-echelon VRP with customers in so-called "grey zones" (these are areas on the edge of the city center and the neighboring zones). On the one hand, deliveries in the city center are carried out by small vehicles respecting some access restrictions. On the other hand, deliveries outside these sections are performed by traditional vehicles for economic reasons. The presented computational experiments show the potential impacts of grey zones on customer assignment to the echelons.

The following articles study truck no-go areas that are only active at some predefined time periods. Muñuzuri et al. (2013) investigate the VRP with access time windows to limit access to some particular areas during some periods. They show how these regulations impose extra costs on the companies, forcing them to use more vehicles. Zhao et al. (2019) address the effects of time constraints on urban freight and local environments using a Beijing agricultural freight case study. The results show that time constraints lead to higher freight prices and slightly lower local emissions. Akyol and De Koster (2018) present a game-theoretic model to obtain efficient joint time windows for freight vehicles to enter the city center. Their model demonstrates that finding timeslots benefitting all stakeholders and improving overall satisfaction with the city is possible.

Existing research on truck no-go areas mainly focuses on the VRP, and there are few studies on the impact of truck no-go areas on the two-echelon network. We expand on the above research to conduct sensitivity analysis experiments, evaluate the effect of the size of truck no-go areas and the time window of the truck no-go areas, and thus establish a concept of heterogeneous time windows for customers.

4. Solution Approach

Although we propose valid inequalities to speed up solving the model, using exact methods for solving medium-scale truck-based robot delivery problems with complex constraints is incredibly challenging. For example, in Wang and Sheu (2019) it is shown that the truck-based drone delivery problem cannot be solved with more than ten nodes using a standard MIP solver. Even their branch-and-price algorithm cannot solve problems with more than 12 nodes. Hence, we propose an ALNS-based metaheuristic solution approach.

The ALNS algorithm is a commonly used algorithm for solving VRPs in industry and academia (Sacramento et al. 2019, Informs 2021), and excels in solving truck-based robot delivery problems (Yu et al. 2022). Thus, we extend a classical ALNS framework (see Yu et al. (2022)) to solve the HTRD, where we incorporate the visiting of multiple customers and time window constraints. Based on the original ALNS framework (see in Appendix A), in the following, we first present a construction heuristic allowing to obtain feasible solutions rapidly (Section 4.1). The construction heuristic is also applied to generate an initial solution for the ALNS approach. Section 4.2 proposes some problem-specific operators to improve the initial solution, and Section 4.3 proposes novel adaptive methods that take full advantage of the association relationship existing between destroy and repair operators, and taking into account the difficulty of achieving a better solution and the quality of the solution obtained.

4.1. Construction Heuristic

We adopt a construction heuristic to assign customers to hubs and parking nodes and then generate the initial routes (including both truck routes and robot routes), see Figure 2. First, we allocate robot-only customers to the nearest hub or parking node available. Second, we allocate truck customers to the nearest hub that can accommodate them within the maximum driving distance of the robot, until the hub reaches its capacity limit for robot routes. Third, we regard these hubs and parking nodes as "constrained customers". Fourth, we apply a modified nearest neighbor search algorithm (MNNS) to construct truck routes that visit the assigned hubs and parking nodes. The MNNS, proposed by Yu et al. (2020), selects the customer with the earliest possible start service time as the "nearest neighbor", considering both distance and time window information. The MNNS iteratively adds the nearest neighbor to the current route, subject to time window and capacity constraints for the truck, until no more customers can be served. Then, the truck returns to the depot to complete the route, and a new truck route is created if there are still customer nodes that have not been visited yet.

4.2. Operators

We draw on the destroy and repair operators proposed by Yu et al. (2022) for truck-based robot delivery problems. The destroy operators consist of (random/greedy) customer removal, parking-route removal, and (random/greedy) route destruction. The repair operators consist of route reconstruction, (random/greedy) customer insertion, and (random/greedy) parking-route insertion. The details of the above destroy and repair operators are shown in Appendix B. Since our problem contains hubs and there are multiple robots that can operate in a hub, generating multiple robot routes, we further propose a hub-routes closure operator and a hub-routes open operator.

1. The **random hub-routes closure** operator randomly chooses a hub-robot route and removes all of its customers.
2. The **greedy hub-routes closure** operator removes the customers of the hub-robot route with the largest cost reduction for a given route.
3. The **random hub-routes open** operator randomly begins a new robot route with one customer node from a hub.
4. The **greedy hub-routes open** operator begins a new robot route, including one customer node from a hub, with the least total travel cost increases.

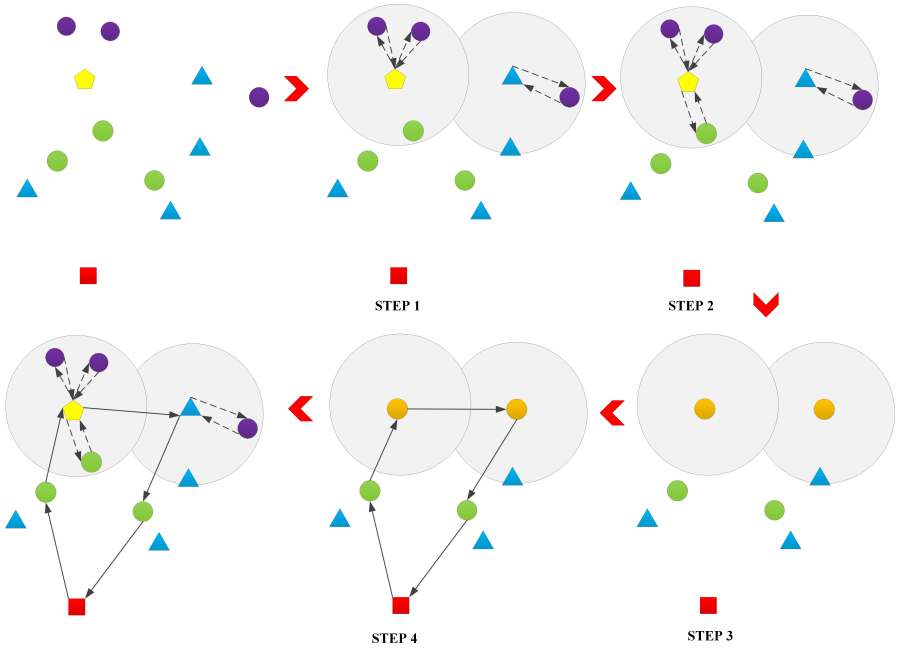


Figure 2: Construction Heuristic Diagram

4.3. Adaptive Methods

Classical ALNS methods usually perform well, and we want to see if adaptive methods that consider the connection between the destroy and repair operators will perform even better. Hence, we first introduce the classical adaptive method (AM1) and the destroy-repair combination weight-adjusted method (AM2) of ALNS. We design new adaptive strategies in which the destroy and repair operators are linked more closely (AM3). Finally, we propose a new adaptive method, which considers both the difficulty and the benefits of obtaining a better solution (AM4).

4.3.1. Classical Adaptive Methods (AM1)

In the classical adaptive methods ($AM1$), for the destroy and repair operators, a score ψ is used to adjust their respective weights. It is computed according to the following formula:

$$\psi = \begin{cases} w_1 & \text{if the new solution is a new global best solution,} \\ w_2 & \text{if the new solution is superior to the current solution,} \\ w_3 & \text{if the new solution is accepted,} \\ w_4 & \text{if the new solution is rejected.} \end{cases}$$

Normally, $w_1 \geq w_2 \geq w_3 \geq w_4 \geq 0$. The components corresponding to the selected destroy and repair operations in the ρ^- and ρ^+ vectors are updated using the following equations: $\rho_a^- = \lambda\rho_a^- + (1-\lambda)\psi$, $\rho_b^+ = \lambda\rho_b^+ + (1-\lambda)\psi$, in which a and b are the indices of the destroy and repair methods that were used in the last iteration of the algorithm, respectively. The components corresponding to the selected destroy and repair methods in the ρ^- and ρ^+ vectors are updated using the equations. The parameter $\lambda \in [0, 1]$ is the decay parameter that controls how sensitive the weightings are to changes in the performance of the destroy and repair methods. The flow chart of $AM1$ is in Figure 3, and the details can be found in Ropke and Pisinger (2006).

4.3.2. Destroy-Repair Combination Weight-Adjusted (AM2)

Now, let us introduce weight-adjusted combinations for the destroy and repair operators. For example, n destroy operators, and m repair operators will generate $n \times m$ destroy-repair combinations, and we need to assign corresponding weights. Then, we select a couple of destroy-repair operators to run the ALNS algorithm and adjust the corresponding weight of the destroy-repair operators according to the algorithm output. More specifically, the elements in the ρ vectors corresponding to the selected destroy-repair operator combinations are updated using the following equation: $\rho_a = \lambda\rho_a + (1-\lambda)\psi$, in which a is the index of the destroy-repair operator combination applied in the algorithm's last iteration. The flow chart of $AM2$ is in Figure 4.

4.3.3. Destroy-Based Repair Weight-Adjusted (AM3)

Finally, let us investigate destroy-based repair weight-adjusted operators, where we select a destroy operator and then choose a repair operator based on the specific destroy operator's information. In case there are n destroy operators

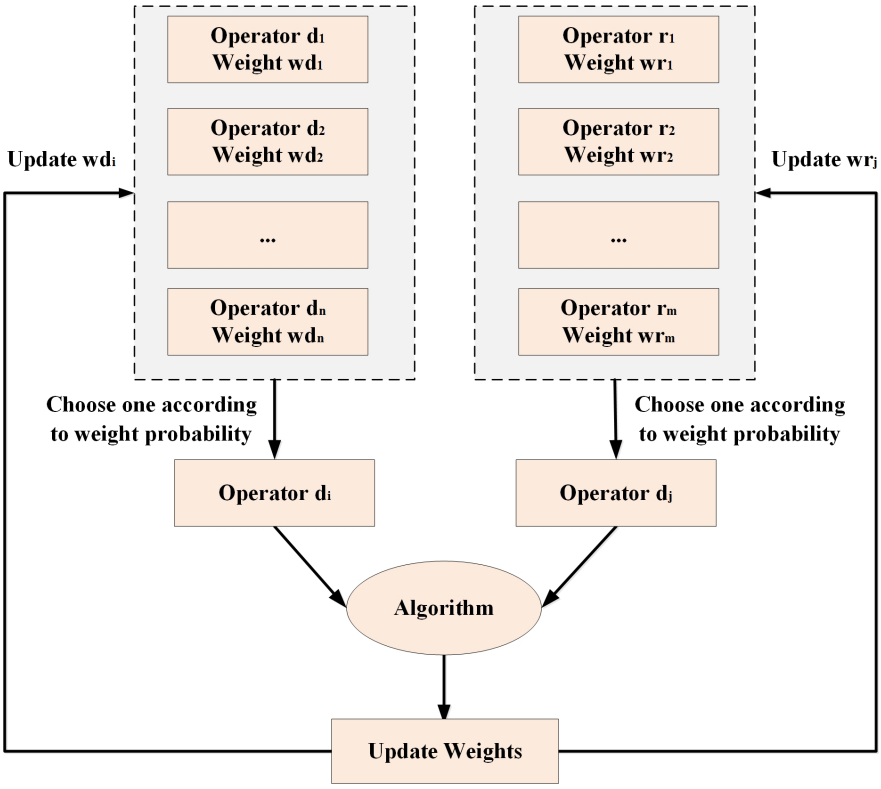


Figure 3: Classical Adaptive Methods (*AM*1)

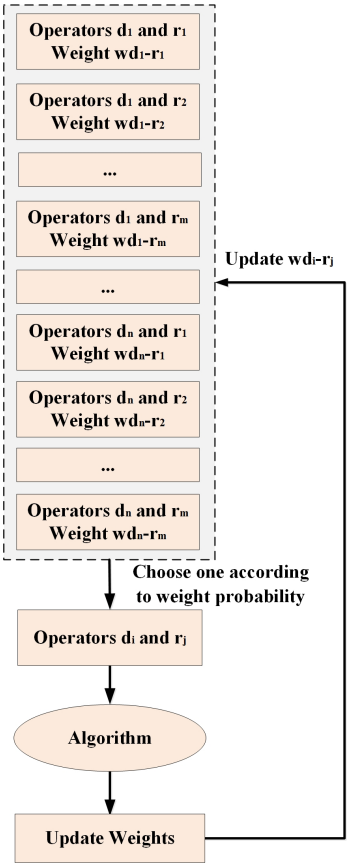


Figure 4: Destroy-repair Combination Weight-adjusted (AM2)

and m repair operators, there are n possibilities for choosing a destroy operator and $n \times m$ possibilities for the repair operator (since the selection of the repair operator is made based on the selection result of the destroy operator). The corresponding update will be made when the weight is updated.

The elements in the ρ^- and ρ^+ vectors corresponding to the selected destroy and repair operations are updated using the following equation: $\rho_a^- = \lambda\rho_a^- + (1-\lambda)\psi$, $\rho_{b|a}^+ = \lambda\rho_{b|a}^+ + (1-\lambda)\psi$, in which a and b are the indices of the destroy and repair operators applied in the algorithm's last iteration, respectively, and $\rho_{b|a}^+$ represents the indices of the components corresponding to the selected repair operator b with the destroy operator a. The flow chart of $AM3$ is in Figure 5.

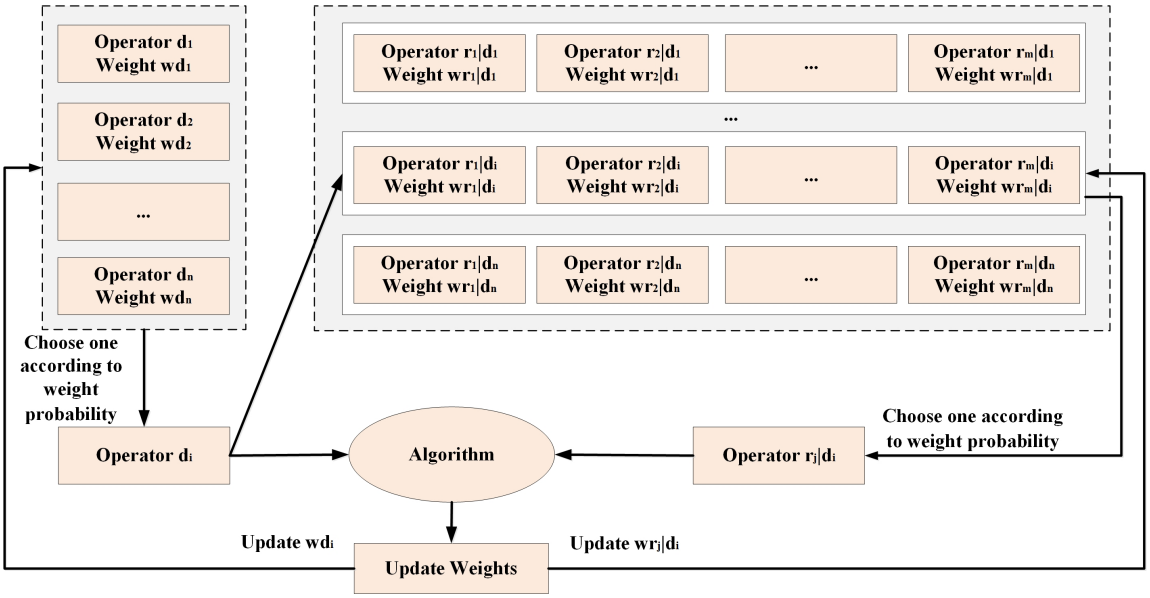


Figure 5: Destroy-based repair weight-adjusted (AM3)

4.3.4. Reward-Based Weight-Adjusted (AM4)

The existing weight adjustment calculation formula, $\psi = max(w_1, w_2, w_3, w_4)$, where w_1, w_2, w_3, and w_4 are fixed values, exhibits a lack of adaptability. This rigidity, to some extent, constrains the flexibility of weight adjustment. Regardless of how beneficial an operation is to the solution, it can only contribute a fixed value, such as adding w_1, which may not be entirely rational.

To address this, we have devised a reward-based method for weight adjustment. This method primarily adjusts the weight in accordance with the profit derived from the cost. Furthermore, the profit generated by this operation is linearly and positively correlated with the number of steps taken without finding a new global optimal solution. This implies that if there are multiple unsuccessful attempts before a successful operation, the accumulated scores from all these failed attempts are attributed to the successful operation. This approach ensures a more dynamic and responsive weight adjustment process.

The reward-based weight-adjusted method sets the w_1 in the weight adjustment formula in Subsection 4.3.1. However, we set $w_1 = max(w_{fix}, min(\Delta/10, w_{fix} + \lfloor \Delta * NumNunImp/MaxNonImp \rfloor + \lfloor \mu/\_\mu \rfloor))$ where Δ and w_{fix} is a fixed value, and $NumNunImp$ is the number of steps without finding a new global optimal solution, $MaxNonImp$ is the maximum number of iterations the solution quality does not improve (see Appendix A for details), μ refers to the decrease in cost brought about by the objective function, and $\_\mu$ depends on the value of the initial solution.

5. Experimental Evaluation

We perform the following five computational experiments. First, we analyze the effect of valid inequalities on the speed of model solving by CPLEX for small-scale instances. Second, the performance of $AM1$, $AM2$, $AM3$, and $AM4$ in the ALNS algorithm is evaluated. Third, we test the ALNS and CPLEX performance on small-scale problems. Fourth, we compare our HTRD model with a two-echelon vehicle routing model (2E-VRP) and a truck-based robot delivery model (TRD) and their fair versions to evaluate how our proposed hybrid model performs. Fifth, we perform sensitivity analysis experiments of the HTRD model to evaluate the impact of truck no-go area size and time windows.

The mathematical programming model is written with Python/Docplex and we use CPLEX v12.8 to solve it. The ALNS-based algorithm is written in Python v3.6.5. Both approaches are run on an Intel(R) Xeon(R) CPU E5-2696 v4 2.2 GHz processor with 128 GB RAM under MS Windows 10. Python is run using a single thread.

We create 20 small-scale, 20 medium-scale and 20 large-scale instances, using Manhattan distances between each node in the network. For the instances, ten of these cases of each size have truck no-go areas generated in a random way

("ssr", "msr" and "lsr" for small/medium/large-scale instances) and ten cases in an aggregated way ("ssa", "msa" and "lsa" for small/medium/large-scale instances). For small-scale instances, we create scenarios of 10 × 10 km^2 and 100 × 100 m^2 blocks, two possible robot hub locations, five possible parking nodes, five possible customer nodes, and one depot. For medium and large-scale instances, we create scenarios of 20 × 20 km^2 and 100 × 100 m^2 blocks, ten possible robot hub locations, 30 possible parking nodes, 50 and 200 possible customer nodes, and one depot. The customers are randomly placed at the corner points of the blocks, with no duplicates. The depot has a time window with a length of eight hours and a range of [0,8]. The customer time window has a length of two hours and an integer start time. Demand is randomly chosen from the range of [1 kg, 2 kg, 5 kg, 10 kg, 20 kg].

We adopt two methods to generate truck no-go areas. The first method assumes truck no-go areas (corresponding to truck no-go customers) are randomly distributed on the map. We set up the trucks to access the first two-thirds of the customer nodes in each instance. We rounded up if the calculated number of truck customers was a fraction. The second method assumes that the customers whose trucks are not allowed to visit are distributed in the city's core area. To this end, we assume block areas are truck no-go areas, and customers in these areas are truck no-go customers. This area is 5×5 km^2 (representing the inner ring of the city, called small truck no-go areas), 7×7 km^2 (the middle ring of the city, called medium truck no-go areas), 9×9 km^2 (the outer ring of the city, called large truck no-go areas). The remaining areas represent the outside of the outer ring of the city. Besides, we assume that the coordinates of the city center are (5, 5). Note it does not mean that truck no-go areas are entirely inaccessible for trucks. We assume that trucks can still travel on a few road sections in these areas, and just the stops and transfer stations are located on these routes. The equipment settings of robots, trucks, hubs, and parking nodes we assumed are listed in Table 2. The detailed equipment setting instructions are in Appendix C.

Table 2: Equipment parameters

Equipments	Parameter	Values
Robot	Speed	6km/h
	Battery capacity	1.5 kwh
	Capacity	50kg
	Travel cost rate	\$ 0.1/km
	Energy consumption rate	0.1 kwh/km
Truck	Speed	25km/h
	Capacity	500kg
	Travel cost rate	\$ 0.6/km
	Time consumption cost rate	\$ 3.5 /per hour
Hub	Hub-robot in a hub	5 robots

The small-scale instances are used in the valid-inequalities, ALNS-CPLEX-performance-analysis, and model-comparison experiments. The medium/large-scale instances are used in experiments related to the ALNS algorithm: adaptive methods, model-comparison, and model-sensitivity-analysis experiments. Note the ALNS algorithm will solve each instance ten times and then take the average solution. The set of ALNS parameters utilized in the numerical analysis, along with their descriptions and values, are based on Yu et al. (2024), as follows. We set the maximum number of ALNS iterations to 10000, 30000, and 10000 for small/medium/large-scale instances in our HTRD experiments, considering the trade-off between solution time and solution quality. The maximum number of iterations without solution improvement is set to 200, and this value is updated after each restart by adding the current iteration count to the existing value. The destruction rate β to 0.3, the decay parameter λ to 0.95, the start temperature per customer of simulated annealing T_{st} to 1000, the cooling rate of simulated annealing α to 0.92, and the four choice weights w_1, w_2, w_3, w_4 to 3, 2, 1, 0, respectively. For reward-based weight-adjusted, we set $\Delta = 100$, and $w_{fix} = 3$ in the computational study.

5.1. Valid Inequalities Analysis Experiments

We analyze the effect of valid inequalities on the speed of solving the HTRD model in small-scale instances in CPLEX (default setting, maximum solution time of 18000s).

Table 3 reports the results of valid inequalities experiments for the HTRD model. Column 1 shows the instances. Column 2 – Column 9 represent the solving time of the HTRD model in different valid inequalities settings. Column 2 represents the solving time of the basic HTRD model, Column 3 – Column 9 represent the solving time of the basic HTRD model with valid inequalities (39)-(40), (41)-(42), (39)-(42), (43)-(44), (43)-(45), (39)-(44), and (39)-(45), respectively. E1(%) represents the gap between the solving time of the HTRD model with different valid inequalities settings and the basic HTRD model settings (baseline).

Table 3 shows that most constraint combinations in the experiment are valid. The valid inequalities (39)-(45) have strong benefits in improving the HTRD model solution speed for ssr-types of instances, which can improve the solution speed by 71.3%. The valid inequalities (39)-(44) have strong benefits in improving the HTRD model solution speed for ssa-types of instances, which can improve the solution speed by 76.3%. Considering both instance classes and aiming for a single setting that is robust across different problem characteristics, we adopt the full valid inequalities (39)–(45) in

Table 3: Valid inequalities analysis experiments

	Basic	+c(39)-(40)	+c(41)-(42)	+c(39)-(42)	+c(43)-(44)	+c(43)-(45)	+c(39)-(44)	+c(39)-(45)
Instances				Time(s)				
ssr1	133.4	73.3	124.2	50.7	8.3	7.6	3.0	6.9
ssr2	18000.0	7797.0	18000.0	10313.1	4691.1	3909.4	1513.3	1629.6
ssr3	69.3	35.8	326.1	38.6	2.0	1.4	2.0	2.5
ssr4	18000.0	18000.0	18000.0	18000.0	18000.0	18000.0	15620.8	11892.8
ssr5	8421.2	12046.3	10494.7	5404.3	1519.6	2504.6	1914.4	998.8
ssr6	18000.0	18000.0	18000.0	18000.0	10205.8	11213.3	2461.4	6135.5
ssr7	18000.0	18000.0	18000.0	18000.0	7592.2	7576.7	4913.9	4638.5
ssr8	18000.0	18000.0	18000.0	18000.0	14387.1	14729.1	6010.3	5477.7
ssr9	16596.9	8321.8	18000.0	18000.0	8659.4	2557.1	226.8	1260.2
ssr10	18000.0	18000.0	18000.0	18000.0	11296.8	15203.9	11915.3	6226.1
Avg time	13322.1	11827.4	13694.5	12380.7	7636.2	7570.3	4458.1	3826.8
E1(%)	0.0	-11.2	2.8	-7.1	-42.7	-43.2	-66.5	-71.3
ssa1	99.9	110.6	266.5	71.3	15.4	10.6	3.4	11.3
ssa2	18000.0	13112.8	18000.0	12265.6	2973.1	4355.7	443.5	598.1
ssa3	13.8	37.9	180.7	22.8	2.1	1.3	1.7	2.1
ssa4	1960.4	895.9	2019.8	334.7	13.5	14.0	12.3	20.6
ssa5	18000.0	11956.3	18000.0	2710.8	3350.6	2345.8	455.7	2285.2
ssa6	18000.0	18000.0	18000.0	18000.0	13604.9	18000.0	8514.2	8485.5
ssa7	18000.0	18000.0	18000.0	18000.0	9258.3	9039.0	3573.3	7164.0
ssa8	18000.0	18000.0	18000.0	18000.0	17753.4	18004.5	15237.5	12239.1
ssa9	18000.0	12230.3	18000.0	10989.0	6123.2	719.1	1619.3	2466.9
ssa10	18000.0	4559.3	18000.0	18000.0	2892.7	2672.9	494.9	653.3
Avg time	12807.4	9690.3	12846.7	9839.4	5598.7	5516.3	3035.6	3392.6
E1(%)	0.0	-24.3	0.3	-23.2	-56.3	-56.9	-76.3	-73.5

all subsequent computational experiments. This choice strikes the best compromise between universality and efficiency, guaranteeing the lowest average run-time without sacrificing solution quality.

Although the valid inequalities can reduce the solving speed of the MIP model by more than 70%, it still takes more than 3000s to solve a small-scale instance (10 nodes) on average with a high-performance computer. Problem sizes of possibly solvable instances using MIP solvers remain small, as computational effort increases exponentially.

5.2. Configuration of ALNS

In this section, we conduct a comparative evaluation of our framework and the proposed adaptive mechanisms on three types of problems: the classic Capacitated Vehicle Routing Problem (CVRP), the Two-Echelon Vehicle Routing Problem (2E-VRP), and our newly introduced HTRD problem. The evaluation is carried out on both standard benchmark datasets and custom-designed instances, covering various problem scales to comprehensively assess performance.

5.2.1. Evaluation on benchmark instances

We conduct experiments using well-established CVRP and 2E-VRP benchmark instances to validate our approach.

For the CVRP, we select the Christofides et al. (1981) dataset for testing. We use our ALNS framework with basic operations such as Random Destroy, Worse Destroy, Random Repair, Greedy Repair, and Regret Repair. The procedure is repeated until the given number of iterations ($\max\{50 \times n, 2000\}$, where n is the number of customers) has been executed (Zhao et al. 2024).

The comparison of results for CVRP known benchmark instances using AM3-based ALNS are in Table 4. Table 4 shows that our algorithm can achieve or approach the optimal solution on small-scale instances (e.g., 50–100 nodes), and for instances with 150 nodes, the best gap remains within 2%.

Table 4: Comparison of Results for CMT Instances using AM3-based ALNS

Instances	n	Q	Opt. Value	Best Found	Avg. Found	Best Gap (%)	Avg Gap (%)
CMT1	50	160	524.61	524.61	524.64	0.00	0.01
CMT2	75	75	835.26	835.77	848.05	0.06	1.53
CMT3	100	200	826.14	832.59	839.40	0.78	1.61
CMT4	150	200	1028.42	1048.06	1058.55	1.91	2.93
CMT12	100	200	819.56	819.56	839.90	0.00	2.48
Avg	/	/	806.80	812.12	822.11	0.55	1.71

The performance of the CVRP-ALNS algorithm based on the four adaptive mechanisms AM1 to AM4 on CMT instances is shown in Table 5. This comparison strictly follows the principle of controlling variables: only the adaptive mechanism is changed in the algorithm. As shown in Table 5, the AM3-based ALNS method achieves the best performance in both best gap and average gap, demonstrating the superiority of the AM3 mechanism in solving the CVRP.

For 2E-VRP, we select the Breunig et al. (2016) dataset for testing. We use our ALNS framework and design some operators (five destroy operators: Worst-Customer Removal, Random-Customer Removal, Route Removal, Satellite Removal, Related-Customer Removal; and four repair operators: Basic Greedy Customer Insertion, Regret-3 Customer Insertion, Random Customer Insertion, Build-New-L2-Routes) that fit this problem, and used 2-opt local search after

Table 5: Performance Comparison of Algorithms (AM Series) on CMT Instances

AM	Opt. Value	Best Found	Avg. Found	Best Gap (%)	Avg. Gap (%)
AM1	806.80	812.31	823.68	0.56	1.90
AM2	806.80	814.86	823.12	0.86	1.85
AM3	806.80	812.12	822.11	0.55	1.71
AM4	806.80	813.98	823.53	0.76	1.87

each iteration to improve solution. The number of iterations is set at 20,000 for instances with fewer than 30 customers, 400,000 iterations for instances with fewer than 100 customers, the same as the setting in Gutierrez et al. (2024).

The comparison of results for 2E-VRP known benchmark instances (2E-VRP-Set2) using AM3-based ALNS are in Table 6. Table 6 shows that under the AM3-based ALNS framework with basic operators previously introduced, the algorithm can achieve optimal in all Set2-based instances.

Table 6: Comparison of Results for 2E-VRP-Set2 Instances using AM3-based ALNS

Instance Name	Opt. Value	Best Found	Avg. Found	Best Gap (%)	Avg. Gap (%)
Set2_E-n22-k4-s8-14	384.96	384.96	384.96	0.00	0.00
Set2_E-n22-k4-s9-19	470.60	470.60	470.60	0.00	0.00
Set2_E-n22-k4-s10-14	371.50	371.50	371.50	0.00	0.00
Set2_E-n22-k4-s11-12	427.22	427.22	427.44	0.00	0.05
Set2_E-n22-k4-s12-16	392.78	392.78	394.37	0.00	0.40
Set2_E-n33-k4-s1-9	730.16	730.16	733.94	0.00	0.52
Set2_E-n33-k4-s2-13	714.63	714.63	719.30	0.00	0.65
Set2_E-n33-k4-s3-17	707.48	707.48	718.61	0.00	1.57
Set2_E-n33-k4-s4-5	778.74	778.74	778.74	0.00	0.00
Set2_E-n33-k4-s7-25	756.85	756.85	762.50	0.00	0.75
Set2_E-n33-k4-s14-22	779.05	779.05	779.05	0.00	0.00
Avg	592.18	592.18	594.64	0.00	0.36

The performance of the 2E-VRP-ALNS algorithm based on the four adaptive mechanisms AM1 to AM4 on 2E-VRP-Set2 instances is shown in Table 7. This comparison strictly follows the principle of controlling variables: only the adaptive mechanism is changed in the algorithm. As shown in Table 7, the AM3-based ALNS method achieves the best performance in both best gap and average gap, demonstrating the superiority of the AM3 mechanism in solving the 2E-VRP.

Table 7: Performance Comparison of Optimization Algorithms

AM	Opt. Value	Best Found	Avg. Found	Best Gap (%)	Avg. Gap (%)
AM1	592.18	592.18	595.52	0.00	0.51
AM2	592.18	592.18	596.05	0.00	0.57
AM3	592.18	592.18	594.64	0.00	0.36
AM4	592.18	592.18	594.67	0.00	0.36

Overall, experimental results demonstrate the effectiveness of our framework and adaptive approach (AM3).

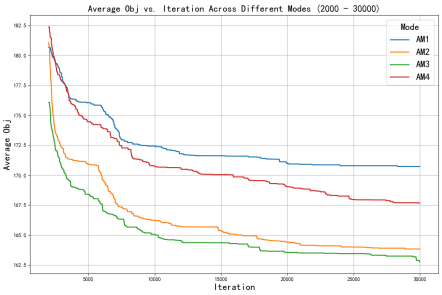
5.2.2. Evaluation based on self-built instances

We evaluate the performance of four adaptive methods (AM1, AM2, AM3, and AM4) in the ALNS algorithm using ten msr/lsr-type instances. In our experiments, we first tackle the more challenging and newly defined HTRD problem. After analyzing the HTRD problem, we perform experiments on CVRP problems. For CVRP, we use our ALNS framework with common operations such as Random Destroy, Worse Destroy, Random Repair, Greedy Repair, and Regret Repair.

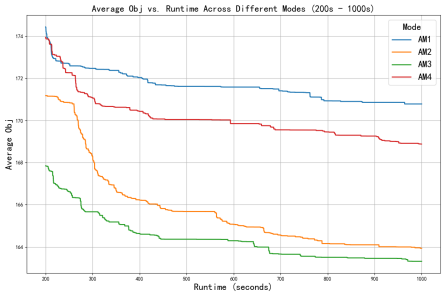
Figures 6 and 7 provide a comprehensive analysis of the ALNS algorithm under different adaptive methods for both HTRD and CVRP. The both experiments all include scenarios with 50 customers and 200 customers. In the graphs, the horizontal axis displays the iteration count or runtime, while the vertical axis shows the average cost.

From Figure 6a and Figure 6b, we observe that for medium-sized instances with 50 customers, the AM3-based ALNS algorithm outperforms the other three adaptive methods in both the number of iterations and running time under our experiment setting in the HTRD problem. We also observe that AM1 performs the worst, and AM2 shows relatively good performance. For larger instances with 200 customers, AM3 maintains its lead in the number of iterations. In terms of running time, AM3 has no advantage before 4000s, but its performance surpasses AM1 and AM4 after 4000s, and surpasses AM2 after 8000s. During the experiment, we also observed that AM2 also has good performance in our experimental setting.

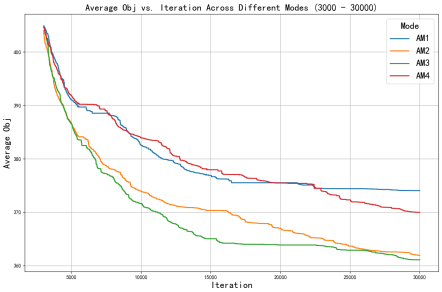
For medium-scale HTRD experiments, we made our choice by balancing solution time and performance. Figure 6a shows that when using the AM3 method, the objective function decreased by about 0.9% from 10,000 to 20,000 iterations, and by only about 0.3% from 20,000 to 30,000 iterations. Although increasing the iterations adds some computational load, this burden remains manageable for these instances. Therefore, to achieve the maximum performance



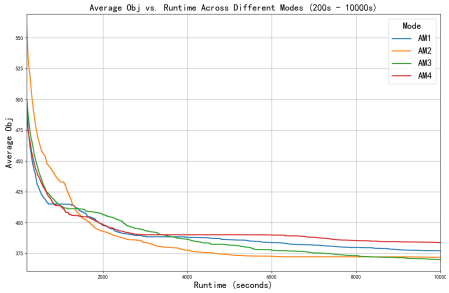
(a) HTRD experiments with 50 customers: Analysis of different modes over iteration



(b) HTRD experiments with 50 customers: Analysis of different modes over time

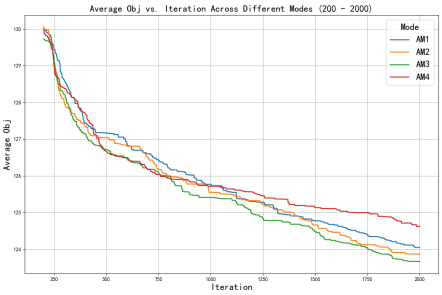


(c) HTRD experiments with 200 customers: Analysis of different modes over iteration

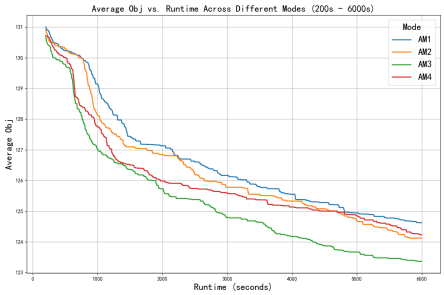


(d) HTRD experiments with 200 customers: Analysis of different modes over time

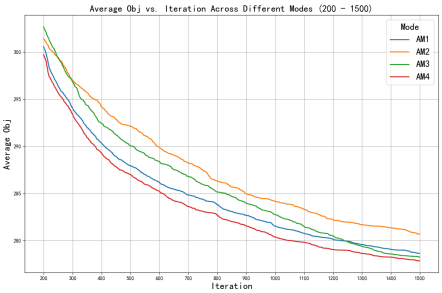
Figure 6: HTRD experiments: Analysis of different modes



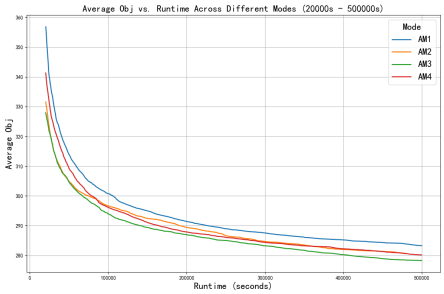
(a) CVRP experiments with 50 customers: Analysis of different modes over iteration



(b) CVRP experiments with 50 customers: Analysis of different modes over time



(c) CVRP experiments with 200 customers: Analysis of different modes over iteration



(d) CVRP experiments with 200 customers: Analysis of different modes over time

Figure 7: CVRP experiments: Analysis of different modes

gain, we opted for 30,000 iterations. It is important to note that while further increases in iterations would continue to reduce the objective function value, the additional improvements would likely be minimal—each extra 10,000 iterations is expected to yield benefits well below 0.3%, while the solution time would increase significantly. For large-sized instances, we choose 10,000 iterations mainly due to strict computational time limitations.

From Figures 7a and 7b, we can see that for the classic CVRP problem with 50 customers under our experimental conditions, the ALNS algorithm using the AM3 method outperforms the other three methods in both the iteration count and overall runtime, with a particularly notable speed advantage. For the larger 200-customer CVRP problem, Figures 7c and 7d reveal a similar trend under our experimental conditions. The AM3-based ALNS algorithm leads in solving time compared to the other methods. However, in terms of iterations, AM3 initially lags behind AM4 and AM1. Interestingly, once the number of iterations surpasses 1300, AM3 overtakes AM1 and quickly catches up to AM4, demonstrating its ability to adapt and improve over time.

From Figure 7d we can see an interesting phenomenon: even after running for 500,000 seconds (approximately 6 days), the ALNS algorithm continues searching for better solutions with all adaptive methods. This underscores the complexity of the 200-customer CVRP problem. On the positive side, this highlights the strength of our ALNS framework, which remains capable of exploring and refining solutions over extended periods, showcasing its advanced optimization capabilities.

Both HTRD and CVRP experimental results indicate that the AM3-based ALNS algorithm not only performs well on the basic CVRP but is also highly adaptable to the more complex HTRD problem in our experimental setting. Based on these findings, we have chosen to adopt AM3 for our future experiments HTRD experiments.

5.3. ALNS and CPLEX Performance on Small-Scale Problems

In this section, we first focus on evaluating the performance of the CPLEX algorithm in solving small-scale problems. In the experiments described in Section 5.1, we observe that CPLEX could efficiently solve problems with 5 customer nodes. Based on this, we conduct experiments by gradually increasing the problem size in increments (e.g., 5, 7 customer nodes). The time limit for CPLEX is set to 18000s to assess its ability to solve problems of varying scales using exact methods.

Although vehicles could in principle revisit parking nodes multiple times, allowing frequent returns to virtual parking nodes drastically increases problem complexity. In fact, if each vehicle is permitted to visit the same parking node twice, even five-customer instances become intractable under our MILP formulation. Hence, we restrict each vehicle to at most one visit per parking node in all CPLEX experiments.

Next, we apply the designed ALNS algorithm to solve small-scale instances and compare the results with those obtained by CPLEX. This comparison aimed to evaluate the ALNS algorithm's performance on small-scale problems. Strong performance on these smaller instances would increase confidence in its potential effectiveness for larger-scale problems. To ensure the consistency of the comparison between the ALNS heuristic method and the CPLEX benchmark, we here enforce the one-visit-per-parking node-per-vehicle limit implicitly in ALNS by imposing a substantial penalty in the objective.

Through experimental studies, we find that CPLEX is able to solve problems with up to 5 customer nodes with one virtual parking node. However, when the number of customer nodes increased to 7, many instances could not be solved within the 18000s time limit. This clearly illustrates the difficulty of solving the HTRD problem.

Table 8 compares the performance of two solvers, ALNS and CPLEX, on small-scale problems. The first row lists the names of the solvers, including CPLEX and our designed ALNS algorithm. The second row displays the title name, included instances, the objective value for each instance obtained by the CPLEX solver, the best objective value for each instance obtained by the ALNS solver over 10 runs, their solving time, and the gap (E2%) between the objective values of the two solvers. Rows 3 through 12 show the results, and row 13 shows the average value. The same also applies to the results of the instances with 7 customers in the table. Note that "5-ssr1" represents an "ssr" type instance 1 with 5 customer nodes. The others follow the same naming convention.

As shown in Table 8, for the instances with 5 and 7 customer nodes, the proposed ALNS algorithm obtains results identical to the optimal solutions found by CPLEX in all but one case (5-ssa2), which highlights its remarkable effectiveness.

5.4. Evaluating HTRD Cost Versus Other Network Architectures

We compare the HTRD model with two classical types of models: the TRD model and the 2E-VRP model, to see the performance of the newly proposed model. In the specific experiments, we use CPLEX to solve small-scale instances and ALNS to solve medium/large-scale instances and conduct model comparison experiments. Note that the TRD model and 2E-VRP model are special cases of the HTRD model. The TRD model is derived from the HTRD model by eliminating hubs, and the 2E-VRP model is obtained from the HTRD model by prohibiting trucks from carrying robots.

Our preliminary experiments show that the HTRD model is feasible for all instances, while the 2E-VRP and TRD models are not feasible for some cases. This clearly shows the advantages of the HTRD model compared to the classic 2E-VRP and TRD models.

Table 8: ALNS and CPLEX Results on Small-Scale Problems

Solver	CPLEX		ALNS			Solver	CPLEX		ALNS		
Instances	Obj	Time(s)	Obj(Best)	Time(s)	E2(%)	Instances	Obj	Time(s)	Obj(Best)	Time(s)	E2(%)
5-ssr1	17.4	6.9	17.4	15.3	0.0	5-ssa1	17.4	11.3	17.4	23.8	0.0
5-ssr2	36.5	1629.6	36.5	11.3	0.0	5-ssa2	38.3	598.1	39.4	41.7	3.0
5-ssr3	17.8	2.5	17.8	17.3	0.0	5-ssa3	14.4	2.1	14.4	38.1	0.0
5-ssr4	43.0	11892.8	43.0	15.1	0.0	5-ssa4	19.6	20.6	19.6	46.5	0.0
5-ssr5	30.4	998.8	30.4	20.2	0.0	5-ssa5	30.4	2285.2	30.4	27.0	0.0
5-ssr6	39.4	6135.5	39.4	18.4	0.0	5-ssa6	37.1	8485.5	37.1	26.5	0.0
5-ssr7	42.1	4638.5	42.1	16.1	0.0	5-ssa7	38.7	7164.0	38.7	32.3	0.0
5-ssr8	43.2	5477.7	43.2	14.6	0.0	5-ssa8	39.5	12239.1	39.5	30.5	0.0
5-ssr9	28.7	1260.2	28.7	15.7	0.0	5-ssa9	28.1	2466.9	28.1	27.6	0.0
5-ssr10	35.0	6226.1	35.0	15.5	0.0	5-ssa10	29.5	653.3	29.5	13.8	0.3
Avg	33.4	3826.9	33.4	15.9	0.0	Avg	29.3	3392.6	29.3	30.8	0.0
7-ssr1	17.9	31.5	17.9	22.1	0.0	7-ssa1	17.9	122.0	18.0	14.3	0.0
7-ssr2	43.3	9431.8	43.3	20.7	0.0	7-ssa2	42.0	16154.2	42.0	22.7	0.0
7-ssr3	44.8	1032.2	44.8	24.8	0.0	7-ssa3	41.0	6722.9	41.0	23.8	0.0
7-ssr4	48.2	18000.0	48.2	17.9	0.0	7-ssa4	42.9	18000.0	42.9	16.3	0.0
7-ssr5	33.6	178.5	33.6	20.7	0.0	7-ssa5	32.8	526.5	32.8	16.3	0.0
7-ssr6	55.0	18000.0	55.0	24.1	0.0	7-ssa6	43.0	18000.0	43.0	7.4	0.0
7-ssr7	41.4	16006.6	41.4	19.4	0.0	7-ssa7	38.7	18000.0	38.7	13.8	0.0
7-ssr8	47.3	16659.5	47.3	17.9	0.0	7-ssa8	46.0	18000.0	46.0	15.2	0.0
7-ssr9	34.9	8247.2	34.9	19.3	0.0	7-ssa9	35.4	18000.0	35.4	14.3	0.0
7-ssr10	37.9	18000.0	37.9	18.0	0.0	7-ssa10	35.7	18000.0	35.7	9.1	0.0
Avg	40.4	10558.7	40.4	20.5	0.0	Avg	37.6	13152.6	37.6	15.3	0.0

Note: When solving the model with CPLEX, the number of virtual stops in this table is set to 1.

In practice, an instance is infeasible because there are not enough hubs or parking nodes to cover the entire delivery area. For a fairer comparison concerning accessibility, we change the parking nodes in the HTRD model to hubs and add them to the 2E-VRP model, called 2E-VRP-fair model. Besides, we also change the hubs in the HTRD model to parking nodes and add them to the TRD model, called TRD-fair model.

Table 9, 10 and 11 report the results of the 2E-VRP, TRD, 2E-VRP-fair, TRD-fair and HTRD model comparison experiments using CPLEX/ALNS to solve small/medium/large-scale instances. Row 1 represents the different models. Cost/Avg cost shows the costs for CPLEX/ALNS solver. Row 3-12 represent the costs of the experiments for 2E-VRP, TRD, 2E-VRP-fair, TRD-fair, and HTRD model. E3(%) represents the gap between the cost of the HTRD model (baseline) and the other models. Note in Table 9, italics represents the current optimal solution (the optimal solution has not been found within 18000s). In Table 9 - 11, "/" represents no feasible solution.

Table 9: Model Comparison Experiments Using CPLEX Solver With Small Scale Instances

CPLEX	2E-VRP	TRD	2E-VRP-fair	TRD-fair	HTRD	CPLEX	2E-VRP	TRD	2E-VRP-fair	TRD-fair	HTRD
Instances	Cost	Cost	Cost	Cost	Cost	Instances	Cost	Cost	Cost	Cost	Cost
ssr1	8.5	36.8	6.0	*36.0*	8.5	ssa1	9.6	37.3	7.0	*36.5*	9.6
ssr2	5.6	37.2	5.6	33.8	5.6	ssa2	5.7	53.0	5.7	39.7	5.7
ssr3	10.4	40.0	9.5	*40.2*	10.4	ssa3	16.7	/	9.3	37.0	10.2
ssr4	10.4	47.0	9.5	*37.8*	10.3	ssa4	31.1	39.2	7.5	*37.6*	9.7
ssr5	10.9	39.9	9.8	*39.9*	10.9	ssa5	19.8	36.7	8.9	*34.9*	9.8
ssr6	11.1	44.1	10.0	40.2	11.1	ssa6	10.9	40.5	10.0	*38.1*	10.9
ssr7	6.9	45.5	6.9	*42.7*	6.9	ssa7	16.5	44.4	6.4	*40.9*	6.4
ssr8	11.0	43.2	11.0	*43.2*	11.0	ssa8	23.9	39.4	10.4	*39.3*	10.4
ssr9	8.2	30.7	6.2	27.3	8.2	ssa9	8.0	28.2	6.1	*27.0*	8.0
ssr10	10.4	35.0	9.4	*34.2*	10.4	ssa10	6.6	29.5	6.6	*29.5*	6.6
Avg cost	9.3	39.9	8.4	37.5	9.3	Avg cost	14.9	/	7.8	36.1	8.7
E3 (%)	0.2	329.4	-10.1	302.6	0.0	E3 (%)	71.1	/	-10.8	312.9	0.0

Table 10: Model Comparison Experiments Using ALNS With Medium Scale Instances

ALNS	2E-VRP	TRD	2E-VRP-fair	TRD-fair	HTRD	ALNS	2E-VRP	TRD	2E-VRP-fair	TRD-fair	HTRD
Instances	Avg cost	Avg cost	Avg cost	Avg cost	Avg cost	Instances	Avg cost	Avg cost	Avg cost	Avg cost	Avg cost
msr1	/	317.2	144.1	307.8	167.7	msa1	/	258.2	150.6	244.2	170.2
msr2	/	336.8	148.1	338.3	204.6	msa2	/	253.1	178.8	259.6	193.8
msr3	147.7	278.4	146.8	275.8	170.8	msa3	186.3	238.9	184.8	236.9	164.9
msr4	136.8	286.9	135.9	280.6	150.8	msa4	145.4	254.5	147.0	242.7	152.8
msr5	/	335.5	130.5	334.3	163.4	msa5	/	294.1	140.3	286.3	162.1
msr6	126.1	291.1	141.5	281.0	139.9	msa6	137.4	279.5	131.7	256.1	142.7
msr7	/	272.2	128.2	266.5	153.5	msa7	171.1	236.1	167.7	218.4	167.4
msr8	/	/	148.1	285.0	172.8	msa8	161.1	245.0	161.5	234.9	159.4
msr9	/	293.5	141.1	297.7	177.5	msa9	/	276.4	164.1	270.3	181.3
msr10	145.4	303.7	157.1	309.5	160.8	msa10	164.7	251.4	158.5	257.5	182.3
Avg cost	/	/	142.1	297.6	166.2	Avg cost	/	258.7	158.5	250.7	167.7
E3 (%)	/	/	-14.5	79.1	0.0	E3 (%)	/	54.3	-5.5	49.5	0.0

Tables 9, 10, and 11 show that the HTRD model significantly outperforms both the 2E-VRP and TRD models. This advantage is evident in small, medium, and large-scale instances. In small-scale instances, the 2E-VRP model is able to obtain the optimal solution for both "ssr" and "ssa" instances using CPLEX. However, when solving these instances with the 2E-VRP model, the average costs are 0.2% and 71.1% higher compared to the HTRD model, respectively. On the other hand, the TRD model encounters infeasible solutions for some samples, particularly in "ssa" instances. For the solvable "ssr" instances, using the TRD model results in an average cost 329.4% higher than the HTRD model. In the medium-to-large-scale experiments, the 2E-VRP model fails to solve all instances in the "msr", "msa", "lsr", and "lsa" categories, and the number of infeasible instances is relatively high. The TRD model also fails to solve all instances in the "msr", "lsr", and "lsa" categories, and only fully solves the "msa" instances. However, even for the "msa" instances, the average cost of the TRD model still falls short of the HTRD model by 54.3%.

Table 9, 10, and 11 also show that in our "fair" setting, the HTRP model performs much better than the TRD-fair model but it is inferior to the 2E-VRP-fair model. The HTRD model outperforms the TRD-fair model: when using the TRD-fair model on ssr/msr/lsr and ssa/msa/lsa instances, the average costs increase by 302.8%/79.1%/127.2%

Table 11: Model Comparison Experiments Using ALNS With Large Scale Instances

ALNS	2E-VRP	TRD	2E-VRP-fair	TRD-fair	HTRD	ALNS	2E-VRP	TRD	2E-VRP-fair	TRD-fair	HTRD
Instances	Avg cost	Avg cost	Avg cost	Avg cost	Avg cost	Instances	Avg cost	Avg cost	Avg cost	Avg cost	Avg cost
lsr1	/	814.1	263.4	811.9	342.6	lsa1	/	/	275.9	711.8	359.2
lsr2	/	836.1	293.8	836.5	370.0	lsa2	/	/	315.8	681.6	375.8
lsr3	/	/	288.8	879.6	380.9	lsa3	/	685.6	306.5	687.9	349.6
lsr4	/	784.2	260.2	781.9	335.6	lsa4	/	639.1	296.3	635.5	323.4
lsr5	/	858.7	284.8	857.3	347.9	lsa5	/	750.8	302.2	744.9	344.7
lsr6	/	822.1	266.5	811.1	349.3	lsa6	325.6	/	282.5	644.2	319.2
lsr7	/	857.8	268.1	854.6	357.9	lsa7	/	699.4	265.0	701.5	332.9
lsr8	/	/	287.9	813.4	399.5	lsa8	/	659.1	292.4	652.1	369.1
lsr9	/	805.5	274.4	803.8	378.4	lsa9	/	739.9	283.7	730.9	370.9
lsr10	/	836.0	280.4	818.5	376.7	lsa10	337.9	724.8	292.8	714.2	342.6
Avg cost	/	/	276.8	826.9	363.9	Avg cost	/	/	291.3	690.5	348.7
E3 (%)	/	/	-23.9	127.2	0.0	E3 (%)	/	/	-16.5	98.0	0.0

and 312.9%/49.5%/98.0% compared to the HTRD model. Besides, the 2E-VRP-fair model outperforms the HTRD model: when using the 2E-VRP-fair model on ssr/msr/lsr and ssa/msa/lsa instances, the average costs decrease by 10.1%/14.5%/23.9% and 10.8%/5.5%/16.5% compared to the HTRD model.

The advantage of the 2E-VRP-fair model is that it has more hubs than the HTRD model. A single hub can store multiple robots and serve multiple customers in the same time period, which is not possible in the parking node, which is likely to be one of the reasons why the 2E-VRP-fair model is better than the HTRD model. However, in the real world, it is unlikely to have such a high density of hubs because it requires a huge investment cost.

To enhance the comparative aspect of our model, we will next consider the usage costs of the hub and parking node. Specifically, we will incorporate the different usage costs of the hub and parking node into our existing model. Subsequently, we will compare and analyze the performance of the model further. This will aid us in better understanding and optimizing the performance of our model.

Our study begins with a comparative experiment focusing on the varying costs associated with visiting a hub. We establish the cost of a visiting robot as 1. Subsequently, we experiment with setting the cost of visiting a hub at different values – 5, 10, 15, and 20 – to observe the impact on the model comparison. Next, we conduct a similar comparison experiment, this time focusing on the different costs associated with visiting a parking node. Here, we set the cost of visiting a hub as 10. We then experiment with setting the cost of visiting a parking node at different values – 0.5, 1.0, 1.5, and 2.0 – to observe the impact on the model comparison. It is important to note that we will utilize all medium/large-scale instances of the "msr"/"lsr" type to conduct these experiments. The results of these model comparison experiments, which use ALNS with different hub/parking node costs in medium/large-scale instances, are reported in Table 12 and 13.

Table 12 and 13 report the results of the model comparison experiments using ALNS with different hub/parking node costs. Row 1 represents the different models. Column 1 and column 5 represent the cost of visiting a hub and a parking node, respectively. Avg cost shows the average costs for ALNS solver. E4(%) represents the gap between the cost of the HTRD model (baseline) and the 2E-VRP-fair models with different hub cost settings. E5(%) represents the gap between the cost of the HTRD model (baseline) and the TRD-fair models with different parking node cost settings.

Table 12: Model Comparison Experiments Using ALNS With Different Hub/Parking Node Cost On Medium Scale Instances

ALNS	2E-VRP-fair	HTRD		ALNS	TRD-fair	HTRD	
Hub cost	Avg cost	Avg cost	E4(%)	Parking cost	Avg cost	Avg cost	E5(%)
5.0	176.2	191.9	-8.2	0.5	301.6	220.1	37.0
10.0	209.1	221.0	-5.4	1.0	308.3	221.1	39.4
15.0	238.1	243.8	-2.3	1.5	311.9	221.5	40.8
20.0	264.5	270.8	-2.3	2.0	315.7	221.9	42.3
25.0	294.6	288.0	2.3	2.5	319.8	222.1	44.0
30.0	321.6	307.6	4.5	3.0	326.3	222.9	46.4

Table 13: Model Comparison Experiments Using ALNS With Different Hub/Parking Node Cost On Large Scale Instances

ALNS	2E-VRP-fair	HTRD		ALNS	TRD-fair	HTRD	
Hub cost	Avg cost	Avg cost	E4(%)	Parking cost	Avg cost	Avg cost	E5(%)
5	350.3	414.3	-15.4	0.5	835.8	460.4	81.5
10	417.1	460.4	-9.4	1	839	460.7	82.1
15	474.1	506.4	-6.4	1.5	865.5	462.4	87.2
20	528	544.4	-3	2	877.4	463.8	89.2
25	581.3	585.7	-0.8	2.5	887.3	464.3	91.1
30	633.4	633.3	0	3	896.4	464.9	92.8

For the cost of visiting hubs, Table 12 and 13 show that gradually increasing the cost of visiting a hub in the 2E-VRP-fair model and the cost of visiting a hub in the HTRD model will gradually increase the objective function value. However, as the cost of visiting a hub increases from 5 to 30, the gap between the objective function values of the 2E-VRP-fair model and the HTRD model significantly decreases, with the gap reducing from -8.2% to 4.5% in medium-scale instances and from -15.4% to 0% in large-scale instances.

For the cost of visiting parking nodes, Table 12 and 13 show that there is a progressive increase in the value of the objective function as the cost of visiting a parking node in the TRD-fair model is incrementally increased. In contrast,

the HTRD model exhibits only a marginal impact on the objective function when the cost of visiting a parking node is gradually increased. This minimal impact on the objective function may be attributed to the greater number of options available in the HTRD model, providing more flexibility. As the cost of visiting parking nodes escalates from 0.5 to 3, the disparity between the objective function values of the TRD-fair model and the HTRD model widens, with the gap increasing from 37.0% to 46.4% in medium-scale instances and from 81.5% to 92.8% in large-scale instances.

In summary, the model comparisons show the HTRD model has the advantage (with TRD and 2E-VRP model) that it can solve more instances. For logistic service providers, the TRD-fair model may encounter challenges due to its inability to decrease cost. On the other hand, the 2E-VRP-fair model, while generally more efficient, may lose its advantages as the cost of accessing the hub increases. Indeed, if these costs reach high levels, a factor that was once a strength could potentially become a liability. Furthermore, the 2E-VRP-fair model may require dense hub deployments, which may not always be feasible in real-world scenarios. The HTRD model is relatively more flexible and hence provides feasible solutions for logistic service providers. On the one hand, it can leave out a few hubs, creating more hubs in high-density customer zones. On the other hand, the HTRD model uses the mothership approach to serve the truck no-go customers in some low-density zones.

5.5. Model Sensitivity Experiments

We now perform a sensitivity analysis on the impact of truck no-go area size and time windows to see how the size and length of the truck no-go area access time window affect the model output. For this analysis, we use CPLEX to solve small-scale instances exactly, allowing us to observe precise changes. For large-scale instances, we rely on the ALNS heuristic to capture overall trends and general performance patterns.

5.5.1. The Impact of Truck No-Go Area Size

The truck no-go area's size may affect the delivery system's overall efficiency. We hence study the impact of the truck no-go area size on the distribution system's output. For ssr/lsr-type instances, we simulate increasing truck no-go area sizes by gradually raising the number of customers located within these restricted zones, ranging from the baseline to the maximum. These instances are solved using CPLEX (for small-scale) and ALNS (for large-scale) to assess their impact. In these cases, the number of customers restricted from truck access serves as a proxy for the size of the truck no-go area. For ssa/lsa-type instances, we simulate spatial expansion of the truck no-go areas by incrementally enlarging the restricted regions—from small to medium and then to large zones within the city. These variations are evaluated using the HTRD model, with solutions obtained through CPLEX or ALNS depending on instance scale.

Table 14 and 15 report the sensitivity analysis results on different sizes of truck no-go areas. Row 1 shows the different sizes of truck no-go areas. Rows 3-12 are detailed optimal/average results of the HTRD model. Row 13 shows the average cost of the HTRD model in different sizes of truck no-go areas. Row 14 (E6(%)) represents the gap between the average cost of the HTRD model with the different sizes of truck no-go areas and the baseline.

Table 14: Impact of Truck No-go Area Size (ssr/ssa instances using CPLEX)

TruckNoGoCust	4(max)	3	2	1	0(baseline)	TruckNoGoCust	Small	Medium	Large(baseline)
Instances	Opt	Opt	Opt	Opt	Opt	Instances	Opt	Opt	Opt
ssr1	17.37	17.37	17.37	17.37	17.37	ssa1	17.41	17.41	17.41
ssr2	43.57	43.57	43.50	36.51	36.51	ssa2	36.54	38.28	38.28
ssr3	17.79	17.79	17.79	17.79	17.79	ssa3	14.42	14.42	14.42
ssr4	42.95	42.95	42.95	42.95	40.89	ssa4	19.57	19.57	19.57
ssr5	30.89	30.45	30.45	30.45	30.45	ssa5	30.39	30.39	30.39
ssr6	42.14	42.14	42.14	39.40	39.40	ssa6	37.13	37.13	37.13
ssr7	42.12	42.12	42.12	42.12	40.63	ssa7	38.66	38.74	38.74
ssr8	52.08	52.08	43.21	43.21	40.53	ssa8	38.51	39.46	39.46
ssr9	34.49	33.37	28.67	28.67	28.67	ssa9	28.13	28.13	28.13
ssr10	38.05	36.15	36.15	35.03	33.08	ssa10	29.52	29.52	29.52
Avg cost	36.15	35.80	34.44	33.35	32.53	Avg cost	29.03	29.30	29.30
E6(%)	11.11	10.04	5.85	2.51	0.00	E6(%)	-0.95	0.00	0.00

Note: When solving the model with CPLEX, the number of virtual stops in this table is set to 1.

Table 14 shows the impact of the truck no-go customer parameter on small-scale instances: In the ssr-type instances, better results are obtained when the number of truck no-go customer approaches zero (i.e., the truck can access all customer nodes). The average performance improves from 36.15 to 32.53, and E6(%) drops from 11.11% to 0%. In the ssa-type instances, the "small" configurations (with averages of 29.03) outperform the "medium" (29.30) and "large" baseline (29.30), with E6(%) values of -0.95%. These results indicate that the number of truck no-go customer nodes has a clear impact on the optimal solution.

Table 15 shows the truck no-go customer experiments results on small-scale instances. For lsr-type instances, the table shows that as the number of customers within the truck no-go area increases, the cost of the distribution system is significantly increased. And generally, as the number of truck no-go customers increases, the cost growth trend also slows down. Doubling the number of truck no-go customers (which equates to an additional 67 customers in this instance) leads to a 17.7% increase in cost. Moreover, when the number of truck no-go customers is increased to 200, the cost escalates by 30.0%. For lsa-type instances, as the size of truck no-go areas decreases, the cost of the distribution

Table 15: Impact of Truck No-go Area Size (lsr/lsa instances using ALNS)

TruckNoGoCust	200(max)	167	134	100	67 (baseline)	TruckNoGoCust	Small	Medium	Large (baseline)
Instances	AC	AC	AC	AC	AC	Instances	AC	AC	AC
lsr1	548.6	524.1	449.4	421.7	342.6	lsa1	342.3	354.1	359.2
lsr2	538.5	506.1	431.0	414.6	370.0	lsa2	338.8	355.2	375.8
lsr3	456.4	485.2	459.0	428.7	380.9	lsa3	341.6	345.4	349.6
lsr4	395.8	403.1	378.3	330.2	335.6	lsa4	332.1	334.6	323.4
lsr5	435.2	409.4	410.3	366.0	347.9	lsa5	326.1	333.4	344.7
lsr6	440.1	451.4	408.1	377.7	349.3	lsa6	287.2	303.0	319.2
lsr7	415.1	389.8	406.5	404.1	357.9	lsa7	315.4	320.5	332.9
lsr8	486.7	498.0	454.8	440.7	399.5	lsa8	340.6	342.9	369.1
lsr9	545.8	446.9	431.1	390.9	378.4	lsa9	347.5	362.5	370.9
lsr10	470.0	473.8	455.9	399.7	376.7	lsa10	339.6	340.9	342.6
Avg cost	473.2	458.8	428.4	397.4	363.9	Avg cost	331.1	339.3	348.7
E6(%)	30.0	26.1	17.7	9.2	0	E6(%)	-5.0	-2.7	0

system is reduced. Overall, the cost of the distribution system could be reduced by 5% if the truck no-go areas become smaller.

Overall, the truck no-go areas have a significant negative impact because they will significantly reduce the efficiency of using trucks. Some customers who could conveniently be delivered by trucks must be served using hub robots or onboard robots. Therefore, city managers need to consider all kinds of factors when setting the truck no-go areas and consider a detailed impact analysis of no-go area variants before implementing them, as they can significantly affect logistics costs.

5.5.2. The Impact of Truck No-go Area Access Time Windows

Our model assumes that truck no-go areas are restricted for all periods. However, some truck no-go areas might be accessible at some times of the day. For example, the city would like to avoid trucks in the no-go areas during peak hours. Therefore, we now study the impact of different access time window lengths on truck no-go areas.

We keep the customer time window unchanged. Then, we add additional access time windows for the truck no-go area. That is, within a specific time window, the truck can visit the customers in the truck no-go area, but outside the time window, trucks are not allowed to visit these customers. Truck no-go area access time windows will lead to a new phenomenon: heterogeneous time windows for customers. A customer has a time window, but different vehicles have different time windows for accessing the customer due to various restrictions, resulting in a shorter time window for a vehicle to visit the customer. For example, customer A is in the truck no-go area and has a time window [3-5]. If the access time window of truck no-go areas is [0-4], then the truck can only serve customer A at the time window [3-4], whereas the robot can serve customer A at the time window [3-5].

In the experiment, we gradually adjust the access time windows of the truck no-go area to observe their impact on the model's output. Table 16 and 17 report the results of the impact of the chosen truck no-go area access time windows. Row 1 represents the truck no-go area access time window, which illustrates that trucks can visit customer nodes in truck no-go areas during this time window. The interval "[1,7]" implies that customers in truck no-go areas are accessible by trucks at times [1,7]. Opt/AC in Row 2 show the optimal/average cost of each instance. Rows 3-12 and 15-24 show detailed average results of each experiment. "Avg cost" represents the average cost of the HTRD model in different time windows of truck no-go areas. E7(%) represents the results gap between the baseline and different truck no-go area access time windows.

Table 16: Impact of Truck No-go Area Access Time Windows (ssr/ssa instances using CPLEX)

TruckNoGoATW	[1-7]	[2-6]	[3-5]	1(baseline)	TruckNoGoATW	[1-7]	[2-6]	[3-5]	large(baseline)
Instances	Opt	Opt	Opt	Opt	Instances	Opt	Opt	Opt	Opt
ssr1	17.37	17.37	17.37	17.37	ssa1	17.41	17.41	17.41	17.41
ssr2	36.51	36.51	36.51	36.51	ssa2	35.84	35.84	35.84	38.28
ssr3	17.79	17.79	17.79	17.79	ssa3	14.42	14.42	14.42	14.42
ssr4	40.89	40.89	40.89	42.95	ssa4	19.57	19.57	19.57	19.57
ssr5	30.45	30.45	30.45	30.45	ssa5	30.39	30.39	30.39	30.39
ssr6	39.40	39.40	39.40	39.40	ssa6	37.13	37.13	37.13	37.13
ssr7	40.63	40.63	40.63	42.12	ssa7	38.66	38.66	38.66	38.74
ssr8	40.53	40.53	40.53	43.21	ssa8	38.51	38.51	38.51	39.46
ssr9	28.67	28.67	28.67	28.67	ssa9	28.13	28.13	28.13	28.13
ssr10	33.08	33.08	33.08	35.03	ssa10	29.52	29.52	29.52	29.52
Avg cost	32.53	32.53	32.53	33.35	Avg cost	28.96	28.96	28.96	29.30
E7(%)	-2.45	-2.45	-2.45	0.00	E7(%)	-1.18	-1.18	-1.18	0.00

Note: When solving the model with CPLEX, the number of virtual stops in this table is set to 1.

Table 16 shows the truck no-go time window experiments results on small-scale instances. This table presents the impact of the truck no-go area access time windows parameter on small-scale instances. For the ssr/ssa-type instances, the results indicate that the access time windows for truck no-go areas affect the objective function. For ssr-type instances, they lead to a 2.45% reduction in the objective function value, while for ssa-type instances, the reduction is

Table 17: Impact of Truck No-go Area Access Time Windows (lsr/lsa instances using ALNS)

TruckNoGoATW	[1-7]	[2-6]	[3-5]	67(baseline)	TruckNoGoATW	[1-7]	[2-6]	[3-5]	large(baseline)
Instances	AC	AC	AC	AC	Instances	AC	AC	AC	AC
lsr1	340.3	340.5	341.6	342.6	lsa1	315.9	316.9	321.6	359.2
lsr2	342.3	351.7	359.4	370.0	lsa2	357.2	359.9	364.9	375.8
lsr3	321.1	357.5	379.9	380.9	lsa3	348.6	349.5	349.3	349.6
lsr4	306.8	312.8	324.6	335.6	lsa4	313.3	314.3	318.3	323.4
lsr5	340.1	341.5	344.1	347.9	lsa5	329.4	333.6	329.7	344.7
lsr6	317.4	325.5	336.5	349.3	lsa6	300.3	303.0	309.7	319.2
lsr7	323.7	337.3	345.0	357.9	lsa7	325.4	328.3	326.4	332.9
lsr8	355.7	361.0	361.0	399.5	lsa8	353.3	356.8	361.0	369.1
lsr9	359.3	363.1	354.7	378.4	lsa9	342.8	349.7	350.6	370.9
lsr10	360.9	351.4	367.9	376.7	lsa10	331.4	331.5	329.1	342.6
Avg cost	336.8	344.2	351.5	363.9	Avg cost	331.8	334.3	336.1	348.7
E7(%)	-7.5	-5.4	-3.4	0.0	E7(%)	-4.9	-4.1	-3.6	0.0

1.18%. However, the size of these time windows has a negligible impact on the objective function in these instances, which may be attributable to stochastic factors in small-scale instances.

Table 17 shows that as the size of the access time window increases, the impact gradually increases. Specifically, when the truck no-go area access time window is two hours ([3,5]), we observe an average 3.4% and 3.6% decrease in the objective function value in lsr and lsa types of instances, respectively. When the truck no-go area access time window is increased to four hours ([2,6]), we see that this leads to a 5.4% and 4.1% decrease in the objective function value in lsr and lsa types of instances, respectively. Subsequent increases in the length of the truck no-go area access time window still affect the reduction of the objective function value.

Overall, the impact of truck no-go areas on logistics performance can be mitigated by setting access time windows for these truck no-go areas. Thus, we can obtain a compromise between the impact of truck distribution on residents' lives and the impact of truck no-go areas on logistics service providers' operations. We also find that the truck no-go area access time windows lead to changes in the number of truck no-go customers in truck no-go areas and also cause heterogeneous time windows for some customers in truck no-go areas.

6. Conclusion and Future Research Directions

Recently, industry and academia have paid attention to robots participating in last-mile delivery. The application scenarios based on TRD and 2E-VRP models have been widely studied, and these studies have demonstrated the potential of robot delivery. This paper combines the characteristics of the TRD and 2E-VRP models in a distribution system. Then, it proposes the HTRD distribution model to automatically select the better distribution strategy of using hubs or parking nodes to maximize distributed resources and improve distribution efficiency. Research shows that the number of solvable cases of the HTRD model is larger than that of TRD and 2E-VRP models under our instances. Even for the instances where all the customers are feasible for three models, the outputs of the HTRD model also outperform those of the 2E-VRP and TRD models. Therefore, allowing the system to select the appropriate distribution mode automatically can help the logistics service providers' distribution accessibility and efficiency.

We propose valid inequalities based on symmetry breaking and time-based lower bounds as a solution approach. Compared with the pure model, the HTRD model with valid inequalities can save more than 70% of the time. We then design two new adaptive methods for classical ALNS algorithms for weight adjustment, one considering the connection between the destroy and repair operators and another considering the difficulty and benefits of obtaining a better solution. Adaptive methods experiments show that the destroy-based repair weight-adjusted method performs well for larger iteration experiments. And destroy-repair combination weight-adjusted performs poorly compared to the classical adaptive, destroy-based repair weight-adjusted, and reward-based weight-adjusted methods.

For sensitivity experiments, we explore the impact of truck no-go areas and truck no-go area access time windows on the output of the delivery system. The truck no-go area access time windows will generate heterogeneous time windows for customers. The experimental results show that as the truck no-go area shrinks, the cost of the distribution system is significantly reduced. Besides, as the length of the truck no-go area access time window increases, the impact gradually decreases. Since setting up truck no-go areas will bring a heavy burden to logistics enterprises, we suggest setting up access time windows for selected truck no-go areas according to local policy criteria to reduce the burden on logistics operations and efficiency.

Future research can investigate more efficient valid inequalities or aim to develop a faster, exact algorithm for the HTRD model based on decomposition methods, for example. Further research avenues are in improving heuristics with a specific focus on finding more effective adaptation methods for the ALNS algorithm so that it performs well with a small or large number of iterations. With regard to the increasing importance of an on-demand economy and same-day deliveries, dynamic versions of the HTRD problems might also be interesting.

Acknowledgements The research is supported by the National Natural Science Foundation of China [grant no. 72301137].

References

Akyol, D. E. and De Koster, R. B. (2018). Determining time windows in urban freight transport: A city cooperative approach. *Transportation Research Part E: Logistics and Transportation Review*, 118:34–50.

Alfandari, L., Ljubić, I., and da Silva, M. D. M. (2022). A tailored benders decomposition approach for last-mile delivery with autonomous robots. *European Journal of Operational Research*, 299(2):510–525.

Alverhed, E., Hellgren, S., Isaksson, H., Olsson, L., Palmqvist, H., and Flodén, J. (2024). Autonomous last-mile delivery robots: a literature review. *European Transport Research Review*, 16(1):4.

Anderluh, A., Nolz, P. C., Hemmelmayr, V. C., and Crainic, T. G. (2021). Multi-objective optimization of a two-echelon vehicle routing problem with vehicle synchronization and 'grey zone'customers arising in urban logistics. *European Journal of Operational Research*, 289(3):940–958.

Andrew, J. H. (2019). Thousands of autonomous delivery robots are about to descend on us college campuses. `https://www.theverge.com/2019/8/20/20812184/starshipdelivery-robot-expansion-college-campus`. Accessed August 20, 2019.

Bakach, I., Campbell, A. M., and Ehmke, J. F. (2021). A two-tier urban delivery network with robot-based deliveries. *Networks*, 2021:1–23.

Boysen, N., Schwerdfeger, S., and Weidinger, F. (2018). Scheduling last-mile deliveries with truck-based autonomous robots. *European Journal of Operational Research*, 271(3):1085–1099.

Breunig, U., Schmid, V., Hartl, R. F., and Vidal, T. (2016). A large neighbourhood based heuristic for two-echelon routing problems. *Computers & Operations Research*, 76:208–225.

Chen, C., Demir, E., and Huang, Y. (2021). An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and delivery robots. *European Journal of Operational Research*, 294(3):1164–1180.

Chevallier, H. (2017). Twinswheel, le livreur du futur. `https://www.franceinter.fr/emissions/cest-deja-demain/cest-deja-demain-29-novembre-2017`. Accessed Novermber 29, 2017.

Christofides, N., Mingozzi, A., and Toth, P. (1981). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical programming*, 20:255–282.

Dellaert, N., Dashty Saridarq, F., Van Woensel, T., and Crainic, T. G. (2019). Branch-and-price–based algorithms for the two-echelon vehicle routing problem with time windows. *Transportation Science*, 53(2):463–479.

Dellaert, N., Van Woensel, T., Crainic, T. G., and Saridarq, F. D. (2021). A multi-commodity two-echelon capacitated vehicle routing problem with time windows: Model formulations and solution approach. *Computers & Operations Research*, 127:105154.

Fontaine, P., Crainic, T. G., Jabali, O., and Rei, W. (2021). Scheduled service network design with resource management for two-tier multimodal city logistics. *European Journal of Operational Research*, 294(2):558–570.

Gu, L. (2018). Delivery robots hit the road in beijing. `http://www.ecns.cn/news/scitech/2018-06-20/detail-ifyvmiee7350792.shtml`. Accessed June 20, 2018.

Gutierrez, A., Labadie, N., and Prins, C. (2024). A two-echelon vehicle routing problem with time-dependent travel times in the city logistics context. *EURO Journal on Transportation and Logistics*, 13:100133.

Heimfarth, A., Ostermeier, M., and Hübner, A. (2022). A mixed truck and robot delivery approach for the daily supply of customers. *European Journal of Operational Research*, 303(1):401–421.

Informs (2021). Enabling on-time, hour-level delivery. `https://www.informs.org/Impact/O.R.-Analytics-Success-Stories/Enabling-On-time-Hour-level-Delivery`. Accessed June 20, 2022.

Joseph, E. B. (2010). Truck restricted streets list. `https://www.cambridgema.gov/traffic/sustainabletransportation/trucks/truckrestrictedstreetslist`. Accessed June 20, 2022.

Li, D. (2017). Jd. com launches robot delivery services in chinese universities. `https://www.chinamoneynetwork.com/2017/06/19/jdcom-launches-robot-delivery-services-in-chinese-universities`. Accessed June 19, 2017.

Li, H., Wang, H., Chen, J., and Bai, M. (2020). Two-echelon vehicle routing problem with time windows and mobile satellites. *Transportation Research Part B Methodological*, 138:179–201.

Li, H., Wang, H., Chen, J., and Bai, M. (2021). Two-echelon vehicle routing problem with satellite bi-synchronization. *European Journal of Operational Research*, 288(3):775–793.

Liu, D., Yan, P., Pu, Z., Wang, Y., and Kaisar, E. I. (2021). Hybrid artificial immune algorithm for optimizing a van-robot e-grocery delivery system. *Transportation Research Part E: Logistics and Transportation Review*, 154:102466.

Madani, B., Ndiaye, M., and Salhi, S. (2024). Hybrid truck-drone delivery system with multi-visits and multi-launch and retrieval locations: Mathematical model and adaptive variable neighborhood search with neighborhood categorization. *European Journal of Operational Research*, 316(1):100–125.

Muñuzuri, J., Grosso, R., Cortés, P., and Guadix, J. (2013). Estimating the extra costs imposed on delivery vehicles using access time windows in a city. *Computers, Environment and Urban Systems*, 41:262–275.

Ostermeier, M., Heimfarth, A., and Hübner, A. (2022). Cost-optimal truck-and-robot routing for last-mile delivery. *Networks*, 79(3):364–389.

Ostermeier, M., Heimfarth, A., and Hübner, A. (2023). The multi-vehicle truck-and-robot routing problem for last-mile delivery. *European Journal of Operational Research*, 310(2):680–697.

Pani, A., Mishra, S., Golias, M., and Figliozzi, M. (2020). Evaluating public acceptance of autonomous delivery robots during covid-19 pandemic. *Transportation research part D: transport and environment*, 89:102600.

Pisinger, D. and Ropke, S. (2019). Large neighborhood search. In *Handbook of Metaheuristics*, pages 99–127. Springer.

Roberti, R. and Wen, M. (2016). The electric traveling salesman problem with time windows. *Transportation Research Part E: Logistics and Transportation Review*, 89:32–52.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472.

Sacramento, D., Pisinger, D., and Ropke, S. (2019). An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. *Transportation Research Part C: Emerging Technologies*, 102:289–315.

Schermer, D., Moeini, M., and Wendt, O. (2019). A matheuristic for the vehicle routing problem with drones and its variants. *Transportation Research Part C: Emerging Technologies*, 106:166–204.

Simoni, M. D., Kutanoglu, E., and Claudel, C. G. (2020). Optimization and analysis of a robot-assisted last mile delivery system. *Transportation Research Part E: Logistics and Transportation Review*, 142:102049.

Sluijk, N., Florio, A. M., Kinable, J., Dellaert, N., and Van Woensel, T. (2023). Two-echelon vehicle routing problems: A literature review. *European Journal of Operational Research*, 304(3):865–886.

Sonneberg, M.-O., Leyerer, M., Kleinschmidt, A., Knigge, F., and Breitner, M. H. (2019). Autonomous unmanned ground vehicles for urban logistics: Optimization of last mile delivery operations.

Srinivas, S., Ramachandiran, S., and Rajendran, S. (2022). Autonomous robot-driven deliveries: A review of recent developments and future directions. *Transportation research part E: logistics and transportation review*, 165:102834.

Tamke, F. and Buscher, U. (2021). A branch-and-cut algorithm for the vehicle routing problem with drones. *Transportation Research Part B: Methodological*, 144:174–203.

Wang, F., Li, H., and Xiong, H. (2024). Truck–drone routing problem with stochastic demand. *European Journal of Operational Research*.

Wang, Y., Li, Q., Guan, X., Xu, M., Liu, Y., and Wang, H. (2021). Two-echelon collaborative multi-depot multi-period vehicle routing problem. *Expert Systems with Applications*, 167:114201.

Wang, Z. and Sheu, J.-B. (2019). Vehicle routing problem with drones. *Transportation research part B: methodological*, 122:350–364.

Yu, S., Puchinger, J., and Sun, S. (2020). Two-echelon urban deliveries using autonomous vehicles. *Transportation Research Part E: Logistics and Transportation Review*, 141:102018.

Yu, S., Puchinger, J., and Sun, S. (2022). Van-based robot hybrid pickup and delivery routing problem. *European Journal of Operational Research*, 298(3):894–914.

Yu, S., Puchinger, J., and Sun, S. (2024). Electric van-based robot deliveries with en-route charging. *European Journal of Operational Research*, 317(3):806–826.

Zhang, S., Campbell, A. M., and Ehmke, J. F. (2019). Impact of congestion pricing schemes on costs and emissions of commercial fleets in urban areas. *Networks*, 73(4):466–489.

Zhao, B., Zhang, J., and Wei, W. (2019). Impact of time restriction and logistics sprawl on urban freight and environment: The case of beijing agricultural freight. *Sustainability*, 11(13):3675.

Zhao, J., Poon, M., Tan, V. Y., and Zhang, Z. (2024). A hybrid genetic search and dynamic programming-based split algorithm for the multi-trip time-dependent vehicle routing problem. *European Journal of Operational Research*, 317(3):921–935.

Appendix A. ALNS algorithm

The proposed ALNS framework, introduced by Yu et al. (2022), consists of the following steps.

The algorithm starts with an initial feasible solution and iteratively improves it until a predefined stopping criterion is met. In this paper, we use the maximum number of iterations (MaxIteNum) as the stopping criterion. In each iteration, a destroy algorithm partially deconstructs the current solution by removing some nodes or routes. Then, a repair algorithm heuristically reconstructs the solution by adding nodes or routes, aiming to obtain a better solution quality. If the solution quality does not improve after a certain number of iterations (MaxNonImp), the algorithm restarts from a new initial solution.

To ensure the feasibility of the reconstructed solutions, we employ a route evaluation method during the repair operations. Moreover, we adopt a simulated annealing-like acceptance criterion that allows the exploration of inferior solutions in the search space. Furthermore, using a weight-adjusting method, we assign and adjust weights to each destroy-and-repair operation according to their performance. This enables the algorithm to adapt to the characteristics of the problem instance and the state of the search. We follow the simulated annealing and weight-adjusting methods proposed by Pisinger and Ropke (2019).

The acceptance probability of the simulated annealing is governed by a temperature parameter T. A candidate solution x^t is always accepted if it improves the objective value of the current best solution x^b ($c(x^t) <= c(x^b)$), and it is accepted with probability $\exp(c(x^b) - c(x^t)/T)$ otherwise. Here $T > 0$ is the current temperature. The initial temperature is set to $T_{st} > 0$ and is gradually reduced by applying an update $T = \alpha \times T$ at each iteration, where $\alpha \in [0, 1]$ is the cooling rate of simulated annealing.

Algorithm 1 presents the proposed ALNS algorithm. The best solution is stored in *recordset* (Line 1). The algorithm begins with an initial solution x, also assigned as the current best solution x^b (Line 2). The counter *NumNunImp* is then initialized to 0 (Line 3), which records the number of iterations without improvement until it exceeds *MaxNunImp*. Then, a destroy-and-repair operation is executed until the stopping criterion (*MaxIteNum*) is reached (Line 4-19). The *restart*() function restarts the ALNS from Line 2 while keeping the iteration number unchanged.

Algorithm 1 *Adapt_Large_Neighborhood_Search*()

1: $recordset = \{\}$
2: **Initialization:** use *RouteReconstruction* operator to get an initial solution x, let $x^b = x$.
3: $NumNunImp = 0$
4: **while** stopping criterion is not met **do**
5: select destroy and repair methods from set.
6: $x^t = r(d(x))$
7: **if** simulated annealing criterion accepts the solution **then**
8: $x = x^t$
9: **end if**
10: **if** $c(x) < c(x^b)$ **then**
11: $x^b = x$, $NumNunImp = 0$
12: **else**
13: $NumNunImp+ = 1$
14: **if** restart conditions is met **then**
15: $recordset.append(x^b)$ and then $restart()$
16: **end if**
17: **end if**
18: update weights and selection parameters
19: **end while**
20: **return:** output best x^b from *recordset*

Appendix B. Destroy and repair operators

Destroy operators:

1. **Random customer removal** operator randomly removes customer nodes from an HTRD route.
2. **Greedy customer removal** operator removes the customer that can yield the largest cost reduction for a given route.
3. **Parking-route removal** operator randomly removes a parking node, and the robot routes depart from and arrive at this parking node are all removed.
4. **Random route destruction** operator randomly selects an HTRD route in the solution to destroy.
5. **Greedy route destruction** operator selects an HTRD route with a minimum number of customer nodes to destroy.

Repair operators:

1. **Route reconstruction** operator ensures we always get a feasible solution. We adopt the same method as used to get the initial solution.
2. **Random customer insertion** operator randomly chooses a customer to insert into a random feasible position in a given route until all customers have been tried.
3. **Greedy customer insertion** operator randomly chooses a customer to insert into a position in the route with the least cost increases.
4. **Random parking-route insertion** operator randomly chooses a parking node to insert into a given route and then generates a robot route from this parking node with one customer node.
5. **Greedy parking-route insertion** operator chooses a parking node to insert into a given route and generates a robot route from this parking node with one customer node, with the least total travel cost increases.

Appendix C. Equipment setting instructions

We have set the parameters for our robot by referencing those of Starship Technologies. The robot's speed is set at 6km/h, a safe walking speed. The battery capacity is set at 1.5 kWh, with an energy consumption rate of 0.1 kWh/km. This allows the robot to travel a distance of 15km, with a single trip covering 7.5km. Since our robot can visit multiple customers in one trip, we have set a slightly larger load capacity of 50kg. As for the operating cost, Starship's is \$1-\$2 per 15km trip. In our simulation experiment, we have set the travel cost rate at \$0.1/km.

For the truck, its operational speed is set at 25km/h in urban deliveries, considering the actual conditions of city roads. The load capacity is set at 500kg, a compromise between weight and volume considerations, guaranteed delivery to a certain number of customers. The travel cost rate for the truck is set at $0.6/km, which is determined based on the operating and maintenance costs of the equipment.