

A HYBRID VARIABLE NEIGHBORHOOD SEARCH FOR A REAL-WORLD PETROL STATION REPLENISHMENT PROBLEM WITH MULTI-COMPARTMENT VEHICLES

ZHISHUO LIU¹, YINAN CHENG¹  AND FANG TIAN^{2,*} 

Abstract. Studies on the Petrol Station Replenishment Problem (PSRP) have many challenges because of the limitations in real-world PSRP, including diverse oil products, various vehicle models, multiple compartments in each vehicle, a finite set of vehicles, restricted tank capacities of petrol stations, and the preparation time and cost for discharge. To address those problems, this paper proposes a novel PSRP model considering the following realistic scenarios: each vehicle has multiple compartments and is allowed for multiple trips; vehicles are heterogeneous; petrol stations have hard time windows; external vehicles can be rented when internal vehicles are not enough; after arrival, each vehicle must prepare for a certain time before discharge. The objective is to minimize the overall costs including vehicles' fixed costs, traveling costs, and preparation costs. A Hybrid Variable Neighborhood Search algorithm (HVNS) combined with a Simulated Annealing-based acceptance criterion is developed to solve the problem. Various experiments are conducted on different scale instances and a practical application of PetroChina. Experimental results show that compared with CPLEX and other heuristics, HVNS can effectively solve the problem in terms of solution quality and computing time.

Mathematics Subject Classification. 90B06.

Received March 13, 2023. Accepted January 12, 2025.

1. INTRODUCTION

From refineries to markets, refined oil products are first transported to oil depots in different local areas by sea, rail, and pipelines, and then from each depot to different stations by road. The process of vehicle transportation is referred to petrol station replenishment; the optimization of vehicle routing becomes an important means to reduce costs and increase the efficiency for petrochemical companies.

Studies on the Petrol Station Replenishment Problem (PSRP) used to focus on homogenous vehicles with single compartments. Nowadays, researchers consider heterogeneous vehicles with multiple compartments and some other practical assumptions. However, compared to real-life examples, *e.g.*, the application of PetroChina, there still exist certain gaps between literature and real life. First, in real life, a vehicle's arrival time at a

Keywords. Petrol station replenishment, vehicle routing problem, multiple compartments, multiple trips, variable neighborhood search.

¹ School of Traffic and Transportation, Beijing Jiaotong University, No. 3 Shangyuancun, Haidian District, Beijing 100044, P.R. China.

² Business Administration Division, Seaver College, Pepperdine University, 24255 Pacific Coast Hwy, Malibu, CA 90263, USA.

*Corresponding author: fang.tian@pepperdine.edu

petrol station is restricted by the real-time inventory of each oil product. If it arrives too early, such that the inventory is still high and there is not enough space to store the volume delivered, the vehicle will have to wait. If it arrives too late, such that an oil product is sold out, the station will have a stockout. Second, vehicles can make multiple trips during an operation period, *e.g.*, a day. Meanwhile, because of different rates of sales and inventory of different products at a station, multiple vehicle deliveries are necessary. However, third, the number of vehicle deliveries should be minimized, as upon the arrival at a station, vehicles need to prepare for discharge, incurring a preparation cost. Last, as self-owned (internal) vehicles are limited, when there are not enough, rental (external) vehicles are needed.

Currently, no literature can comprehensively capture all the above-mentioned scenarios. To fill the literature gap, in this paper we establish an innovative model that is referred to as a Multi-compartment, Multi-trip, Heterogeneous-fleet Vehicle Routing Problem with Time Windows (MMHVRPTW). It is a complicated Vehicle Routing Problem (VRP) with six features: (i) there are multiple compartments in each vehicle; (ii) a vehicle can make multiple trips during an operation period; (iii) vehicles are heterogeneous; (iv) the delivery of each product at each station has a hard time window; (v) one compartment can serve multiple stations; (vi) when internal vehicles are not enough, external vehicles can be rented; and (vii) after arrival, vehicles must prepare (park and rest) for a certain time before discharge, to eliminate the static electricity generated on the route. To solve this problem, we propose a Hybrid Variable Neighborhood Search algorithm (HVNS), which introduces a Simulated Annealing-based acceptance criterion into the original Variable Neighborhood Search (VNS) and makes the algorithm accept worse solutions with a certain probability. In addition, a neighborhood, Remove Vehicle, is devised to extend the scope of the neighborhood search. The model and method are validated through a series of experiments on small-scale instances generated from real-life data, benchmark instances, and a practical application of PetroChina. Therefore, the study can provide theoretical and decision support to petrochemical companies in making petrol station replenishment plans.

The paper contributes to the literature and application from the following aspects.

- (1) The paper extends the vehicle routing problem with time windows to a multi-compartment, multi-trip, and heterogeneous-vehicle scenario setting. It is among the first research considering the following features in the real-life scenarios of the PSRP: the restricted tank capacities of petrol stations, external vehicle rentals, and vehicles' preparation time and cost before discharge.
- (2) HVNS with a Simulated Annealing-based acceptance criterion is developed for MMHVRPTW. A neighborhood structure, Remove Vehicle with three insertion operators, is designed for HVNS, effectively improving the original VNS performance.
- (3) The model and algorithm are efficient to solve problems of different scales from PetroChina.

The paper is structured as follows. Section 2 reviews the related literature. Section 3 describes the characteristics of the problem. Section 4 establishes the mathematical model. Section 5 proposes HVNS together with several neighborhood structures to solve the problem. Section 6 conducts a series of experiments to test the performance and effectiveness of the algorithm. Section 7 concludes the paper.

2. LITERATURE REVIEW

For the small-category classification, MMHVRPTW is a variant of the Multi-Compartment Vehicle Routing Problem (MCVRP). For the large category, it belongs to the petrol station replenishment problem. In this section, we summarize studies on those two problems.

Multi-compartment vehicle routing problem. The MCVRP has been widely used across different areas, including classified waste collection [19,26,34], gasoline delivery [8,24,40], grocery delivery [5,7,25,30], agriculture contexts [23], etc. In those applications, multiple compartments are used to deliver different products that cannot be mixed on the same vehicle. According to the practical needs of different MCVRP problems, researchers mainly discuss the following factors in their studies: (1) split customers (whether a customer can be visited several times for delivering different types of products) [23,34], (2) split deliveries (whether a customer can be visited for

several times for delivering one type of products) [8, 24]; (3) split compartments (whether a compartment can deliver a type of products to multiple customers) [8, 12]; (4) customers' inventory [8]; (5) heterogeneous vehicles [23, 33]; (6) multiple trips [12, 24]; (7) multiple depots [1, 12]; (8) multiple periods [1, 10]; and (9) flexible compartment sizes [17, 19].

Reviewing the methodology to solve the MCVRP, exact methods and heuristics are widely adopted. Currently, exact methods used in MCVRP problems include the branch-and-cut method, branch-and-price method, branch-and-bound algorithm, column-generation algorithm, and Lagrangian method. Lahyani *et al.* [23] discuss an olive oil recovery problem in Tunisia. Vehicles are allowed to make multiple trips, but different levels of olive oil cannot be mixed. In a compartment, if the oil level on a later trip is higher than that on the previous trip, the compartment needs to be cleaned. The problem is solved by the branch-and-cut method. Coelho and Laporte [8] study four multi-compartment gasoline distribution problems and develop several branch-and-cut methods for the multi-period inventory routing problem. Heßler [17] discusses the MCVRP in a flexible compartment size setting, including continuous and discrete sizes. Two branch-and-cut algorithms and a branch-price-and-cut algorithm are developed in the study.

As the MCVRP is NP-hard, exact approaches are unable to solve large-scale problems within an acceptable time. About 76% of MCVRP problems are solved by heuristic algorithms [31]. Mendoza *et al.* [28] is an application of classical construction heuristics. The MCVRP has random demands; three constructive heuristics are proposed based on stochastic programming. There are many applications of population search heuristics. Reed *et al.* [34] develop an ant colony optimization to solve a basic MCVRP for the collection of recycling waste from households. Rabbani *et al.* [33] study the MCVRP for garbage collection using both internal and external vehicles. They develop two hybrid genetic algorithms and show that both algorithms are superior to the genetic algorithm. The local search meta-heuristics also have various applications. Henke *et al.* [19] propose an MCVRP with flexible compartment sizes for collecting glass waste of different colors from customers. The fleet is homogeneous, but the compartment size can vary discretely, and the number of compartments can be fewer than the number of product types. The authors develop a heuristic of VNS to solve this problem. Alinaghian and Shokouhi [1] study the multi-depot MCVRP. There are multiple products; at each depot, the inventory of each product and vehicle number are certain. A hybrid adaptive large neighborhood search (ALNS) is proposed to solve the problem. Chen *et al.* [7] focus on an MCVRP with time windows for fresh food delivery and develop a VNS method to solve it. Eshtehadi *et al.* [15] consider a realistic homogeneous MCVRP with hard time windows and speed optimization for ambient, chilled, and frozen products. An enhanced ALNS is developed.

MMHVRPTW proposed in this paper uses heterogeneous vehicles, each with multiple compartments. Customers can be split, because a petrol station may be visited several times to deliver different types of products although only one visit is allowed for delivery of each type of product. Compartments can also be split, as the product in each compartment can fulfill the demand of multiple customers. All stations have hard time windows. All vehicles can make multiple trips. In addition, this paper considers some other practical constraints in the petrol station replenishment problem, such as the necessary preparation time before discharging products to customers, the limited number of self-owned internal vehicles, and available external vehicles for rent. Compared with other literature on the MCVRP, this paper is much more complex, increasing the difficulty of solving the problem.

Petrol station replenishment problem. In real life, some petrochemical companies use vehicles with single compartments for oil product delivery, requiring each vehicle to deliver only one product on each trip. Some scholars use single-compartment vehicles in their PSRP studies. For example, Wei *et al.* [41] focus on a PSRP with single-compartment vehicles. They discuss unknown demands, allowing multiple trips made by a vehicle and multiple visits to a tank, but requiring a vehicle to visit one tank on each trip. A discrete time mathematical model in the initiative distribution mode is developed in this study.

Scholars develop various methods to solve the PSRP with multi-compartment vehicles. Avella *et al.* [2] employ the branch-and-price method to solve a PSRP with compartmentalized compartments. Yahyaoui *et al.* [42] aim to minimize the total travel distance in their study on a homogeneous-fleet MCVRP model. An adaptive VNS

and a genetic algorithm are developed to solve the PSRP. Boctor *et al.* [3] study a trip packing problem as a sub-problem of the PSRP. They assign existing trips to a heterogeneous fleet. Coelho and Laporte [8] compare four problems in the multi-compartment gasoline distribution: split tanks and split compartments, spit tanks and unsplit compartments, unsplit tanks and split compartments, and unsplit tanks and unsplit compartments. Several exact branch-and-cut algorithms are developed to solve those problems.

Studies that are close to this paper can show how this paper fills the gap between current literature and practice. Wang *et al.* [40] study a PSRP with time windows and multiple trips. The objective is to minimize the total fixed and travel costs, not including vehicle preparation costs incurred upon each visit before discharge. Wang *et al.* [40] discuss a split-tank and split-compartment PSRP, aiming to minimize the duration. The tank capacity, accommodating time, stockout time, external vehicles, and vehicles' preparation process are ignored in their study. Cornillier *et al.* [9–12] conduct a series of studies on the MCVRP for the PSRP. Cornillier *et al.* [9] adopt an accurate algorithm to solve MCVRP with a single depot and multiple vehicles. On this basis, Cornillier *et al.* [10] study the MCVRP with a limited number of vehicles. Then, in [11], time window constraints are introduced, and two heuristics are designed. The closest study among all is [12], who consider constraints of multiple depots, heterogeneous vehicles with multiple compartments, a limited number of vehicles, hard time windows, and unsplit oil tanks at petrol stations. Aiming to maximize profits, the authors construct a heuristic based on the route selection model. Compared to our study, this paper has the following limitations. (i) They assume that each station can be visited only once, requiring the demand on all products must be fulfilled by one trip of one vehicle. In real life, however, products have various rates of sale, creating a lot of differences in the demand, accommodating time, and stockout time of different products. It is difficult and inefficient to use one vehicle to satisfy the constraints required by all products at a station. (ii) They assume that internal vehicles are limited but do not consider using external vehicles, which is more restricted than real-life experience. (iii) There is no constraint on the tank capacity in their study. (iv) After arrival, vehicles must prepare for a certain time before discharge, to eliminate the static electricity generated on the route. This paper does not consider such a need. (v) This study uses unsplit compartments, requiring that each compartment can serve only one station, which can be relaxed in real life.

Compared with studies in literature, our paper is based on the practical needs of PetroChina, making a series of applicable assumptions. We allow different vehicles to deliver different products to a station and one compartment to serve multiple stations, consider the tank capacity at petrol stations, require that a discharge must be after the accommodating time and before the stockout time, consider vehicles' preparation time and cost for the static elimination before discharge, and allow to rent external vehicles when internal vehicles are not enough. Those constraints all make the PSRP studied in this paper more complex and difficult to formulate and solve.

3. PROBLEM DESCRIPTION

We study a directed network with $N + 2$ nodes, including N petrol stations (denoted by nodes $1, 2, \dots, N$) and one oil depot (denoted by node 0 as the starting node of a trip and by node $N + 1$ as the ending node of the trip). $V^* = \{1, 2, \dots, N\}$ is the set of petrol stations and $V = \{0, 1, \dots, N + 1\}$ is the set of nodes. Let $V^0 = \{0\} \cup V^*$ and $V^{N+1} = \{N + 1\} \cup V^*$. The distance between nodes $i, j \in V$ is denoted by d_{ij} . Note that $d_{ij} = 0$ if $i = j$ or if $i, j \in \{0, N + 1\}$.

Assume that $n + 1$ products are stored at the depot and delivered to petrol stations, including n oil products and one virtual non-oil product, which is specifically for the depot use only. $R = \{0, 1, \dots, n\}$ is the set of products. The inventory of all products at the depot is sufficient. The demand of product $r \in R$ at node $i \in V$ is denoted by Q_i^r , which is known. $Q_0^r = Q_{N+1}^r = 0$ for all r .

Vehicles deliver products from the depot to petrol stations. The fleet owns $K > 1$ internal heterogeneous vehicles; we use $L_1 = \{1, 2, \dots, K\}$ to denote the set of internal vehicles. When the internal vehicles are not enough, unlimited external homogeneous vehicles are available for rent. We use $L_2 = \{K + 1, \dots, M\}$ to denote

the set of external vehicles, where $M \gg K$. $L = L_1 \cup L_2$ is the set of all vehicles. Vehicle $k \in L$ has a fixed cost β_k and a unit mileage cost α_k . External vehicles incur higher fixed costs than internal vehicles.

Products cannot be mixed. All vehicles have multiple compartments so that they can carry multiple products in different compartments. One compartment can carry only one product on each trip but may deliver to multiple stations. A compartment can also carry different products on different trips. Vehicle k has F_k homogeneous compartments; $D_k = \{1, 2, \dots, F_k\}$ is the set of its compartments. Compartments are not necessary to be fully loaded but cannot be overloaded; each compartment of vehicle k has a capacity of c_k . Vehicle k 's total capacity $C_k = F_k c_k$.

The depot and all stations are open during an operation period, *e.g.*, a day, which is denoted by $[0, T]$. Between two operation periods, *e.g.*, overnight, vehicles park at the depot. After the depot opens at time 0, vehicles load products, visit some petrol stations, discharge the products, and then return to the depot – this entire process is considered a trip. Vehicles' loading time is known as t_{load} ; they travel at a known constant speed $v_{vehicle}$, and their discharging speed is v_d . They can make multiple trips during $[0, T]$ as long as the last trip is completed before T . We refer to $M_k = \{1, \dots, W_k\}$, the set of trips made by vehicle k , as the tour of the vehicle. $W_k = \frac{\sum_{i \in V^*} \sum_{r \in R} Q_i^r}{C_k}$ is the maximum number of trips vehicle k can make.

After arriving at a petrol station, a vehicle must spend some time, p , preparing for discharge. An associated preparation cost, γ , is then incurred, regardless of the number or volume of products discharged at the station. Therefore, petrol stations prefer less frequent vehicle arrivals. For this reason, split deliveries of the same product at a station are not allowed. Then, if a vehicle carries product r to station i on a trip, the delivery volume must equal to Q_i^r , the station's demand for the product. Besides, Q_i^r cannot exceed the smallest vehicle capacity. Moreover, the total visits to the station cannot exceed the total number of products at the station. Note that split deliveries of different products are allowed.

At each petrol station, each oil product is stored in a tank, with a known capacity; H_i^r is the capacity of the tank storing product r at station i . The inventory in each tank decreases at a known rate, denoted by s_i^r , referred to as product r 's rate of sale at station i . When a vehicle starts to discharge product r at station i , the available space of the corresponding tank must be able to accommodate (greater than or equal to) the delivered (ordered) volume, Q_i^r . We mark the moment when the available space equals to the delivered volume and refer to it as the accommodating time of product r at station i , denoted by e_i^r . Denoting the initial inventory (at time 0) of product r at station i by I_i^r , which is known, the accommodating time is calculated as:

$$e_i^r = \begin{cases} \frac{I_i^r + Q_i^r - H_i^r}{s_i^r}, & \text{if } I_i^r + Q_i^r - H_i^r \leq 0; \\ 0, & \text{else.} \end{cases} \tag{1}$$

Products can never run out of stock. That is, before a vehicle discharges a product, its inventory in the tank must be greater than zero. We refer to the moment when the product inventory reaches zero as the stockout time of product r at station i , denoted by l_i^r :

$$l_i^r = \frac{I_i^r}{s_i^r}. \tag{2}$$

To guarantee that tanks do not overflow, vehicles cannot start discharge before the accommodating time. If they are ready (for discharge) earlier than the accommodating time, they must wait. To avoid stockout, vehicles cannot start discharge after the stockout time. Then, e_i^r and l_i^r determine station i 's time window for vehicles to discharge product r : $[e_i^r, l_i^r]$.

4. MATHEMATICAL MODEL

Section 3 describes a vehicle routing problem with hard time windows for heterogeneous vehicles, each with multiple compartments and allowed to make multiple trips (MMHVRPTW). In this section, we establish its mathematical model.

To minimize the total costs, the problem needs to determine the stations each vehicle visits and the products it delivers on each trip, and the arrival time and discharge starting time at each station for each product. The decision variables are described below.

Z_{ikmd}^r : binary variable; $Z_{ikmd}^r = 1$ indicates that products in compartment $d \in D_k$ of vehicle $k \in L$ on trip $m \in M_k$ will be used to fulfill the demand of station $i \in V^*$ on product $r \in R$.

$x_{ijkm}^{r_1, r_2}$: binary variable; $x_{ijkm}^{r_1, r_2} = 1$ indicates that after delivering product $r_1 \in R$ to node $i \in V^0$ on trip $m \in M_k$, vehicle $k \in L$ will continue to deliver product $r_2 \in R$ to node $j \in V^{N+1}$.

t_{ikm1}^r : the moment when vehicle $k \in L$ arrives at node $i \in V$ on trip $m \in M_k$ for delivering product $r \in R$.

t_{ikm2}^r : the moment when vehicle $k \in L$ starts to discharge product $r \in R$ at node $i \in V^*$ on trip $m \in M_k$.
 $t_{0k12}^r = t_{load}$.

The objective of MMHVRPTW is to minimize the total cost, as shown in equation (3), including (i) the total mileage costs of all vehicles, as shown in equation (4), (ii) the total fixed costs of all vehicles, as shown in equation (5), and (iii) the total preparation costs, as shown in equation (8). Binary variable $S_k = 1$ indicates that vehicle k is used. Binary variable $E_{km} = 1$ indicates that trip $m \in M_k$ of vehicle k is used. N_{visit} calculates the number of all visits. N_{order} counts the number of all orders. Binary variable $p_i^r = 1$ indicates that station i has demand on product r .

$$\min \text{cost} = \text{cost}_1 + \text{cost}_2 + \text{cost}_3; \tag{3}$$

$$\text{cost}_1 = \sum_{r_1, r_2 \in R} \sum_{j \in V^0} \sum_{j \in V^{N+1}} \sum_{k \in L} \sum_{m \in M_k} \alpha_k d_{ij} x_{ijkm}^{r_1, r_2}; \tag{4}$$

$$\text{cost}_2 = \sum_{k \in L} S_k \beta_k; \tag{5}$$

$$S_k = \begin{cases} 1, & \text{if } \sum_{m \in M_k} E_{km} \geq 1, \\ 0, & \text{else} \end{cases}, \quad \forall k \in L; \tag{6}$$

$$E_{km} = \begin{cases} 1, & \text{if } \sum_{r_1, r_2 \in R} \sum_{i \in V^0} \sum_{j \in V^{N+1}} x_{ijkm}^{r_1, r_2} \geq 1, \\ 0, & \text{else} \end{cases}, \quad \forall k \in L \text{ and } m \in M_k; \tag{7}$$

$$\text{cost}_3 = \gamma N_{visit}; \tag{8}$$

$$N_{visit} = N_{order} - \sum_{r_1, r_2 \in R} \sum_{i \in V^*} \sum_{k \in L} \sum_{m \in M_k} x_{iikm}^{r_1, r_2}; \tag{9}$$

$$N_{order} = \sum_{i \in V, r \in R} p_i^r; \tag{10}$$

$$p_i^r = \begin{cases} 1, & \text{if } Q_i^r > 0, \\ 0, & \text{else} \end{cases}, \quad \forall r \in R \text{ and } i \in V. \tag{11}$$

Constraints (12)–(19) describe the routing requirements of MMHVRPTW. Constraints (12) and (13) guarantee that if $Q_i^r = 0$, no vehicle will visit station i to deliver product r , but if $Q_i^r > 0$, station i will be visited only once to receive the delivery of product r . Constraint (14) requires that after a vehicle fulfills an order, it must leave the corresponding tank. Constraints (15) and (16) imply that all vehicles must depart from and return to the depot on each trip. Constraints (17) and (18) explain the relationship between $x_{ijkm}^{r_1, r_2}$ and Z_{jkmd}^r , indicating that vehicles are available for all orders. Constraint (19) guarantees that a vehicle cannot start trip $m + 1$ unless trip m is completed.

$$\sum_{r_1 \in R} \sum_{i \in V^0} \sum_{k \in L} \sum_{m \in M_k} x_{ijkm}^{r_1, r_2} = p_j^{r_2}, \quad \forall j \in V^* \text{ and } r_2 \in R; \tag{12}$$

$$\sum_{r_2 \in R} \sum_{j \in V^{N+1}} \sum_{k \in L} \sum_{m \in M_k} x_{ijkm}^{r_1, r_2} = p_i^{r_1}, \quad \forall i \in V^* \text{ and } r_1 \in R; \tag{13}$$

$$\sum_{i \in V^0} \sum_{r_1 \in R} x_{ijkm}^{r_1, r_2} = \sum_{i \in V^{N+1}} \sum_{r_1 \in R} x_{jikm}^{r_2, r_1}, \quad \forall j \in V^* \text{ and } r_2 \in R; \quad (14)$$

$$\sum_{j \in V^*} \sum_{r \in R} x_{0jkm}^{0, r} = \sum_{i \in V^*} \sum_{r \in R} x_{i(N+1)km}^{r, 0} = E_{km}, \quad \forall k \in L \text{ and } m \in M_k; \quad (15)$$

$$\sum_{j \in V^*} \sum_{r_2 \in R} \sum_{r_1 \in R \setminus \{0\}} x_{0jkm}^{r_1, r_2} = \sum_{i \in V^*} \sum_{r_2 \in R} \sum_{r_1 \in R \setminus \{0\}} x_{i(N+1)km}^{r_2, r_1} = 0, \quad \forall k \in L \text{ and } m \in M_k; \quad (16)$$

$$\sum_{d \in D_k} Z_{ikmd}^{r_1} \geq \sum_{j \in V^{N+1}} \sum_{r_2 \in R} x_{ijkm}^{r_1, r_2}, \quad \forall k \in L, m \in M_k, r_1 \in R, \text{ and } i \in V^*; \quad (17)$$

$$\sum_{d \in D_k} Z_{jkmd}^{r_2} \geq \sum_{i \in V^0} \sum_{r_2 \in R} x_{ijkm}^{r_1, r_2}, \quad \forall k \in L, m \in M_k, r_2 \in R, \text{ and } j \in V^*; \quad (18)$$

$$E_{km} \geq E_{k(m+1)}, \quad \forall k \in L \text{ and } m \in M_k. \quad (19)$$

Constraints (20)–(27) describe the vehicle use and loading requirements. Constraint (20) guarantees that external vehicles will not be used until all internal vehicles are used. Constraint (21) supports the requirement that vehicles cannot be overloaded. Constraint (22) defines a binary variable; $B_{ikm}^r = 1$ indicates that the demand on product r at station i is fulfilled by vehicle k on trip m . Constraints (23) and (24) require that if a station has demand on a product, it must be fulfilled by one vehicle, although multiple compartments may be used. Constraint (25) defines a binary variable; $y_{kmd}^r = 1$ indicates that vehicle k carries product r in compartment d on trip m . Constraint (26) guarantees that on each trip, one compartment can only carry one product. Constraint (27) supports the requirement that compartments cannot be overloaded.

$$S_k \geq S_{k'}, \quad \forall k \in L_1 \text{ and } k' \in L_2; \quad (20)$$

$$\sum_{r_1, r_2 \in R} \sum_{i \in V^0} \sum_{j \in V^{N+1}} x_{ijkm}^{r_1, r_2} Q_j^{r_2} \leq C_k, \quad \forall k \in L \text{ and } m \in M_k; \quad (21)$$

$$B_{ikm}^r = \begin{cases} 1, & \text{if } \sum_{d \in D_k} Z_{ikmd}^r \geq 1, \\ 0, & \text{else} \end{cases}, \quad \forall i \in V^*, k \in L, m \in M_k, \text{ and } r \in R; \quad (22)$$

$$\sum_{k \in L} \sum_{m \in M_k} B_{ikm}^r = p_i^r, \quad \forall i \in V^* \text{ and } r \in R; \quad (23)$$

$$\sum_{d \in D_k} \sum_{k \in L} \sum_{m \in M_k} Z_{ikmd}^r \geq p_i^r, \quad \forall i \in V^* \text{ and } r \in R; \quad (24)$$

$$y_{kmd}^r = \begin{cases} 1, & \text{if } \sum_{i \in V^*} Z_{ikmd}^r \geq 1, \\ 0, & \text{else} \end{cases}, \quad \forall k \in L, m \in M_k, d \in D_k, \text{ and } r \in R; \quad (25)$$

$$\sum_{r \in R} y_{kmd}^r \leq 1, \quad \forall k \in L, m \in M_k, \text{ and } d \in D_k; \quad (26)$$

$$\sum_{i \in V^*} B_{ikm}^r Q_i^r \leq \sum_{d \in D_k} y_{kmd}^r C_k, \quad \forall k \in L, m \in M_k, \text{ and } r \in R. \quad (27)$$

Constraints (28)–(34) describe the time requirements. Constraint (28) represents the time relationship between two consecutive trips of a vehicle. Constraint (29) indicates that the time a vehicle starts to discharge at a station must be within the time window of the station. Constraint (30) defines a binary variable. $g_{ikm}^r = 0$ indicates that product r is the first product discharged by vehicle k at station i during a visit of trip m . In this case, the vehicle must park and rest for time p after arrival and before discharge, incurring a preparation cost, γ . If there exists $r_1 \in R$ such that $x_{iikm}^{r_1, r} = 1$, then $g_{ikm}^r = 1$, indicating that no preparation is need for vehicle k to discharge product r at station i during a visit of trip m . Constraint (31) defines wt_{ikm}^r , the wait time of vehicle k for product r of station i on trip m ; $wt_{(N+1)km}^r = 0$. Constraints (32) and (33) represent

the time relationship between two consecutive orders of a vehicle. Constraint (34) ensures vehicles return to the depot within the operation period.

$$t_{0k(m+1)2}^r \geq t_{(N+1)km1}^r + t_{\text{load}}, \quad \forall k \in L, m \in M_k, \text{ and } r \in R; \quad (28)$$

$$e_i^r \leq t_{ikm2}^r \leq l_i^r, \quad \forall i \in V^*, k \in L, m \in M_k, \text{ and } r \in R; \quad (29)$$

$$g_{ikm}^r = \begin{cases} 1, & \text{if } \sum_{r_1 \in R} x_{iikm}^{r_1, r} = 1, \\ 0, & \text{else} \end{cases}, \quad \forall i \in V^*, k \in L, m \in M_k, \text{ and } r, r' \in R; \quad (30)$$

$$wt_{ikm}^r = \max\{0, e_i^r - t_{ikm1}^r - (1 - g_{ikm}^r)p\}, \quad \forall i \in V^*, k \in L, m \in M_k, \text{ and } r \in R; \quad (31)$$

$$t_{jkm2}^r \geq t_{jkm1}^r + wt_{jkm}^r + (1 - g_{jkm}^r)p - M(1 - x_{ijkm}^{r', r}), \quad \forall i \in V^0, j \in V^{N+1}, k \in L, m \in M_k, \text{ and } r, r' \in R; \quad (32)$$

$$t_{jkm1}^r \geq t_{ikm2}^{r'} + \left[\frac{Q_i^{r'}}{v_d} + \frac{d_{ij}}{v_{\text{vehicle}}} \right] - M(1 - x_{ijkm}^{r', r}), \quad \forall i \in V^0, j \in V^{N+1}, k \in L, m \in M_k, \text{ and } r, r' \in R; \quad (33)$$

$$t_{ikm1}^r \leq T, \quad \forall i \in V, k \in L, m \in M_k, \text{ and } r \in R. \quad (34)$$

5. SOLUTION METHOD

As a variant of the VRP, MMHVRPTW is NP-hard. For small-scale instances, the optimal solution can be found by using exact approaches such as the branch-and-price method [27]. However, for large-scale instances, finding a superior solution within an acceptable computing time is usually impossible. Therefore, we design a meta-heuristic method to obtain a satisfactory solution to MMHVRPTW within a reasonable computing time.

VNS is an improved local search algorithm [6]. Employing the idea of changing the neighborhood, it improves the current solution from the local optima [16]. VNS includes two procedures, the local search, and shaking. The typical algorithm of the local search procedure is Variable Neighborhood Descent (VND), including the local search process and neighborhood change process. When the algorithm finds an incumbent solution locally optimal, it escapes from the local optimum by changing the neighborhood. In the shaking procedure, the algorithm changes the current solution by shaking the neighborhood, to avoid being trapped in a local area and to increase the chance of finding the global optimum.

Since proposed by Mladenović and Hansen [29], the VNS method has been widely used to solve various VRPs [4, 18]. For example, Rezgui *et al.* [35] develop a VNS for a pickup and delivery problem with electric modular vehicles. In [37]'s multi-depot green VRP, a hybrid VNS-and-Tabu-search method with some new problem-specific neighborhood structures is developed. Kyriakakis *et al.* [22] integrate the VNS with ant colony optimization to solve the cumulative capacitated VRP. Jabir *et al.* [20] develop a hybrid ant-colony VNS meta-heuristic for the multi-depot green VRP.

This paper develops a hybrid VNS algorithm (HVNS) to solve MMHVRPTW. To further improve the global search ability, a Simulated Annealing-based acceptance criterion is embedded into VNS, so that the algorithm can accept worse solutions with a certain probability. The general idea of HVNS is to iteratively improve the current solution (starting from an initial solution) to find the optimal one. The detailed steps are:

- (1) Generate an initial solution x .
- (2) Perform the shaking process on x : from the operator set $\{N_r | r = 1, 2, \dots, r_{\max}\}$, select an operator N_r and apply it on x , $N_r(x)$, to obtain a new solution x' .
- (3) Apply the VND local search throughout the neighborhood set of x' , $\{M_k(x') | k = 1, 2, \dots, k_{\max}\}$.
- (4) Once a better solution x'' is found, or a worse one is accepted by the Simulated Annealing-based acceptance criterion, use x'' to replace x' and restart step 3. Otherwise, move on to the next neighborhood in the neighborhood set.



FIGURE 1. Illustration of the Relocate operator (N_1).



FIGURE 2. Illustration of the Exchange operator (N_2).



FIGURE 3. Illustration of the 2-opt operator (N_3).

- (5) When the last neighborhood $M_{k_{\max}}(x')$ is reached and there is no better solution, use the current solution x' to replace x .
- (6) Select another operator N_r from the operator set $\{N_r | r = 1, 2, \dots, r_{\max}\}$ and repeat the entire process, until a pre-set maximum number of iterations, $Iter_{\max}$, is reached.

Generating an initial solution. We use a greedy algorithm to generate an initial solution that satisfies all constraints in our model. First, if there are any internal vehicles available, then randomly select one; if all internal vehicles are used, then randomly select an external vehicle. Considering the time window constraint and vehicle capacity constraint, generate a set of available orders for the vehicle. Each order indicates the demand for a product at a station, corresponding to a tank storing the product at the station. Therefore, the set of orders determines a set of tanks (at their corresponding stations). Then, select the nearest tank from the set of available orders to visit. Note that the distance between tanks at the same station is zero. Keep selecting the next nearest tank to visit until no tank can satisfy the constraints. If there are still orders unfulfilled, then repeat the process for the vehicle to construct another trip. If the vehicle is not available to make more trips, then select another vehicle to construct trips, until all orders are fulfilled.

Shaking. The shaking procedure in HVNS is to disturb the current solution, facilitating the algorithm to improve the local optima, and enhancing the global search capability. Three well-known neighborhood operators can be found in the literature: Relocate (N_1), Exchange (N_2), and 2-opt (N_3). All three operators obtain new neighborhoods by adjusting the sequence of tanks visited by a vehicle on a trip.

N_1 : Relocate. Randomly select a vehicle and a trip (of the vehicle). From the list of orders on the trip, randomly select an order and relocate the order to a new location. For example, order 7 is relocated from between orders 6 and 8 to between orders 4 and 5, as shown in Figure 1.

N_2 : Exchange. Randomly select a vehicle and a trip. Randomly select two orders from the trip and exchange their locations. Figure 2 shows an example of the operator, which exchanges the locations of orders 3 and 6.

N_3 : 2-opt. Randomly select two non-adjacent orders from a trip. Reverse all orders between them. Figure 3 shows an example of reversing the orders between 3 and 6.

VND local searching. After the shaking procedure, we apply the VND algorithm to improve the current solution by local search. The idea is to search through the neighborhood set $\{M_k | k = 1, 2, \dots, k_{\max}\}$ and apply them to the current solution. If a better solution is found, or a worse one is accepted by the Simulated Annealing-based acceptance criterion, then replace the current solution and restart the search from M_1 . If no

better solution is found after searching $M_{k_{\max}}$, then use the current solution either as the best solution to start another shaking procedure or as the optimal solution.

Based on present neighborhood structures for VRP [1, 19, 32], this paper designs eight structures to solve MMRVRPTW. The first three structures are obtained by directly applying the three neighborhood operators: $N_1 \rightarrow M_1$, $N_2 \rightarrow M_2$, and $N_3 \rightarrow M_3$. The other five are as follows.

N_4 : Relocate Trip. Select a trip of a vehicle and insert it into the tour (set of trips) of another vehicle. All selections are random.

N_5 : Partial Exchange. Select two trips; they can be of the same vehicle or different vehicles. Select one segment (including several consecutive orders) from each trip and exchange the two segments. All selections are random.

N_6 : Station Shift. Select a petrol station and a vehicle. Remove all orders received by the station from their current trips and insert the orders into the tour of the vehicle. Since more frequent visits incur higher costs, this neighborhood can effectively reduce visits to the station, lowering the preparation cost. All selections are random.

N_7 : Remove Vehicle. Randomly select a vehicle and remove all of its trips, resulting in an incomplete solution. Then, randomly select an operator from the insertion operator set to reinsert all orders on those trips into that incomplete solution, in order to obtain a feasible solution. The set consists of three insertion operators in the ALNS literature, including Greedy Insertion [38], Greedy Insertion Perturbation [36], and Regret Insertion [36]. This neighborhood can quickly reduce the number of vehicles used. However, there may not exist any feasible complete solution, in which case our algorithm moves to another neighborhood.

N_8 : Shift Vehicle. Randomly select a vehicle. Insert all of its trips into the tour of an idle but smaller vehicle that can still fulfill all orders on those trips.

Acceptance criterion. To improve the global search capability of VNS, the acceptance criterion based on Simulated Annealing is used to accept or reject a solution. A new solution, x_{new} , is always accepted if it is better than the current solution, x_{current} : $\text{cost}(x_{\text{new}}) < \text{cost}(x_{\text{current}})$. If $\text{cost}(x_{\text{new}}) \geq \text{cost}(x_{\text{current}})$, x_{new} is accepted with probability $p = e^{-\frac{\text{cost}(x_{\text{new}}) - \text{cost}(x_{\text{current}})}{\tilde{T}}}$. \tilde{T} is the current temperature. \tilde{T} is initially set to $T_0 = \text{cost}(x_{\text{initial}})\mu$ where x_{initial} is the initial solution and μ is the initial temperature parameter. For each iteration, \tilde{T} decreases at a temperature cooling rate δ : $\tilde{T} = \delta\tilde{T}$, $0 < \delta < 1$.

6. EXPERIMENTAL STUDIES

Our experimental studies are performed on (i) small-scale instances, (ii) benchmark instances, and (iii) a practical application. The small-scale instances are generated from the real-life data of PetroChina, including two to five petrol stations with four to ten orders. The benchmark instances are generated from Solomon's benchmark instance sets. The number of petrol stations varies from 25 to 50 and then to 100, and the number of orders increases from 75 to 150 and then to around 200. The practical application is the distribution network of PetroChina, including 73 petrol stations and 177 orders. To verify the effectiveness of our approach, we conduct four experiments. In the first experiment, we compare the CPLEX method and HVNS on the small-scale instances. In the second and third experiments, HVNS is compared with VNS and other two ALNS heuristics for different-scale benchmark instances. Finally, HVNS is applied to a real-world instance.

In our presented heuristics, HVNS, three parameters need to be tuned: the maximum number of iterations, $Iter_{\max}$, the initial temperature parameter, μ , and the temperature cooling rate, δ . Parameter-tuning experiments are conducted using six benchmark instances, each with 100 petrol stations. Details of those experiments are introduced in Section A.1. Results from those experiments indicate that it is reasonable and locally optimal to set $Iter_{\max} = 10$, $\mu = 100$, and $\delta = 0.999$.

All algorithms are coded in Python 3.7 and are performed on a computer with Core i7-3.4 GHz CPU, 16 GB RAM, and Windows 7 (64-bit) operation system.

6.1. Small-scale instances

Based on the real-life data of PetroChina, we generate a small-scale instance set, including 28 small-scale instances. Each instance is named in the format of $Sabc-d$. $a \in \{2, 3, 4, 5\}$ represents the number of petrol stations. $b \in \{2, 3\}$ is the maximum number of product types at a station. $c \in \{1, 2, 3, 4\}$ describes the operation period length, the use of internal and external vehicles, and whether different products ordered by a station have the same time window. They may be delivered on one trip together if they have the same time window. If they have different time windows, they must be delivered on different trips. When $c = 1$ or 2 , the operation period is long and internal vehicles are enough. When $c = 3$ or 4 , the operation period is short and internal vehicles are not enough. Products demanded by a station have the same time window for $c = 1$ or 3 . They have different time windows for $c = 2$ or 4 . d is the number of orders included in the instance.

To analyze the performance of our HVNS heuristics, we first use the commercial solver CPLEX 12.2 to solve MMHVRPTW on the generated instances. The running time limit is set at 7200 s. We show the optimal solution (if found within the time limit) together with the corresponding running time, or the best solution found within the time limit (if the optimal solution was not found), the numbers of total vehicles, external vehicles, and total trips in Table 1. When applying HVNS for each instance, we conduct ten experiments and select the best one (with the lowest cost). Comparing the HVNS result with the optimal or best solution obtained by CPLEX, the reduced cost percentage is shown in column *gap (%)*.

Results show that HVNS can solve all small-scale MMHVRPTW instances very efficiently. For S433-10, the best solution found by CPLEX within the time limit is 5708.98 RMB and the solution obtained by HVNS is 5673.46 RMB, improved (reduced) by 0.62%. For all other 27 instances, CPLEX and HVNS can find the same optimal solutions, but HVNS takes a significantly shorter time: the average running time of CPLEX is 486.26 s and it is only 0.27 s for HVNS.

The CPLEX results can also support our instance setting. First, for the Sab3-d and Sab4-d instances, external vehicles are widely used because of the lack of internal vehicles. Second, for instances with long operation periods, *e.g.*, S421-8, S431-10, and S521-10, only one internal vehicle is used, so it has to make multiple trips. Third, for instances that assume that different products have different time windows at a station, separate trips are made to deliver different products to the station. In particular, we use Gantt charts to show details of the solution routes for the S52c-10 instances, as shown in Figure 4. For S521-10, internal vehicle #3 makes two trips. On the first trip, it visits stations #5 and #3; on the second trip, it visits stations #4, #2, and #1. At station #5, it discharges 18000 liters of product #1 and 3000 liters of product #2. At station #3, it discharges products #1 (19000 liters) and #2 (5000 liters). At station #4, it discharges products #2 (18000 liters) and #1 (4000 liters). At station #2, it discharges products #2 (18000 liters) and #1 (4000 liters). At station #1, it discharges products #1 (19000 liters) and #2 (4000 liters). At stations #5, #3, #4, and #1, the discharge of the second product starts right after the vehicle finishes discharging the first product. At station #2, after discharging product #2, because the accommodating time for product #1 has not been reached yet, the vehicle must wait at the station until the accommodating time. The details of instances S522-10, S523-10, and S524-10 are shown in Figures 4a, 4b, 4c, and 4d, respectively.

6.2. Benchmark instances

Instance generation. We use Solomon [39]'s benchmark instances for the VRP with time windows to generate our benchmark instances for MMHVRPTW. To adapt Solomon's instances to MMHVRPTW, we use four adaptation strategies, S1, S2, S3, and S4, to divide the product demand and compartment capacity, and to differentiate the vehicle type. S1 is proposed by El Fallahi *et al.* [14], assuming that there are three products in MMHVRPTW. It evenly divides the demand of each station into three parts while keeping the time windows unchanged. It also divides each vehicle into three compartments, so vehicles are homogeneous. Based on [14], we propose three more adapting strategies, S2, S3, and S4. S2 divides everything unevenly. Assume that q_i is the total demand of station i . The demand for product #1 is attributed to be $q_i^1 = \frac{q_i}{k_1}$, where k_1 is randomly selected from $\{3, 4, 5\}$. The demand of product #2 is $q_i^2 = \frac{q_i - q_i^1}{k_2}$, where $k_2 \in 2, 3, 4$ is random. The demand for

TABLE 1. Experiments on small-scale instances.

Instance	CPLEX					HVNS		
	Cost (RMB)	Time (s)	Total vehicles	External vehicles	Total trips	Cost (RMB)	Time (s)	Gap (%)
S221-4	2374.92	0.792	2	0	2	2374.92	0.034	0
S231-6	5016.84	5.912	4	0	4	5016.84	0.329	0
S321-6	3130.11	1.11	2	0	2	3130.11	0.089	0
S331-9	5771.5	1044.229	4	0	4	5771.5	0.209	0
S421-8	3542.4	3.439	1	0	2	3542.4	0.081	0
S431-10	3723.2	114.105	1	0	2	3723.2	0.222	0
S521-10	4292.29	38.579	1	0	2	4292.29	0.138	0
S222-4	2374.92	0.434	2	0	2	2374.92	0.041	0
S232-6	3732.84	5.255	3	0	3	3732.84	0.173	0
S322-6	4491.45	3.191	3	0	3	4491.45	0.048	0
S332-9	3205.27	186.558	2	0	2	3205.27	0.183	0
S422-8	3855.24	9.112	2	0	2	3855.24	0.072	0
S432-10	3855.24	470.773	2	0	2	3855.24	0.211	0
S522-10	4604.96	622.367	2	0	2	4604.96	0.547	0
S223-4	2177.57	0.49	1	0	1	2177.57	0.082	0
S233-6	5746.57	1.371	3	1	3	5746.57	0.243	0
S323-6	7713.57	2.048	4	1	4	7713.57	1.359	0
S333-9	11 461.97	992.717	6	4	6	11 461.97	0.279	0
S423-8	5475.79	3.744	2	1	2	5475.79	0.087	0
S433-10	5708.98	7200	3	2	3	5673.46	0.875	-0.62
S523-10	6225.26	2393.208	2	1	2	6225.26	0.118	0
S224-4	3208.92	0.442	2	1	2	3208.92	0.074	0
S234-6	3208.92	1.166	2	1	2	3208.92	0.069	0
S324-6	4069.59	1.311	2	1	2	4069.59	0.231	0
S334-9	4069.56	21.718	2	1	2	4069.56	0.424	0
S424-8	4715.92	6.631	2	1	2	4715.92	0.277	0
S434-10	4722.64	243.968	2	1	2	4722.64	0.845	0
S524-10	8334.49	209.627	4	3	4	8334.49	0.29	0
Average	-	486.26	-	-	-	-	0.27	-

product #3 is $q_i^3 = q_i - q_i^1 - q_i^2$. The time windows remain unchanged. Vehicles are divided into three types, so they are heterogeneous, with two compartments, three compartments, and four compartments, respectively. S3 performs proportional divisions. The demands of products #1, #2, and #3 are $\frac{2}{9}$, $\frac{1}{3}$, and $\frac{4}{9}$ of the total demand, respectively. Vehicles are heterogeneous – there are three types of vehicles, with a capacity ratio of 1:1.5:2.25. All vehicles have three compartments. Adopting S4, vehicles are divided in the same way as S3. The demand of a station is randomly divided into one product, two products with equal demands, and three products with equal demands. Another difference between S3 and S4 is the instance scale: S4 is specifically for large-scale instances.

We employ three sets of Solomon's benchmark instances, with 25, 50, and 100 petrol stations included in each instance, respectively. From the first Solomon set, we randomly select 19 instances and apply strategies S1, S2, or S3 on them, obtaining a set of 19 MMHVRPTW instances, set I. Each set-I instance has 25 petrol stations and 75 orders. 18 Solomon instances are selected from the second set and are processed by strategies S1, S2, or S3, resulting in set II, a set of 18 MMHVRPTW instances, each with 50 stations and 150 orders. Set III is constructed based on the third Solomon set, adapted by strategy S4, including 17 MMHVRPTW instances, each with 100 stations and around 200 orders.

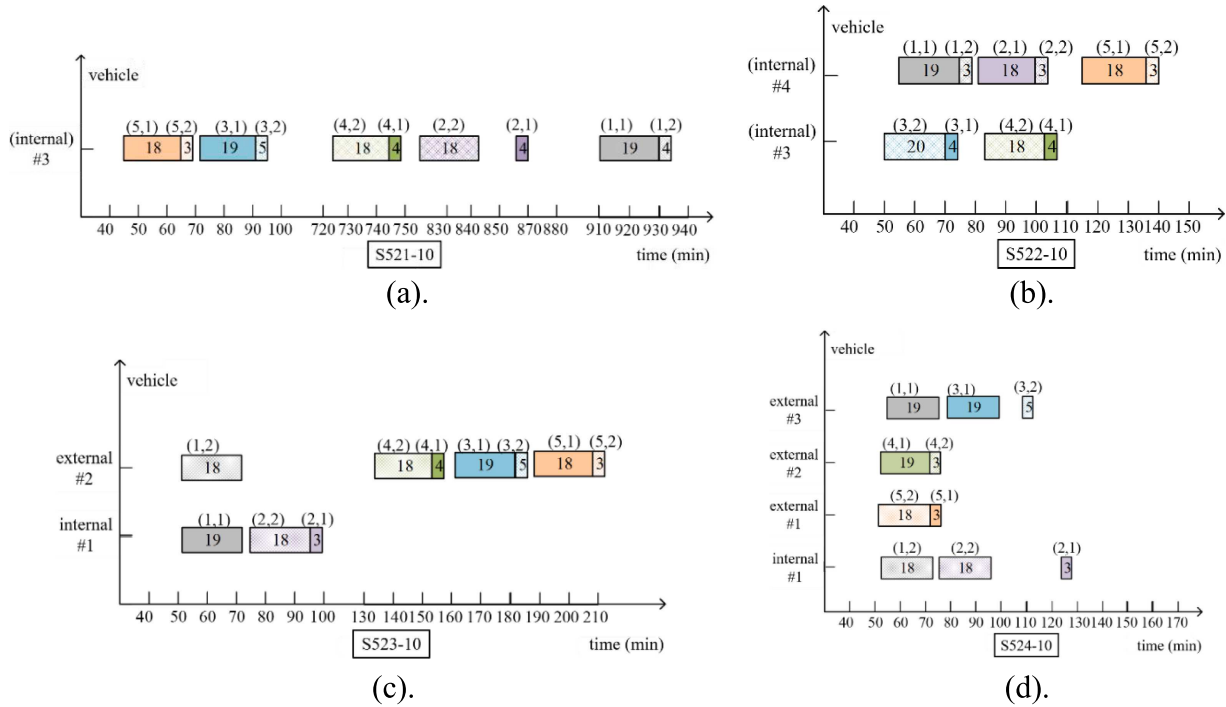


FIGURE 4. Gantt charts of small-scale instances. (a) Gantt chart of S521-10. (b) Gantt chart of S522-10. (c) Gantt chart of S523-10. (d) Gantt chart of S524-10.

Methodology application. According to our experiment, CPLEX is only capable of solving MMHVRPTW problems of very small scales. When there are more than ten orders, CPLEX is not capable of finding superior solutions within our time limit, 7200s. As all adapted benchmark instances have at least 75 orders, CPLEX is not used to test the performance of the HVNS.

Methods employed for the comparison with HVNS include the original VNS, ALNS, and enhanced ALNS (EALNS). To examine the impact of the Simulated Annealing-based acceptance criterion and the Remove Vehicle neighborhood, we employ the original VNS without using the acceptance criteria and the neighborhood.

To further verify the performance of the HVNS, we use the ALNS introduced by Wang *et al.* [40] and the EALNS introduced by Eshtehadi *et al.* [15]. The EALNS can be directly applied on MMHVRPTW; parameter values given by Eshtehadi *et al.* [15] are also used in this study. The ALNS needs to be adapted. Since deliveries can be split in [40] but cannot be split in the MMHVRPTW, when applying the ALNS, we use only the destroy and partial repair operators with the integral insertion and split-trip insertion strategies employed by Wang *et al.* [40]. Such adaptation limits the selection of operators and insertion strategies but does not change elements in those two sets. Therefore, we use the same parameter values given by Wang *et al.* [40] in this study. Section A.1 introduces detailed parameter information of those two methods.

For each benchmark instance, VNS, HVNS, ALNS, and EALNS are performed for ten runs, respectively, and HVNS, ALNS, and EALNS use the same initial solution. As the average HVNS computing times with set I, II, and III instances are 126.82, 452.15, and 659.51 s, respectively, we set the running times for ALNS and EALNS to be 126.82 (452.15, and 659.51) seconds for set-I (II, and III) instances.

Result analysis. Applying those four methods on the 54 instances (19 instances in set I, 18 in set II, and 17 in set III), we show the detailed simulation results in Section A.2. For each instance and methodology pair, the solution (cost) shown in Table A.4 is the minimum cost from the ten runs.

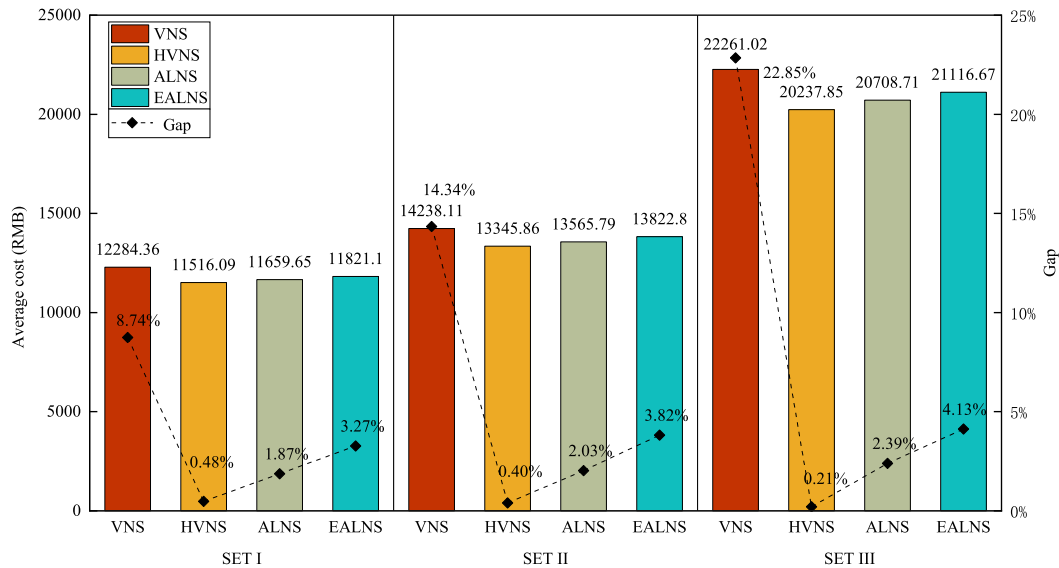


FIGURE 5. Experimental results of the different methods.

For analysis, we first calculate the average cost for each set using each method, represented by the bars in Figure 5. It clearly shows the performance advantage of the HVNS.

For each instance, we find the best solution (minimum cost) of the four methods and then, from this best solution, we calculate the percentage deviation of the cost obtained by each method. The average percentage deviation for each set using each method is represented by the dots and dotted lines in Figure 5. It also provides a comparison of the performance of the four methods.

For each instance set and for the entire instance group, we conduct a series of hypothesis tests to compare performances of HVNS with the other three methods. The proposed hypotheses are:

- H1:** HVNS incurs a lower cost than VNS.
- H2:** HVNS incurs a lower cost than ALNS.
- H3:** HVNS incurs a lower cost than EALNS.

To test these hypotheses, we select the paired student T test for two reasons. First, different methods are applied on the same instance sets: set I (with 19 instances that have 25 petrol stations), set II (with 18 instances that have 50 petrol stations), set III (with 17 instances that have 100 petrol stations), and the set with all 54 generated instances. Second, for each set, the sample size is small: 19, 18, 17, and 54, respectively.

Table 2 shows details of those tests. The p -values of testing H1 are 0.000456, 0.001564, $7.052e-05$, and $1e-08$, all smaller than a general significance level such as $\alpha = 0.005$. Therefore, the null hypothesis should be rejected in all cases and H1 is supported. For similar reasons, H2 and H3 are also supported.

6.3. Practical application

We use the distribution network of PetroChina as a practical application, to verify the practicability of our study. The operation period is 12 h, during which the company needs to deliver three oil products from a depot to 73 petrol stations in the city. Three types of internal vehicles and one type of external vehicle are available for fulfilling 177 orders, with a total quantity of 1 432 700 liters. More detailed data such as the geographical distribution of the warehouse and gasoline stations, and information on some orders and the fleet are shown in Section A.3.

TABLE 2. Hypothesis test on benchmark instances.

Hypothesis	Sample Sample size df	Set I 19 18	Set II 18 17	Set III 17 16	All instances 54 53
H1					
$H_0 : \text{Cost}_{\text{HVNS}} \geq \text{Cost}_{\text{VNS}}$	Test statistics	-3.963	-3.4397	-4.9629	-6.5979
$H_a : \text{Cost}_{\text{HVNS}} < \text{Cost}_{\text{VNS}}$	<i>p</i> -value	0.000456	0.001564	7.052e-05	1e-08
H2					
$H_0 : \text{Cost}_{\text{HVNS}} \geq \text{Cost}_{\text{ALNS}}$	Test statistics	-6.6996	-4.1555	-3.7826	-5.8155
$H_a : \text{Cost}_{\text{HVNS}} < \text{Cost}_{\text{ALNS}}$	<i>p</i> -value	1.389e-06	0.0003311	0.0008158	1.777e-07
H3					
$H_0 : \text{Cost}_{\text{HVNS}} \geq \text{Cost}_{\text{EALNS}}$	Test statistics	-6.5657	-4.4753	-4.2858	-6.7131
$H_a : \text{Cost}_{\text{HVNS}} < \text{Cost}_{\text{EALNS}}$	<i>p</i> -value	1.807e-06	0.0001665	0.0002836	6.527e-09

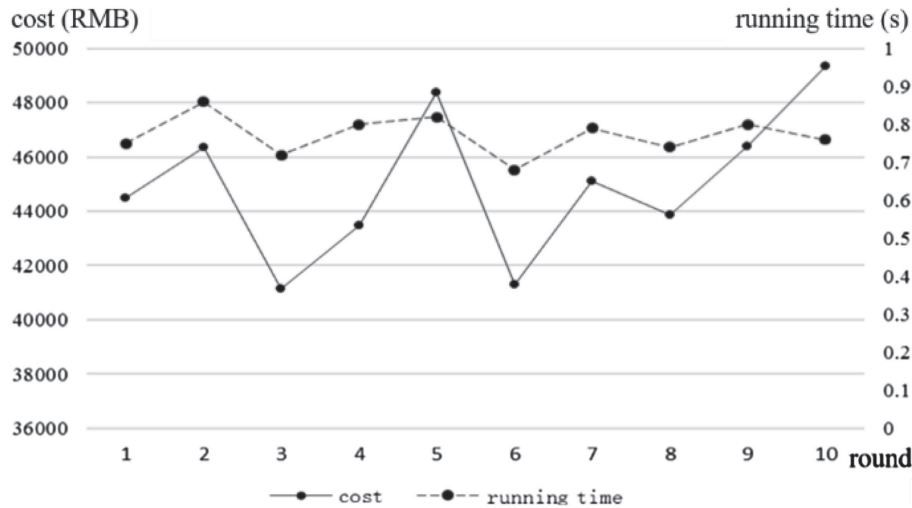


FIGURE 6. Generating the initial solution for ten rounds.

We run the initial solution code for ten rounds. Results are shown in Figure 6. For all ten rounds, the initial solution can be generated within one second, incurring a cost between 41 134.44 RMB and 49 332.04 RMB.

We take the initial solution with the minimum cost, 41 134.44 RMB, and that with the maximum cost, 49 332.04 RMB, to apply the HVNS, respectively. The results are shown in Figure 7. Detailed data are given in the Appendix A. In both cases, the optimization algorithm can optimize the initial solution to a large extent. When the initial cost is 41 134.44 RMB, the best solution is 20 618.1 RMB, decreased by 49.88%. When the initial cost is 49 332.04 RMB, the best solution is 19 979.61 RMB, which is decreased by 59.50%. Comparing the two cases, a common prediction is that the case with a lower initial cost can reach a lower optimal cost. However, according to our experiment, the optimization effect is more obvious when the initial cost is higher, contrary to the common prediction. However, the running time is also longer when the initial cost is higher, in general. Section A.4 shows details of the ten runs starting with different initial solutions.

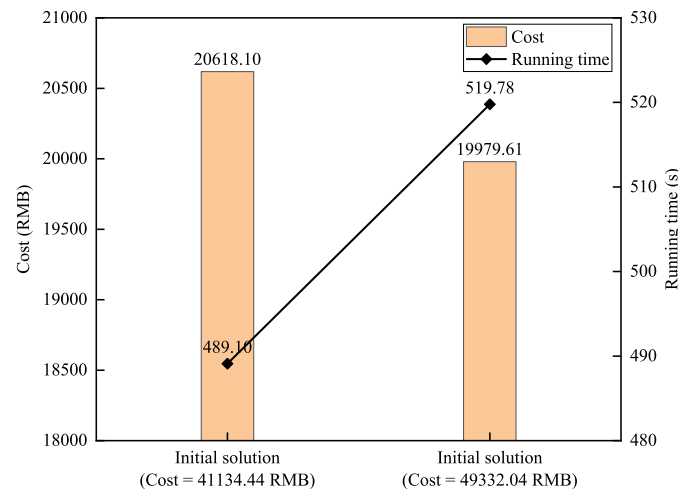


FIGURE 7. Comparison between different initial solutions.

7. CONCLUSION

This paper studies a petrol station replenishment problem using heterogeneous multi-compartment vehicles, each making multiple trips to deliver several products to petrol stations. Based on the practical needs of PetroChina, the paper captures a lot of practical needs and constraints of the PSRP, such as a hard time window for delivering each product at each station, the limited capacity of each tank at a station, the necessary preparation time and cost generated because of static elimination, etc. We establish a mathematical model for the problem, referred to as MMHVRPTW, to minimize the total cost. The model is solved by HVNS with eight neighborhood structures. A Simulated Annealing-based acceptance criterion is embedded in VNS. We design the Remove Vehicle neighborhood with three insertion operators to improve the performance of the heuristic. The proposed model and algorithm are validated *via* various experiments, including small-scale instances generated from real-life data, benchmark instances, and a practical application of PetroChina. We show that HVNS is superior to CPLEX and the other two ALNS methods in terms of solution quality. We also compare HVNS with VNS to verify the efficiency of the acceptance criterion and the Remove Vehicle neighborhood.

However, some problems in real-life PSRP are still not captured by this paper yet. For example, we consider only one depot with sufficient inventory of all products. In some cities, there are multiple oil depots with limited inventory. Stockout in real life is also allowed, although it incurs a significant cost. Petrol stations prefer vehicles to arrive and discharge oil products during off-peak hours. In addition, some enterprises adopt Vendor Managed Inventory (VMI), and suppliers need to make decisions on both delivery quantity and vehicle scheduling. Based on those limitations, the following research directions are considered in our future studies: (i) a distribution problem with multiple depots, soft time windows, and off-peak-hour delivery; (ii) an inventory routing problem for the joint optimization of delivery quantity and vehicle scheduling.

DATA AVAILABILITY STATEMENT

No new data/codes were created or analyzed in this study.

REFERENCES

- [1] M. Alinaghian and N. Shokouhi, Multi-depot multi-compartment vehicle routing problem, solved by a hybrid adaptive large neighborhood search. *Omega* **76** (2018) 85–99.
- [2] P. Avella, M. Boccia and A. Sforza, Solving a fuel delivery problem by heuristic and exact approaches. *Eur. J. Oper. Res.* **152** (2004) 170–179.
- [3] F.F. Boctor, J. Renaud and F. Cornillier, Trip packing in petrol stations replenishment. *Omega* **39** (2011) 86–98.

- [4] O. Bräysy, A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS J. Comput.* **15** (2003) 347–368.
- [5] E.D. Chajakis and M. Guignard, Scheduling deliveries in vehicles with multiple compartments. *J. Global Optim.* **26** (2003) 43–78.
- [6] B. Chen, R. Qu, R. Bai and W. Laesanklang, A variable neighborhood search algorithm with reinforcement learning for a real-life periodic vehicle routing problem with time windows and open routes. *RAIRO-Oper. Res.* **54** (2020) 1467–1494.
- [7] J. Chen, B. Dan and J. Shi, A variable neighborhood search approach for the multi-compartment vehicle routing problem with time windows considering carbon emission. *J. Clean. Prod.* **277** (2020) 123932.
- [8] L.C. Coelho and G. Laporte, Classification, models and exact algorithms for multi-compartment delivery problems. *Eur. J. Oper. Res.* **242** (2015) 854–864.
- [9] F. Cornillier, F.F. Boctor, G. Laporte and J. Renaud, An exact algorithm for the petrol station replenishment problem. *J. Oper. Res. Soc.* **59** (2008) 607–615.
- [10] F. Cornillier, F.F. Boctor, G. Laporte and J. Renaud, A heuristic for the multi-period petrol station replenishment problem. *Eur. J. Oper. Res.* **191** (2008) 295–305.
- [11] F. Cornillier, G. Laporte, F.F. Boctor and J. Renaud, The petrol station replenishment problem with time windows. *Comput. Oper. Res.* **36** (2009) 919–935.
- [12] F. Cornillier, F. Boctor and J. Renaud, Heuristics for the multi-depot petrol station replenishment problem with time windows. *Eur. J. Oper. Res.* **220** (2012) 361–369.
- [13] E. Demir, T. Bektaş and G. Laporte, An adaptive large neighborhood search heuristic for the pollution-routing problem. *Eur. J. Oper. Res.* **223** (2012) 346–359.
- [14] A. El Fallahi, C. Prins and R.W. Calvo, A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. *Comput. Oper. Res.* **35** (2008) 1725–1741.
- [15] R. Eshtehadi, E. Demir and Y. Huang, Solving the vehicle routing problem with multi-compartment vehicles for city logistics. *Comput. Oper. Res.* **115** (2020) 104859.
- [16] P. Hansen, N. Mladenovic and J.A. Moreno Perez, Variable neighborhood search. *Eur. J. Oper. Res.* **191** (2008) 593–770.
- [17] K. Heßler, Exact algorithms for the multi-compartment vehicle routing problem with flexible compartment sizes. *Eur. J. Oper. Res.* **294** (2021) 188–205.
- [18] V.C. Hemmelmayr, K.F. Doerner and R.F. Hartl, A variable neighborhood search heuristic for periodic routing problems. *Eur. J. Oper. Res.* **195** (2009) 791–802.
- [19] T. Henke, M.G. Speranza and G. Wäscher, The multi-compartment vehicle routing problem with flexible compartment sizes. *Eur. J. Oper. Res.* **246** (2015) 730–743.
- [20] E. Jabir, V.V. Panicker and R. Sridharan, Design and development of a hybrid ant colony-variable neighbourhood search algorithm for a multi-depot green vehicle routing problem. *Transp. Res. Part D: Transp. Environ.* **57** (2017) 422–457.
- [21] M. Keskin and B. Çatay, Partial recharge strategies for the electric vehicle routing problem with time windows. *Transp. Res. Part C: Emerging Technol.* **65** (2016) 111–127.
- [22] N.A. Kyriakakis, M. Marinaki and Y. Marinakis, A hybrid ant colony optimization-variable neighborhood descent approach for the cumulative capacitated vehicle routing problem. *Comput. Oper. Res.* **134** (2021) 105397.
- [23] R. Lahyani, L.C. Coelho, M. Khemakhem, G. Laporte and F. Semet, A multi-compartment vehicle routing problem arising in the collection of olive oil in Tunisia. *Omega* **51** (2015) 1–10.
- [24] Z. Liu and X. Zuo, Inventory routing problem with split delivery and variable time windows for customers with small capacity and large sales. *IEEE Trans. Intell. Transp. Syst.* **25** (2024) 10375–10388.
- [25] Z. Liu, Y. Li, J. Xu and D. Bai, Multi-compartment electric vehicle routing problem for perishable products. *Int. J. Crowd Sci.* **8** (2024) 38–48.
- [26] Z. Liu, L. Sun, X. Zuo and H. Li, Heterogeneous electric vehicle routing problem with multiple compartments and multiple trips for the collection of classified waste. *Int. J. Crowd Sci.* **8** (2024) 130–139.
- [27] H. Luo, M. Dridi and O. Grunder, A branch-and-price algorithm for a routing and scheduling problem from economic and environmental perspectives. *RAIRO-Oper. Res.* **56** (2022) 3267–3292.
- [28] J.E. Mendoza, B. Castanier, C. Guéret, A.L. Medaglia and N. Velasco, Constructive heuristics for the multicompartment vehicle routing problem with stochastic demands. *Transp. Sci.* **45** (2011) 346–363.
- [29] N. Mladenović and P. Hansen, Variable neighborhood search. *Comput. Oper. Res.* **24** (1997) 1097–1100.
- [30] M. Ostermeier and A. Hübner, Vehicle selection for a multi-compartment vehicle routing problem. *Eur. J. Oper. Res.* **269** (2018) 682–694.

- [31] M. Ostermeier, T. Henke, A. Hübner and G. Wäscher, Multi-compartment vehicle routing problems: state-of-the-art, modeling framework and future directions. *Eur. J. Oper. Res.* **292** (2021) 799–817.
- [32] D. Popović, M. Vidović and G. Radivojević, Variable neighborhood search heuristic for the inventory routing problem in fuel delivery. *Expert Syst. App.* **39** (2012) 13390–13398.
- [33] M. Rabbani, H. Farrokhi-Asl and H. Rafiei, A hybrid genetic algorithm for waste collection problem by heterogeneous fleet of vehicles with multiple separated compartments. *J. Intell. Fuzzy Syst.* **30** (2016) 1817–1830.
- [34] M. Reed, A. Yiannakou and R. Evering, An ant colony algorithm for the multi-compartment vehicle routing problem. *Appl. Soft Comput.* **15** (2014) 169–176.
- [35] D. Rezgui, J.C. Siala, W. Aggoune-Mtalaa and H. Bouziri, Application of a variable neighborhood search algorithm to a fleet size and mix vehicle routing problem with electric modular vehicles. *Comput. Ind. Eng.* **130** (2019) 537–550.
- [36] S. Ropke and D. Pisinger, An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transp. Sci.* **40** (2006) 455–472.
- [37] M.E.H. Sadati and B. Çatay, A hybrid variable neighborhood search approach for the multi-depot green vehicle routing problem. *Transp. Res. Part E: Logistics Transp. Rev.* **149** (2021) 102293.
- [38] P. Shaw, Using constraint programming and local search methods to solve vehicle routing problems, in *International Conference on Principles and Practice of Constraint Programming*. Springer (1998) 417–431.
- [39] M.M. Solomon, Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* **35** (1987) 254–265.
- [40] L. Wang, J. Kinable and T. Van Woensel, The fuel replenishment problem: a split-delivery multi-compartment vehicle routing problem with multiple trips. *Comput. Oper. Res.* **118** (2020) 104904.
- [41] X.-T. Wei, Q. Liao, H.-R. Zhang, Y.-T. Liang, B.-H. Wang, N. Xu and M. Yuan, MILP formulations for highway petrol station replenishment in initiative distribution mode. *Petroleum Sci.* **18** (2021) 994–1010.
- [42] H. Yahyaoui, I. Kaabachi, S. Krichen and A. Dekdouk, Two metaheuristic approaches for solving the multi-compartment vehicle routing problem. *Oper. Res.* **20** (2020) 2085–2108.

Please help to maintain this journal in open access!



This journal is currently published in open access under the Subscribe to Open model (S2O). We are thankful to our subscribers and supporters for making it possible to publish this journal in open access in the current year, free of charge for authors and readers.

Check with your library that it subscribes to the journal, or consider making a personal donation to the S2O programme by contacting subscribers@edpsciences.org.

More information, including a list of supporters and financial transparency reports, is available at <https://edpsciences.org/en/subscribe-to-open-s2o>.

APPENDIX A.

A.1. Parameter tuning and setting

HVNS has three parameters that need to be tuned: the maximum number of iterations, $Iter_{max}$, the initial temperature parameter, μ , and the temperature cooling rate, δ . We use the parameter tuning method introduced by Demir *et al.* [13] and Keskin and Çatay [21] to set values of those parameters. Parameter-tuning experiments are conducted using six benchmark instances: C1_100, C3_100, R7_100, R9_100, RC12_100, and RC14_100. Details of those instances are shown in Table A.5.

Table A.1 shows the parameter-tuning result. According to Demir *et al.* [13], we set initial values for the parameters: $Iter_{max} = 8$, $\mu = 100$ and $\delta = 0.999$. We first consider six different values of $Iter_{max}$. For each $Iter_{max}$ value, we perform ten runs of HVNS with each instance and compute the deviation percentage of solutions from the ten runs, as recorded in column *dev (%)*. Column *time (s)* is the average computing time. Comparing both the deviation and time, we set $Iter_{max}$ to 10 as the deviation is very small at 0.21% and the computing time is acceptable. Fixing $Iter_{max}$ at 10, we tune the other two parameters in the same manner. Experimental results show that $\mu = 100$ and $\delta = 0.999$ are the local optimal values for the initial temperature parameter and the temperature cooling rate.

TABLE A.1. Parameter tuning of HVNS.

Parameter	$Iter_{max}$			μ		δ	
	Value	dev (%)	Time (s)	Value	dev (%)	Value	dev (%)
Initial value	8	2.42	503.5	100	0.11	0.999	0.14
Tuned value	6	3.83	385.6	50	0.28	0.9986	0.27
Tuned value	7	3.27	420.3	60	0.23	0.9987	0.18
Tuned value	9	0.68	585.8	70	0.18	0.9988	0.23
Tuned value	10	0.21	638.2	80	0.15	0.9989	0.18
Tuned value	11	0.24	725.7	90	0.15	0.9991	0.16
Tuned value	12	0.20	811.4	110	0.20	0.9992	0.21
Tuned value	13	0.22	907.3	120	0.14	0.9993	0.16

TABLE A.2. Parameter setting of the EALNS

Parameter	Description	Value
r^P	Roulette wheel parameter	0.1
N_w	Number of iterations needed for the roulette wheel mechanism	100
σ_1	Score when a new best solution is found	10
σ_2	Score when a new solution is accepted (better than the current solution)	5
σ_3	Score when a new solution is accepted (worse than the current solution)	1
$P_{initial}$	Starting temperature parameter	100
ch	Cooling rate	0.999
$\bar{\gamma}$	Upper bound for nodes to be removed	$\lfloor \ln(N) \rfloor$
$\underline{\gamma}$	Lower bound for nodes to be removed	$\lfloor \log_{1.4}(N) \rfloor$
ϕ_1	First Shaw parameter	0.5
ϕ_2	Second Shaw parameter	0.25
ϕ_3	Third Shaw parameter	0.15
ϕ_4	Fourth Shaw parameter	0.25
μ	Noise parameter	0.1

As the EALNS (introduced by Eshtehadi *et al.* [15]) is directly applied on MMHVRPTW, we use the same parameter values given by Eshtehadi *et al.* [15] in this study. The EALNS has 14 parameters for MMHVRPTW; they are shown on Table A.2.

We adapt the ALNS (introduced by Wang *et al.* [40]) in MMHVRPTW. However, our adaptation only limits the selection of operators and insertion strategies but does not change elements in those two sets. Therefore, we use the same parameter values given by Wang *et al.* [40] in this study. The ALNS has 11 parameters for MMHVRPTW; they are shown in Table A.3.

A.2. Experimental results using different methods on benchmark instances

For each benchmark instance, VNS, HVNS, ALNS, and EALNS are performed for ten runs, respectively. The minimum costs and the running times of the ten runs are shown in Table A.4. Columns S_a and No indicate the adaptation strategy and the number of orders in the instance. Column *best (RMB)* represents the best solution (minimum cost) found by the four methods. Column *cost (RMB)* is the minimum cost found in the ten runs by a method and *time (s)* is the running time. *gap (%)* shows the percentage deviations of the cost of each method from the best one.

A.3. Details of the practical application

The geographical distribution of the warehouse and gasoline stations in the real-world instance is shown in Figure A.1.

Table A.5 displays the data of some practical orders. Note that for order #0, the depot has zero demand for the virtual non-oil product and an entire time window.

TABLE A.3. Parameter setting of the ALNS.

Parameter	Description	Value
ξ	Reaction factor	0.1
N_{sg}	Number of iterations per segment	100
σ_1	Score when a new best solution is found	5
σ_2	Score when a new solution is accepted (better than the current solution)	3
σ_3	Score when a new solution is accepted (worse than the current solution)	1
T	Initial temperature	200
κ	Cooling rate	0.9995
ρ	Destroy rate for removal operators	0.45
φ	First Shaw parameter	0.2
ψ	Second Shaw parameter	0.25
χ	Third Shaw parameter	0.1

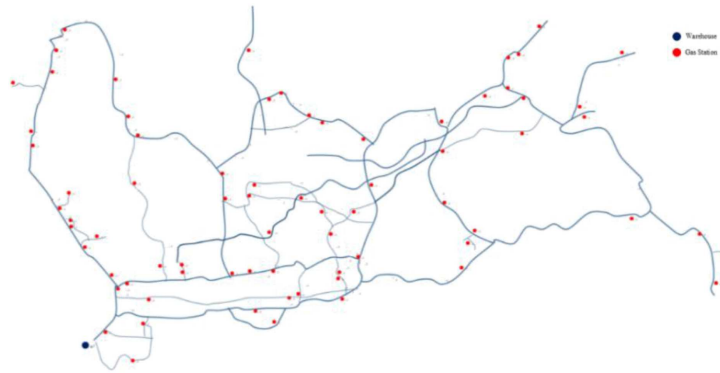


FIGURE A.1. The geographical distribution of gasoline stations and warehouse.

Table A.6 shows the information of the fleet.

A.4. Comparison between different initial solutions

Table A.7 shows details of the ten runs starting with different initial solutions.

TABLE A.5. Information on some orders.

Order ID	Petrol station ID	Product ID	Demand (liters)	Time window (minutes)
0	0	0	0	[0, 720]
1	1	2	3000	[15.38, 686.62]
2	1	1	14 500	[15.38, 675.12]
3	2	1	10 500	[570, 666.19]
4	2	3	13 000	[94, 663.69]
5	3	1	5500	[417, 677.03]
6	3	3	10 000	[22.47, 623]
7	4	2	11 000	[353, 417]
8	4	1	2500	[135, 679.03]
9	4	3	11 000	[23.47, 670.53]
10	5	1	7000	[57, 653.16]
11	5	3	21 000	[108, 639.16]

TABLE A.6. Information of the fleet.

Vehicle type	Internal /external	Number of vehicles	Number of compartments F_k	Compartment capacity c_k (liters)	Fixed cost β_k (RMB)	Unit mileage cost α_k (RMB)
1	Internal	3	4	12 000	500	3.11
2	Internal	3	3	11 000	400	2.85
3	Internal	3	3	7000	300	1.69
4	External	–	4	12 000	1000	3.11

TABLE A.7. Comparison between different initial solutions.

Round	Initial cost = 41 134.44 RMB			Initial cost = 49 332.04 RMB		
	Running time (seconds)	Cost (RMB)	Cost decrease (%)	Running time (seconds)	Cost (RMB)	Cost decrease (%)
1	343.83	21 668.11	47.32	551.96	21 315.94	56.79
2	296.95	21 473.48	47.80	707.62	21 684.05	56.04
3	333.85	20 959.35	49.05	579.38	22 280.8	54.84
4	371.78	22 463.92	45.39	577.76	21 789.17	55.83
5	350.25	21 551.86	47.61	369.03	22 373.17	54.65
6	251.92	22 607.43	45.04	650.48	21 591.82	56.23
7	355.68	22 071.59	46.34	397.75	21 899.78	55.61
8	344.98	21 895.03	46.77	519.78	19 979.61	59.50
9	425.8	21 629.2	47.42	370.32	20 701.65	58.04
10	489.1	20 618.1	49.88	533.53	21 218.91	56.99
Average	356.41	–	–	525.76	–	–