Computers & Industrial Engineering 172 (2022) 108597

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Modeling and solving a multi-trip multi-distribution center vehicle routing problem with lower-bound capacity constraints

Van Son Nguyen [a,*], Quang Dung Pham [b], Thanh Hoang Nguyen [c], Quoc Trung Bui [b]

[a] Academy of Cryptography Techniques, No. 141 Chien Thang Road, Hanoi, Viet Nam
[b] Hanoi University of Science and Technology, No. 1 Dai Co Viet Road, Hanoi, Viet Nam
[c] Phenikaa University, Hanoi, Viet Nam

ARTICLE INFO

Keywords:
Routing
Dairy product transportation
Multi-trip multi-distribution center VRPTW
Lower-bound capacity constraints

ABSTRACT

Vehicle routing problem (VRP) for delivering goods from distribution centers to customers is one of the main operations in logistics. Optimizing route plans for vehicles allows companies to save a huge amount of operational costs. The VRP problem is also one of the most studied problems in the domain of operations research. There are several variants of the VRP problem that have been considered in the literature. This paper proposes a new variant of the vehicle routing problem taking into account most of the well-studied features, especially with a new constraint on the lower bound of the capacity of vehicles which has not been investigated in the literature. The problem requirements come from one of the biggest dairy distribution companies in Vietnam. With over 1000 customer points included in a plan on average, the company takes at least one working day to make a route plan. We formulate the considered problem as a mixed-integer linear programming problem, analyze the challenges of the lower-bound capacity constraints and propose an adaptive large neighborhood search framework for solving it. Experimental results on large realistic and randomly generated instances show the efficiency and the applicability of the proposed method. The application of the proposed method led to the rapidity in generating the solution; this task that used to take one day is decreased to just two hours.

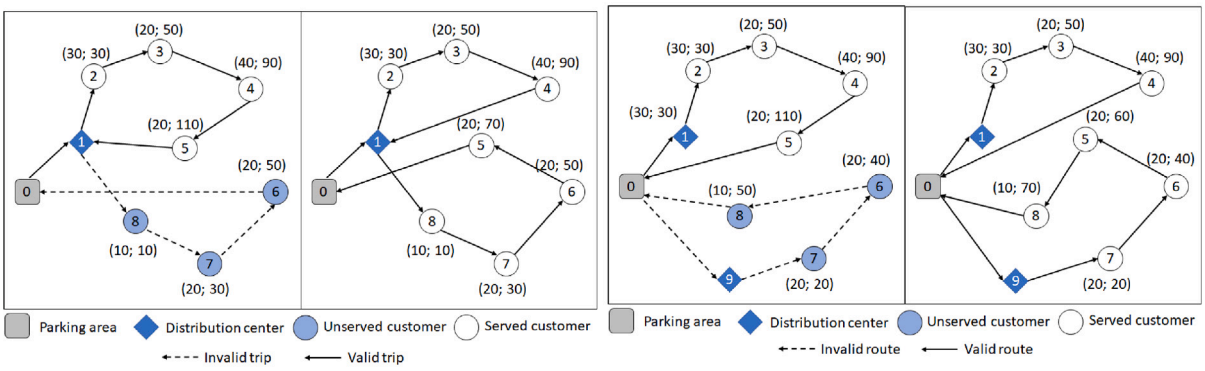
1. Introduction

Transportation is an important function of a supply chain, which includes the act of getting the finished goods from distribution centers to customers. For manufacturing companies, efficient routing of vehicles to distribute goods to customers is an intense problem faced to reduce the relevant costs of the total investment and operational costs. The problem is known as a vehicle routing problem (VRP). The VRP aims to generate the best routes for vehicles. Each vehicle will be loaded at a depot, routed to service a sequence of customers, and then returned to the depot. Each customer will be visited exactly once, and the demands of all customers must be satisfied. The objectives associated with the VRP vary across the literature. Its different variants have been widely explored in the literature, such as Capacitated VRP (CVRP) and VRP with time window (VRPTW). These problems are often more straightforward than real-life problems. Nowadays, a new tendency is mainly focused on applying these problems to real-life applications. Some typical real-world applications of VRPs are investigated, such as VRP for distributing beer (bottles and cans) in the Colombo region (Ganepola et al., 2018), VRP for e-commerce package delivery (Zhen et al., 2020), and VRP for autonomous delivery robots in urban logistics (Chen et al., 2021). In the case of VRP for distributing frozen and perishable items (e.g., foods, dairy products) in particular, tackling the VRP and making effective use of vehicle capacity simultaneously can enhance customer satisfaction and save transit costs. In this paper, we consider a VRP for the dairy product transportation in one of the biggest dairy company in Vietnam, which is the first companies in Vietnam to be included in Forbes Asia's 200 Best Under A Billion list (Forbes, 2010). The motivation of our study is to solve the problem of the company in Vietnam as well as similar delivery problems around the world. This paper fills a gap in the literature on the problem by combining some real-world factors, some of which exist in literature while the others have not been reported.

In our context, the company receives high volumes and demands from many customers having diverse positions every day. The company owns a fixed fleet of vehicles (in-house vehicles) and can lease several vehicles from several affiliated third-party logistics companies

* Corresponding author.
E-mail addresses: sonnv198@actvn.edu.vn (V.S. Nguyen), dungpq@soict.hust.edu.vn (Q.D. Pham), hoang.nguyenthanh@phenikaa-uni.edu.vn (T.H. Nguyen), trungbq@soict.hust.edu.vn (Q.T. Bui).

https://doi.org/10.1016/j.cie.2022.108597
Received 12 October 2021; Received in revised form 2 August 2022; Accepted 18 August 2022
Available online 25 August 2022



(a) Node transfer on different trips (b) Node transfer on different vehicles

Fig. 1. An example of node transfers to satisfy the capacity constraints, where the lower and the upper boundaries are 70 and 110, respectively.

(outsourced vehicles) when needed. Due to the limited capacity, the fixed fleet size, and time window constraints, vehicles must deliver dairy product units from multiple distribution centers to customers and operate multiple trips. Following their business strategy, the top priority objective is to minimize the number of unserved customers. Since outsourced vehicles are not always available, the secondary objective is to minimize the number of vehicles needed. The last objective is to minimize the total travel distance. As traditional constraints, the total customer demands in a trip should not exceed the maximum capacity. However, some trips of vehicles are scheduled to carry too little cargo in real-world situations due to tight time windows. For example, a vehicle with a capacity of 10 tons carries an amount of goods which is 100 kg, which is not realistic. These trips are canceled because this schedule provides low profitability and causes resource imbalances. Due to this reason, new lower-bound capacity constraints on vehicles are considered as the hard constraints in this paper. More precisely, the total weight of products transported in each trip must be within a given range depending on the capacity of the operating vehicle. The presence of lower-bound capacity constraints makes the problem more challenging. One of the strategies we apply is to handle lower-bound capacity constraints as soft constraints (accept violating trip), then transfer some customers of a feasible trip to other underweight trips as illustrated in Fig. 1. In this figure, the bound of capacity is $[70, 110]$ for each vehicle. If a vehicle is scheduled to perform two trips according to a sequence of points 0, 1, 2, 3, 4, 5, 1, 8, 7, 6, 0 (see Fig. 1(a)) then the second trip of the vehicle is violated the capacity constraint. In this case, there are three unserved customers. However, if we transfer customer 5 from the first trip to the second trip, the vehicle can perform both trips (i.e., there are no unserved customers). The same transfer is also applied to trips of different vehicles (see Fig. 1(b)). These transfers increase the number of served customers without increasing the number of vehicles needed.

In our problem, vehicles are located in different parking areas and must arrive at a distribution center to load dairy products. A parking area in this study can be viewed as a distribution center, and the distribution center can be viewed as a customer that must be visited. However, the modeling challenge here resides in the representation of multiple trips and the existence of three types of points: points visited one time in a one-way direction (parking areas), points visited many times (distribution centers) and points visited at most once (customers). Moreover, the loading time at a distribution center is not fixed for each trip. The loading time depends on the type and quantity of products loaded and, therefore, on the specific customers assigned to the trip. Besides, some customers were scheduled to be served by a given delivery person. However, the delivery was not completed in the past (in the previous delivery plan) due to some unpredicted reasons (e.g., traffic incidents). Hence, the same delivery person must deliver to these customers in the current plan. This fact implies that each vehicle has a set of specified customers only served by this vehicle.

To the best of our knowledge, the lower-bound capacity constraint has not been investigated in the literature for the vehicle routing problem class. Besides, the combination of these real-life constraints makes the problem more challenging. This paper defines the above problem as a multi-trip multi-distribution center vehicle routing problem with lower-bound capacity constraints (MTDLC-VR problem). The innovations of this article are as follows:

- We define a novel dairy product transportation problem for real-world applications that combine the following features:
  - There are three types of spatial points in a geographical region: parking points, distribution center points, and customer points.
  - Each vehicle can operate multiple trips and load dairy products at different distribution centers on different trips.
  - The total weight of dairy products loaded on the vehicle must be greater than or equal to the minimum weight on each trip.
  - The service-dependent loading times at points are considered. The loading time at distribution centers and the unloading time at customers depend on the demand of customers served on each trip.
  - Each vehicle has a set of forbidden customers and a set of prespecified customers.
- We formulate the considered problem with a mixed-integer linear programming (MILP) model. We performed experiments on small-scale real instances, adapted benchmark instances, and generated instances using GUROBI Optimizer to validate our model.
- We analyze the challenges of the new lower-bound capacity constraints, consider this fact as a soft constraint and put it (with a specified coefficient) to the objective function for comparison. We propose three efficient adapted construction algorithms to solve this problem. A strategy including generating initial solutions and subsequently applying an adaptive large neighborhood search (A-ALNS) to improve the quality of solutions is also proposed. The performance of the proposed A-ALNS algorithm is compared with that of other heuristics (including constructive algorithms and the local search algorithm in Pham et al., 2017) by conducting extensive numerical experiments to evaluate the applicability of the proposed algorithm in real-world applications. The results reveal that our algorithms can effectively solve large-scale instances with up to 1256 customers, four distribution centers, and two parking areas. The instances and our results are available on https://github.com/sonnv188/MTDLC-VR.git for further research and comparison.

This paper is structured as follows. A survey on research sources is presented in Section 2. Section 3 presents the description and a

MILP model for the MTDLC-VR problem. Section 4 analyzes the challenges of the new lower-bound capacity constraints on construction heuristics and proposes algorithms for solving the MTDLC-VR problem. Section 5 reports the computational experimentation. Conclusion and future works are given in Section 6.

2. Literature review

Dantzig and Ramser (1959) first introduced the VRP formulation as the truck dispatching problem that provides the optimal route of vehicles in a fleet to serve the demand for oil of several gas stations from a central hub. Since then, several variants of the problem have been put forward, including CVRP, VRPTW, VRP with multiple depots (MDVRP) and VRP with multiple trips (MTVRP). The most classical VRP is CVRP, where each customer has a given demand that has to be satisfied, and the total demand of the customers served in the same route does not exceed the upper-bound vehicle capacity. A review of some exact algorithms which were proposed in the past few decades can be found in Toth and Vigo (2014). The CVRP belongs to the class of NP-hard combinatorial problems. This statement is validated by the volume of literature that proposes incomplete algorithms to solve the problem (Ribeiro & Laporte, 2012). Recently, a hybrid large neighborhood search algorithm (Akpinar, 2016), integration of Lagrangian split with VNS (Bouzid et al., 2017), and an improved hybrid firefly algorithm (Altabeeb et al., 2019) are also proposed for solving this problem.

VRPTW is a well-known simple extension of VRP that extends the CVRP by adding time windows to a depot and customers. General and state-of-the-art surveys on this problem class are provided in Toth and Vigo (2014). In Song and Ko (2016), the authors implemented the Priority-based Heuristic Algorithm for solving perishable food products delivery problems to maximize customer satisfaction. In their model, customer satisfaction includes the freshness of delivered food. Thus, the time window constraint of each customer demand is relatively tight. Especially, the works in Baradaran et al. (2019) investigated VRP with multiple prioritized time windows for distributing confectionery and chocolate products in Iran. Due to the complexity of their problem, a binary artificial bee colony algorithm tuned via the Taguchi method is developed to solve it. Then, Konstantakopoulos et al. (2020) reviewed VRPTW applications related to freight transportation that most logistics and distribution companies face in their daily operations.

The MDVRP arises as a generalization of VRP, where vehicles depart from and return to multiple depot locations. Montoya-Torres et al. (2015) surveyed several studies on the MDVRP based on either complete or incomplete methods. In terms of complete algorithms, Bettinelli et al. (2011) presented a branch-and-cut-and-price algorithm to find the optimal solution for a variation of VRP in which vehicles have different capacities and fixed costs, located at different depots. An exact method-based heuristic is proposed by Gulczynski et al. (2011) to solve the MDVRP problem in which each subproblem becomes a single depot VRP and evolves independently in its domain space. In Du et al. (2017), four fuzzy simulation-based heuristic algorithms are also designed to search the exact solution of the MDVRP for hazardous materials transportation. Regarding incomplete algorithms, Alinaghian and Shokouhi (2018) and Bezerra et al. (2018) proposed variants of the Variable Neighborhood Search algorithm for solving the MDVRP problem. In Alinaghian and Shokouhi (2018), the authors considered a multi-depot multi-compartment vehicle routing problem, where the cargo space of each vehicle has multiple separate compartments to transport different products simultaneously. In the context of our problem, these compartments of vehicles are adjustable. Therefore, we only consider constraints on the total available cargo space on each vehicle. Recently, many heuristic methods have been developed in the context of the MDVRP, including the genetic algorithm (Vidal et al., 2014) and the local search algorithm (Kramer et al., 2019). A complex and difficult variant of MDVRP arising in several real-life applications is Multi-Depot Open Vehicle Routing Problem (MDOVRP), which was firstly proposed by Tarantilis and Kiranoudis (2002) for distributing fresh meat in Greece. In the MDOVRP, vehicles are not required to return to the depots at the end of their routes. Recently, Liu et al. (2014) proposed a new exact formulation for the MDOVRP. Lalla-Ruiz et al. (2016) improved some constraints in Liu et al. (2014) and indicated that their proposed model outperforms the existing one. However, the complex objectives and constraints often lead to various solution spaces. These challenges lead to considerable research efforts in embedding learning mechanisms for more efficient neighborhood search through adaptive neighborhood selection. In Lahyani et al. (2019), the authors proposed a hybrid ALNS algorithm that is competitive compared with solutions published by Lalla-Ruiz et al. (2016) and Liu et al. (2014). Their algorithm combines the ALNS strategy with three insertion, five removal heuristics, and four post-optimization local search procedures. In our work, the ALNS algorithm is also adapted for dealing with large problem instances. Regarding the proposed model, a vehicle can also visit different depots on different trips, but it must return to a parking area after serving the last customer. Furthermore, the advanced extension of the problem by combining MDVRP with MTVRP is studied in our model.

The MTVRP was first proposed by Fleischmann (1990), where a vehicle makes several journeys during a day. Olivera and Viera (2007) proposed a mathematical programming model based on a set covering formulation for MTVRP. Recently, a branch-and-price algorithm in Hernandez et al. (2016), and the first exact solution framework based on a novel structure-based formulation in Paradiso et al. (2020) were presented to solve the MTVRP. For incomplete algorithms, it is common to solve the MTVRP by combining VRP and packing procedure (Babaee Tirkolaee et al., 2017; François et al., 2016; Louveaux & Salazar-González, 2016). The authors in Louveaux and Salazar-González (2016) studied the one vehicle version with multi-trip in a route (a trip is called *leg* by the authors) and considered the capacity of the vehicle as an output (i.e., the vehicle capacity is not fixed). That is, their model proposed an approach to select the best capacity and the best route, in order to minimize the acquisition cost function and the distance traveled. In François et al. (2016), the authors proposed an adapted ALNS algorithm that iteratively modifies a CVRP solution and applies bin packing techniques to assign the created routes to available vehicles. Their model assumed that the serving time at the depot was fixed. In contrast, the loading time-dependent at the depot is considered in works of Pan et al. (2021) which designs routes to replenish stocks for a cafe company around Singapore. They also developed a hybrid ALNS algorithm that employs two-variable neighborhood descend operators. However, both of those papers considered only one single depot. A survey on the MTVRP can be found in Cattaruzza et al. (2018).

A combination of the above problems has more practical applications. Recently, more attention has been focused on the multi-trip multi-depot VRP with specific combinations of real-life constraints. They also apply their model to transportation problems for different kinds of commodities. For example, Masmoudi et al. (2016) proposed three metaheuristics to solve the multi-trip multi-depot heterogeneous Dial-a-Ride problem. Two-hybrid bee algorithms and an ALNS are presented. These methods are highly effective and efficient. Wang (2018) also considered a multi-trip multi-depot VRP problem in which vehicles must pick up meals from multiple suppliers and deliver them to customers. Their problem under logistics resource sharing is solved by two popular algorithms: local search and ALNS. In the ALNS algorithm, the authors proposed a supplier-oriented removal operator and three rules to choose a specific range of insertion for each insertion operator. Their results show that the sharing service has a significant advantage over the exclusive service. In addition, Zhen et al. (2020) first proposed a multi-trip multi-depot VRPTW with release dates. Their problem originates from the e-commerce package delivery in China. In their model, each trip starts and ends at the same depot. The release date introduced in their paper is when the package requested by a customer

Table 1
A summary of the related papers.

Reference	Constraints							Solution methods	
	Multi-trip	Multi-depot	Trip limitation	No start at depot	Service-dependent time	Time window	Lower-bound capacity	Complete	Incomplete
Masmoudi et al. (2016)	√	√	√		√	√			√
Alinaghian and Shokouhi (2018)		√			√	√		√	√
Wang (2018)	√	√	√			√			√
Paradiso et al. (2020)	√					√		√	
Zhen et al. (2020)	√	√				√			√
Pan et al. (2021)	√		√		√	√			√
Our work	√	√	√	√	√	√	√		√

in the distribution center can be delivered by a vehicle. Therefore, they considered the practices that vehicles can only deliver available packages that have been released. It is solved by a hybrid particle swarm optimization algorithm and a hybrid genetic algorithm. A summary of the assumptions of related papers is presented in Table 1.

Our research has similarities with the above studies, such as vehicles performing multiple trips from multiple depots that guaranteed time window and upper-bound capacity constraints and the service-dependent time at each point. The difference of our study is exploring logistics services and combining real-world constraints in dairy product delivery, where dairy product companies want to maximize the number of served customers and minimize the number of vehicles related to their availability. In our model, there are three types of points with different accessibility: points visited one time in one-way direction (parking areas), points visited many times (distribution centers), and points visited at most once (customers). Besides, the existence of service sharing leads to the objective function of minimizing outsourcing costs. The loading time at distribution centers and the unloading time at customers are depended on the product type and the product quantity. Moreover, in the context of the MTDLC-VR problem, dairy products are always available at distribution centers. Managers are often concerned with occupancy product rates in vehicles. Thus, new lower-bound capacity constraints are added to our model. To the best of our knowledge, no work has been conducted to address this problem.

3. Problem description and formulation

In this section, we formally define the MTDLC-VR problem and then formulate a MILP model that can be directly solved by commercial solvers. A detailed description of this problem is as follows.

3.1. Problem description

A vehicle scheduling system receives a large number of requests to transport dairy products from distribution centers to customers during fixed working hours. Each request consists of information about the customer location, a valid time window, and demand weight. The system performs scheduling and routing a fleet of heterogeneous vehicles that execute their itinerary consisting of several trips: a vehicle departs from its parking area, operates a sequence of trips, and returns to its parking area on a working day. This problem aims to obtain the maximum number of served customers, the minimum number of vehicles needed, and the best routes of vehicles to minimize the total travel distance.

The problem can be formally defined as an optimization problem (called the MTDLC-VR problem) on a directed network $G = (V, E)$, where V is a set of vertices that consists of a set of parking areas PK, a set of distribution centers D and a set of customers C, and set E is an arc set that corresponds to connections between the vertices. In this context, no arc starts from a parking area to the other parking areas, from parking areas to customers, from a distribution center to parking areas, and from a distribution center to the other distribution centers. A travel time $t_{i,j}$ and a travel distance $d_{i,j}$ is associated with each arc $(i, j) \in E$. We denote a set of products by P. Each product $p \in P$ has weight $w(p)$ and is available for delivery at each distribution center. Each distribution center $dp \in D$ has a window of working time $[e(dp), l(dp)]$, the waiting time duration $t_{wait}(dp)$ to load products and the duration of loading a weight unit $t_{unit}(dp)$. At each distribution center, the more cargo the vehicle carries on a trip, the longer it takes to load. The notation K denotes a set of heterogeneous vehicles. Each parking area $pk \in PK$ has its own fleet of vehicles that must leave and return within a predefined time window $[e(pk), l(pk)]$. For each vehicle k, the number of performed trips does not exceed $q(k)$. Besides, each vehicle $k \in K$ has an upper-bound of total weight $\bar{c}(k)$ and a priority coefficient f_k. Since the cost of using an in-house vehicle is less than the cost of using an outsourced vehicle, $f_k < f_{k'}$ if vehicle k is an in-house vehicle and vehicle k' is an outsourced vehicle. Especially, transportation companies often refuse to perform trips that do not reach the minimum total weight to optimize the profits. To cover this practical constraint, we define a lower-bound capacity $\underline{c}(k)$ for each vehicle k. In addition, since some customers have specific requirements for vehicles (e.g., heavy vehicles or vehicles without licenses are not allowed to access the special customer), each vehicle has a set of customers that it cannot serve. A parameter $rc(k, c)$ is associated with vehicle $k \in K$ and customer $c \in C$ where $rc(k, c) = 1$ implies that customer c must be served by vehicle k, $rc(k, c) = 0$ otherwise. A parameter $rp(k, p)$ specifies that vehicle $k \in K$ can carry product $p \in P$ or not (for example, vehicles with no cooling compartment cannot carry frozen commodities). Each customer $c \in C$ has a demand $dm(c, p)$ for each product $p \in P$. Customers must be served in their time windows (the earliest and latest arrival time at customer $c \in C$ are $e(c)$ and $l(c)$, respectively). We introduce parameters $t_{wait}(c)$ and $t_{unit}(c)$ representing the average waiting time duration to start service and the unloading time duration for a weight unit at customer $c \in C$, respectively.

An example of vehicle itineraries is illustrated in Fig. 2 in which a vehicle's itinerary is scheduled as a sequence of points $0, 1, 2, 3, 4, 5, 6, 7, 8, 0$. This vehicle performs two trips, and the goods on each trip are loaded at different distribution centers. While another vehicle must follow the itinerary $9, 10, 11, 12, 13, 10, 14, 15, 16, 9$. The goods delivered to customers on different trips are loaded at the same distribution center in this itinerary. The bound of capacity is $[70, 110]$ for each vehicle.

3.2. Notations and definitions

Notations and definitions used for the MTDLC-VR problem are presented in two tables. Table 2 lists all notations and the relevant parameters used. Table 3 shows the modeling variables.

3.3. Model formulation

This section introduces a MILP formulation for the MTDLC-VR problem with all information (i.e., information of vehicles, customers, distribution centers, parking areas and requests) is known in advance. We denote the number of unserved customers by $g_r = \eta - \sum_{i \in C} y_i$, the number of vehicles needed by $g_v = \sum_{k \in K} f_k z^{k,1}$, and the total travel distance by $g_c = \sum_{k \in K} \sum_{q=1}^{q(k)} \sum_{(i,j) \in E} d_{i,j} x_{i,j}^{k,q}$. The MTDLC-VR problem formulation is as follows:

$$\text{Minimize} F = \widehat{f} g_r + g_v + g_c \quad (1)$$

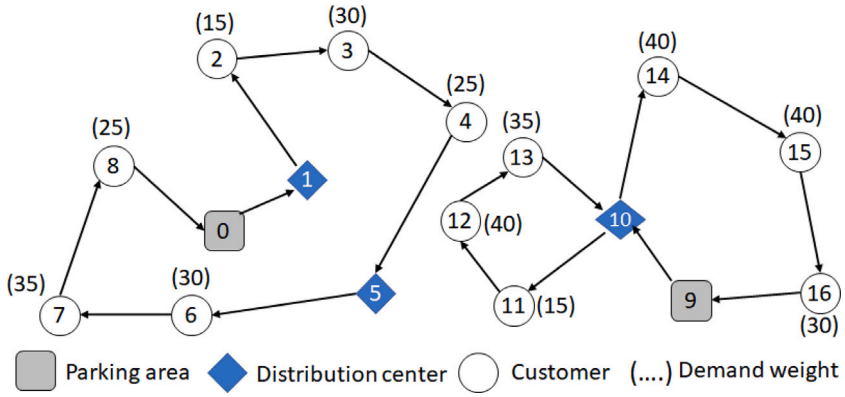


Fig. 2. An example of vehicle itineraries in the MTDLC-VR problem.

Table 2
Sets and parameters.

Notation	Definition
PK	Set of parking areas. For each $pk \in PK$: $[e(pk), l(pk)]$: time window of parking area pk.
P	Set of products. For each $p \in P$: $w(p)$: the weight of an unit of product p.
D	Set of distribution centers. For each $dp \in D$: $[e(dp), l(dp)]$: time window of distribution center dp. $t_{wait}(dp)$: the waiting duration for starting service at distribution center dp. $t_{unit}(dp)$: the time duration for loading a weight unit at distribution center dp.
C	Set of customers. We denote the number of customers by η. For each $c \in C$: $[e(c), l(c)]$: time window of customer c. $t_{wait}(c)$: the waiting duration for delivery at customer c. $t_{unit}(c)$: the time duration for unloading a weight unit at customer c. $d(c, p)$: the demand for product p of customer c.
V	Set of all points. $V = PK \cup D \cup C$
K	List of vehicles. $K = \{0, 1, 2, \dots, \kappa - 1\}$, for each $k \in K$: $p(k) \in P$: the parking area of vehicle k. f_k: priority coefficient for using vehicle k. $\underline{c}(k)$: the lower boundary of capacity of vehicle k. $\overline{c}(k)$: the upper boundary of capacity of vehicle k. $q(k)$: the number of trips of vehicle k.
E	Set of arcs defined by arc (i, j), $\forall (i, j) \in (PK \times D) \cup (D \times C) \cup (C \times C) \cup (C \times D) \cup (C \times PK)$ and $i \neq j$ $\delta^+(i) = \{j \mid (i, j) \in E\}$ $\delta^-(i) = \{j \mid (j, i) \in E\}$
$dm_w(i, p)$	A parameter presents total weight demand for product p at point i, $\forall i \in V, \forall p \in P$. $dm_w(i, p) = 0, \forall i \in PK \cup D, \forall p \in P$. $dm_w(i, p) = dm(i, p) * w(p), \forall i \in C, \forall p \in P$.
$dr(i)$	A parameter presents the serving duration at point $i, \forall i \in V$. $dr(pk) = 0, \ \forall pk \in PK$. $dr(dp) = t_{wait}(dp), \ \forall dp \in D$. $dr(i) = t_{wait}(i) + \sum_{p \in P} dm_w(i, p) t_{unit}(i), \forall i \in C$.
$rc(k, c)$	Access restriction of vehicle k to customer c, $\forall c \in C, \forall k \in K$. $rc(k, c) = 1$: vehicle k can access to customer c $rc(k, c) = 0$: otherwise.
$rp(k, p)$	$rp(k, p) = 1$: vehicle k is allowed to carry product p, $\forall k \in K, \forall p \in P$. $rp(k, p) = 0$: otherwise.
$vc(k, c)$	$vc(k, c)$ is equal to 1 if customer c is a specified customer of vehicle k, or equal to zero otherwise, $\forall k \in K, \forall c \in C$.
$t_{i,j}$	The travel time from point i to point j, $\forall (i, j) \in E$.
$d_{i,j}$	The travel distance from point i to point j, $\forall (i, j) \in E$.
$\hat{f}$	Penalty coefficient of one unserved customer.
M	M is a big constant.

Table 3
Modeling variables.

Notation	Definition
$x_{i,j}^{k,q}$	A binary variable equals one if vehicle k travels on arc (i, j) in the qth trip; otherwise, equals zero, $\forall k \in K$, $\forall (i, j) \in E$, $\forall q = 1, 2, \dots, q(k)$.
$w_p^{k,q}$	The total weight of product p transported by vehicle k in the qth trip, $\forall k \in K$, $\forall q = 1, 2, \dots, q(k)$, $\forall p \in P$.
y_i	A binary variable equals one if customer i is served; otherwise, equals zero, $\forall i \in C$.
$z^{k,q}$	A binary variable equals one if vehicle k executes the qth trip; otherwise, equals zero, $\forall k \in K, \forall q = 1, 2, \dots, q(k)$.
$s_i^{k,q}$	A variable to identify the starting service time of vehicle k at point i in the qth trip, $\forall k \in K, \forall q = 1, 2, \dots, q(k)$, $\forall i \in C$.

subject to

- Flow conservation constraints

$$\sum_{j\in\delta^+(i)}\sum_{q=1}^{q(k)} x_{i,j}^{k,q} = \sum_{j\in\delta^-(i)}\sum_{q=1}^{q(k)} x_{j,i}^{k,q},\ \forall k\in K, \forall i\in V \tag{2}$$

- Flow conservation constraints on each trip

$$\sum_{j\in\delta^+(i)} x_{i,j}^{k,q} = \sum_{j\in\delta^-(i)} x_{j,i}^{k,q},\ \forall k\in K, \forall q = 1,2,\dots,q(k), \forall i\in C \tag{3}$$

- If vehicle k comes back to distribution center dp, it must travel to a customer on the next trip

$$\sum_{j\in\delta^-(dp)} x_{j,dp}^{k,q-1} = \sum_{j\in\delta^+(dp)} x_{dp,j}^{k,q},\ \forall k\in K, \forall q = 2,3,\dots,q(k), \forall dp\in D \tag{4}$$

- The starting time of vehicles is equal to the earliest service time of its parking area

$$\mathsf{s}_{p(k)}^{k,1} = e(p(k)),\ \forall k\in K \tag{5}$$

- Constraints on the start of service time at the next point of a distribution center in a trip, where the serving duration at a distribution center includes the duration for loading all goods on a trip

$$\mathsf{s}_i^{k,q} + t_{i,j} + dr(i) + M(x_{i,j}^{k,q} - 1) + \sum_{p\in P} w_p^{k,q} t_{unit}(i) \le \mathsf{s}_j^{k,q}, \tag{6}$$
$$\forall k\in K, \forall q = 1,2,\dots,q(k),$$
$$\forall (i,j)\in D\times C$$

- Constraints on the start of service time between two points, where they are not distribution centers

$$\mathsf{s}_i^{k,q} + t_{i,j} + dr(i) + M(x_{i,j}^{k,q} - 1) \le \mathsf{s}_j^{k,q}, \tag{7}$$
$$\forall k\in K, \forall q = 1,2,\dots,q(k),$$
$$\forall (i,j)\in (PK\times D)\cup(C\times C)\cup(C\times PK), i\ne j$$

- Constraints on the start of service time between the last visited point in a trip and the visited distribution center for the next trip

$$\mathsf{s}_i^{k,q-1} + t_{i,j} + dr(i) + M(z^{k,q} - 1) + M(x_{i,j}^{k,q-1} - 1) \le \mathsf{s}_j^{k,q}, \tag{8}$$
$$\forall k\in K, \forall q = 2,3,\dots,q(k), \forall (i,j)\in C\times D$$

- Constraints on time windows

$$\mathsf{s}_i^{k,q} + M(\sum_{j\in\delta^+(i)} x_{i,j}^{k,q} - 1) \le l(i), \tag{9}$$

$$\mathsf{s}_i^{k,q} + M(1 - \sum_{j\in\delta^+(i)} x_{i,j}^{k,q}) \ge e(i), \tag{10}$$
$$\forall k\in K, \forall q = 1,2,\dots,q(k), \forall i\in V$$

- Constraints define the total weight of products loaded at each distribution center

$$w_p^{k,q} = \sum_{(i,j)\in E} dm_w(i,p) x_{i,j}^{k,q},\ \forall k\in K, \forall q = 1,2,\dots,q(k), \forall p\in P \tag{11}$$

- Constraints on the capacity of vehicles, where the total weight of products loaded at a distribution center on each vehicle must be greater than or equal to the lower boundary and less than or equal to the upper boundary of vehicle capacity

$$\sum_{p\in P} w_p^{k,q} + M(z^{k,q} - 1) \le \overline{c}(k), \tag{12}$$

$$\sum_{p\in P} w_p^{k,q} + M(1 - z^{k,q}) \ge \underline{c}(k), \tag{13}$$
$$\forall k\in K, \forall q = 1,2,\dots,q(k)$$

- Each customer is visited at most once

$$\sum_{k\in K}\sum_{q=1}^{q(k)}\sum_{i\in\delta^-(j)} x_{i,j}^{k,q} \le 1,\ \forall j\in C \tag{14}$$

- Each distribution center is visited at most once on a trip

$$\sum_{dp\in D}\sum_{j\in C} x_{dp,j}^{k,q} = z^{k,q},\ \forall k\in K, \forall q = 1,2,\dots,q(k) \tag{15}$$

- Each vehicle starts from its parking area at most once

$$\sum_{dp\in D} x_{p(k),dp}^{k,1} = z^{k,1},\ \forall k\in K \tag{16}$$

- Each vehicle must only come back to its parking area at most once

$$\sum_{q=1}^{q(k)}\sum_{j\in C} x_{j,p(k)}^{k,q} = z^{k,1},\ \forall k\in K \tag{17}$$

- Constraints define relationships between decision variables

$$z^{k,q} \le \sum_{(i,j)} x_{i,j}^{k,q},\ \forall k\in K, \forall q = 1,2,\dots,q(k),\ \forall (i,j)\in E \tag{18}$$

$$x_{i,j}^{k,q} \le z^{k,q}, \forall k\in K, \forall q = 1,2,\dots,q(k), \forall (i,j)\in E \tag{19}$$

$$\sum_{k\in K}\sum_{q=1}^{q(k)}\sum_{i\in\delta^-(j)} x_{i,j}^{k,q} = y_j,\ \forall j\in C \tag{20}$$

- Constraints on the sequence of trips

$$z^{k,q+1} \le z^{k,q},\ \forall k\in K, \forall q = 1,2,\dots,q(k)-1 \tag{21}$$

- Constraints on the restriction of vehicles that have access to customers

$$\sum_{q=1}^{q(k)}\sum_{j\in\delta^-(i)} x_{j,i}^{k,q} \le M.rc(k,i),\ \forall k\in K, \forall i\in V \tag{22}$$

- Constraints on the restriction of vehicles cannot carry some specified products

$$\sum_{q=1}^{q(k)} w_p^{k,q} \le M.rp(k,p),\ \forall k\in K, \forall p\in P \tag{23}$$

- Constraints on the prespecified customers of each vehicle

$$vc(k,i) \le \sum_{q=1}^{q(k)}\sum_{j\in\delta^-(i)} x_{j,i}^{k,q},\ \forall k\in K, \forall i\in C \tag{24}$$

The objective function (1) states that the number of unserved customers, the number of vehicles needed and total travel distance should be minimized. The number of unserved customers is firstly minimized with a large value of $\hat{f}$. If f_k is a large constant ($\forall k\in K$), the model will primarily minimize the number of vehicles needed.

4. The solution methods

4.1. Notations for heuristic algorithms and solution evaluation

Due to the huge computational complexity, GUROBI Optimizer is only able to solve small problem instances of the MTDLC-VR problem. Therefore, one of our contributions is to propose heuristic algorithms to handle large instances of the problem. To simplify the presentation of our proposed heuristic algorithms, we give some notations that are summarized as follows.

A solution of the MTDLC-VR problem is denoted by $s = \langle s^0, s^1, s^2, \dots, s^{\kappa-1}\rangle$, where s^k is a route of vehicle k represented by a sequence of points $\langle s_1^k, s_2^k, \dots, s_{\ln(k)}^k\rangle$. The points s_1^k and $s_{\ln(k)}^k$ are the starting and the terminal points of vehicle k, respectively (these points correspond to a parking area). Points s_i^k, $(i = 2,\dots,\ln(k)-1)$ are intermediate points

that are visited by vehicle k. An empty solution contains routes that are initialized without intermediate points (i.e. $\ln(k) = 2$). In our model, multiple vehicles can start and end at the same parking area. They can access a distribution center, simultaneously. Moreover, each vehicle can arrive at a distribution center many times for multi-trip. Therefore, to ease the formulation in which each point is visited at most once, we introduce a set of logical points $LD(p)$ associated with physical point $p \in PK \cup D$, i.e. each point of $LD(p)$ refers to physical point p, $\forall p \in PK \cup D$. Let $D^* = \bigcup_{dp \in D} LD(dp)$ and $PK^* = \bigcup_{pk \in PK} LD(pk)$. From these setting, in the route of vehicle k, $s^k_1, s^k_{\ln(k)} \in PK^*$, $s^k_i \in C \cup D^*$, $(i = 2, \ldots, \ln(k) - 1)$. We denote:

- $V^* = PK^* \cup D^* \cup C$.
- $\ln(k)$: the length (number of points) of route s^k, $\forall k \in K$.
- $nx(p)$: the successor of point p in its route, $\forall p \in V^* \setminus PK^*$.
- $pr(p)$: the previous point of point p in its route, $\forall p \in V^* \setminus PK^*$
- $ix(p)$: the index of the route containing point $p, \forall p \in V^*$. $ix(p) = \perp$ if point p is not in any route.
- $ox(p)$: the order of point p in the route $ix(p), \forall p \in V^*$
- $ac(p)$: the accumulated travel distance of the route from the starting point of route $ix(p)$ to $p, \forall p \in V^*$
- $ad(s^k, s^k_i)$: the number of distribution center points visited by vehicle k before leaving point s^k_i of the route s^k.
- $I(s^k, p, s^k_i)$: the operator that inserts point p right after point s^k_i of route s^k, and returns a new route.
- $R(s^k, s^k_i)$: the operator that removes point s^k_i from route s^k, and returns a new route.
- $\mathcal{R}(s, s^k, s^b)$: the operator that replaces route s^k of solution s by the new route s^b and returns a new solution.

The objective function F in the MILP model described above combines three components: the number of unserved customers g_r, the number of using vehicle g_v, and the total travel distance g_c in a linear way with coefficients. Due to the problem requirements described by the company, g_r is treated with the highest priority, and g_c is treated with the lowest priority. Hence, in our proposed heuristic algorithms, we form an objective function $F(s)$ over a solution s as a vector of three components $(g_r(s), g_v(s), g_c(s))$. Three components of $F(s)$ are treated in a lexicographic order: given two solutions s_1 and s_2, we denote $F(s_1) < F(s_2)$ if:

- $g_r(s_1) < g_r(s_2)$ or
- $g_r(s_1) = g_r(s_2)$ and $g_v(s_1) < g_v(s_2)$ or
- $g_r(s_1) = g_r(s_2)$ and $g_v(s_1) = g_v(s_2)$ and $g_c(s_1) < g_c(s_2)$

4.2. Analysis of the challenges of the new capacity constraints in the MTDLC-VR problem

An impressive number of heuristics have been proposed for the VRP. Initially, these were mainly standard construction algorithms, whereas more recently, powerful improvement heuristic approaches have been developed. Several attempts have also been made to integrate construction and improvement heuristics. Improvement heuristics are often used to improve initial solutions generated by construction algorithms. In this section, we first review some most widely-used construction algorithms and analyze the impact of real-life constraints (12) and (13) on these algorithms. And then, we propose adapted construction algorithms to construct feasible solutions to the MTDLC-VR problem.

4.2.1. A review of construction heuristics

Construction heuristics work by inserting customers one at a time into partial routes until a feasible solution is obtained. Construction heuristics are mainly distinguished by how to select customers and by the method used to determine where a customer should be inserted. In this part, we mainly concentrate on the most widely used construction heuristics including Saving algorithm (SA) (Clarke & Wright, 1964), Greedy Insertion algorithm (GIA) (Ioannou et al., 2001), cluster-first route-second (CF-RS) methods like the Sweep heuristics (SW) (Gillett & Miller, 1974), and route-first cluster-second heuristics (RF-CS) (Prins et al., 2014).

The SA algorithm and its adaptions have been successfully used for solving the VRP (Abdelmaguid et al., 2009). The SA computes the savings of all customer pairs and sorts in non-increasing orders. In the first stage, each customer appears separately in a route. In the second stage, the sorted customer pairs are examined sequentially, and two routes are merged whenever this is feasible. The method has been improved recently: combination with tournament and roulette wheel selections (Pichpibul & Kawtummachai, 2013) and the combination with MILP (Li et al., 2016). In Mirzaei and Seifi (2015), the authors implemented an efficient saving-based algorithm to solve the routing problem for distributing perishable products.

The GIA proposed firstly by Ioannou et al. (2001) can be classified as a route-construction sequential approach. This algorithm results in relatively trivial-to-good solutions in a reasonable amount of time. There are two steps to insert a new customer into a route in the algorithm. In the first step, The best customer to be inserted in the current route is determined. After that, the algorithm explores the best insertion position in the current route. As soon as the current customer cannot be inserted into the current route, it is inserted into a new route. Their procedure is repeated until all customers are served; or there is no feasible insertion. In Zou and Dessouky (2018), an extension of this method was proposed by changing the definition of the impact measure to determine the best customer at each iteration. The GIA is also used in Liang et al. (2020) as an insertion operator in the ALNS algorithm to solve the distribution problem of a supermarket in China.

Other classical heuristics are two-phase methods. The solution process is decomposed into two fundamental components: clustering and routing. In the CF-RS approach, customers are first grouped into clusters, and each route is separately defined by solving a Traveling Salesman Problem within each cluster. The most widely used technique, the SW, has been proposed by Gillett and Miller (1974) for the clustering phase and referred to as the first example of this approach. Another adaptation of these ideas can be found in Na et al. (2011). An application of SW to the MDVRP problem and performance comparison between SW and SA can be found in Granada-Echeverri et al. (2018).

In contrast, an RF-CS method constructs a giant TSP tour of overall customers in the first stage and later subdivides into feasible trips. There are some advantages to this approach. The computation of a giant tour is sometimes more straightforward than the clustering in the first phase. Moreover, Beasley (1983) observed that the second phase could be solved exactly as the shortest path problem in an auxiliary graph. Although this exciting property is clear, almost twenty years later, Laporte et al. (2000) stated that there is no computational experience showing that the approach is competitive with other approaches. However, since 2002, the RF-CS approach has led in the last decade to successful construction heuristics and metaheuristics for CVRP, CARP, and VRPTW problems. The detail can be found in Prins et al. (2014).

We see that these methods are used to insert customers sequentially to a route without violating the vehicle capacity constraint. Additionally, we have to check for violations of the time windows when a customer is inserted. However, due to the existence of a lower-bound capacity constraint, the violation of this constraint is increased when the first customer is inserted into a vehicle and eliminated when the weight carried by the vehicle is greater than or equal to the lower-bound capacity. The impact of this constraint is described in detail in the following section.

4.2.2. The challenges of the capacity constraints on construction heuristics

In this section, we analyze the challenge faced when constructing an initial solution in the case that the lower-bound capacity constraint is considered as a hard constraint. During the route constructed by heuristic algorithms, customers are inserted into the route so that all the constraints are satisfied, including constraints (12) and (13), saying

that the total weight of dairy products delivered in each trip must not exceed the upper boundary of vehicle capacity or less than the lower boundary of vehicle capacity. We observe that the qth trip of vehicle k only satisfies both constraints (12) and (13) if one of the following conditions is verified:

- $\ln(k) = 2$.
- $\underline{c}(k) \leq \sum_{i \in \{i \mid ad(s^k,i)=q\}} \sum_{p \in P} dm_w(s_i^k, p) \leq \overline{c}(k)$.

In the initial state, there is no violation. When we insert a first customer into a trip of a vehicle, the violation of the lower-bound capacity constraints occurs (the corresponding vehicle is loaded with a certain amount of good but this amount is less than the lower bound). The violation will only disappear when customers are inserted into this trip such that the accumulated weight of products is enough to satisfy the lower-bound capacity constraint (13). At the same time, this weight must satisfy the upper-bound capacity constraint (12). It implies that we must explore a set of customers that satisfies the lower-bound and upper-bound capacity constraints at the same time or we insert nothing. However, customers are sequentially inserted into a route so that all the constraints are satisfied by some classical heuristics such as the SA and the GIA. Thus, we found that these algorithms cannot be applied to this problem directly.

For the CF-RS approach, we have to assign customers to a cluster that satisfies constraints (12) and (13) at the same time during the clustering phase. For each cluster, the problem can be defined as follows. We denote a set of numbers $S = \{\alpha_0, \alpha_1 \ldots, \alpha_{n-1}\}$ where α_i is the demand of unassigned customer i and a target range $[\underline{c}, \overline{c}]$ where $\underline{c}$ and $\overline{c}$ are the lower and upper boundaries of the vehicle capacity, respectively. We have to find a subset whose sum is within the range $[T, T + \Delta T]$, where $T = \underline{c}$ and $\Delta T = \overline{c} - \underline{c}$.

Proposition 1. *The above problem is NP-complete.*

Proof. We found that the original problem can be decomposed into two dependent problems:

- Problem A: given a set of numbers $S = \{\alpha_0, \alpha_1 \ldots, \alpha_{n-1}\}$, we have to find a subset $S^* \subset S$ such that the sum of its elements is equal to ΔT.
- Problem B: given a set of numbers $S' = \{\alpha_0, \alpha_1 \ldots, \alpha_{n-1}, \beta_0, \beta_1, \ldots, \beta_{m-1}\}$, where $\beta_0, \ldots, \beta_{m-1}$ are different elements of S^*, we have to find a subset such that the sum of its elements is equal to $T + \Delta T$.

We found that problem B is actually an instance of the original problem. Indeed, by removing the elements of S^* from the outcome of problem B, we obtain the outcome of the original problem. Both problems A and B are commonly known as Karp's 21 NP-complete (Karp, 2009). □

That proposition shows that if the number of customers is large, the clustering process to make a feasible trip that satisfies constraints (12) and (13), in general, cannot be solved in polynomial time. Therefore, to obtain a viable solution with a large number of customers served, the CF-RS approach is probably not effective.

On the other hand, for building feasible trips, instead of trying to find a set of customers that satisfies all the constraints like the above approaches, the RF-CS approach relaxes the vehicle capacity constraints to build a giant TSP tour and then splits the giant tour into feasible trips. This approach is effective because it reduces the complexity of the real-life capacity constraints which has been specified. Moreover, in Prins et al. (2014), the authors stated that the giant tour determined in the first phase could be used to consider an ordering of customers or a priority list before building the routes. In the context of this problem, they are represented by constraints (22)–(24). It is why we prefer to speak about the effectiveness of the RF-CS approach in this part and the sequel.

4.2.3. Splitting procedure

In order to guarantee constraint (12), after a customer point is added in a route, we consider inserting a distribution center point $dp \in D^*$ into the route of vehicle k so that vehicle k must come back to distribution center dp to load cargo for the next customer if need be. Also, the number of visited distribution centers by vehicle k after leaving a point $c \in C$ must not be greater than the maximum number of trips of vehicle k. Constraint (13) is only checked after all feasible customers are inserted into a route. It implies that we must relax the constraint when we consider inserting a customer $c \in C$ into route s^k. After that, when we have inserted distribution center points, we check constraint (13) for trips just constructed. Therefore, the MTDLC-VR problem can be considered as a problem of two decision levels. In the first level, we insert customers to routes checking the feasibility of the constructed routes without taking into account the constraint (13). In the second level, the constraint (13) is checked on each route, and then we remove infeasible trips. It is meant that two different problems are solved at two levels, but the solution obtained in the second level depends on the solution of the first level.

We denote the operator that inserts distribution center points into route s^k by $SP(s^k)$. The operator is described in Algorithm A1 (see the Appendix). Lines 2–4 find the nearest distribution center for the first trip. The constraint (12) is checked at line 7. Lines 12–13 select the nearest distribution center point and insert that point into the route s^k. Each added distribution center point splits the last tour into two consecutive trips in which the first order trip satisfies the constraint (12). Customers that cause constraint violations on the last allowed trip are removed from its route (lines 8–10). Constraint (13) on each route s^k is checked in operator $RCV(s^k)$ which is depicted in Algorithm A2 (see the Appendix). After trips of vehicle k are created by adding distribution center points, we compute the total weight on each trip. G is a list of all customers on the considered trip. If the trip does not satisfy the constraint (13) (line 7), it will be removed from its route (lines 8–10).

4.3. Adapted construction algorithms with splitting procedure

In this section, based on widely used construction algorithms SA, SW, and GIA, we propose three adapted construction algorithms that are used in the construction phase. Based on the idea of the RF-CS approach, SA, SW, and GIA algorithms are adapted for the MTDLC-VR problem by temporarily relaxing the constraint (13).

- **An Adapted Saving Algorithm (vSA)**: In order to adapt the SA for the MTDLC-VR problem, including time window constraints, it is necessary to take into account the route direction, the travel time, the waiting time, and the serving time. The constraint (13) is relaxed. Typically, each parking area where vehicles begin the first trip is naturally connected to the nearest distribution center point. Thus, the saving list still must be build based on the relationship between customers and distribution centers. We compute the saving list using a modified formula in Çatay (2010) for MDVRPTW by the following equation:

$$M_{c,c'}^{dp} = (d_{dp,c}^* + d_{dp,c'}^* - d_{c,c'})/d_{c,c'}, \ \forall dp \in D^*, \forall c, c' \in C. \tag{25}$$

where $d_{dp,c}^*$ denotes the distance from distribution center dp to customer c, is computed by the formula $d_{dp,c}^* = \min_{p \in D^*}(d_{p,c} + d_{c,p}) - d_{dp,c}$ (the saving per unit distance traveled when a vehicle departs from distribution center dp, serves customer c, and comes back to the distribution center dp instead of the nearest distribution center for the next trip). The detail is shown in Algorithm A3 (see Appendix). A list of saving values is computed at line 2. If both two customers i and j are unscheduled, we consider routes starting at the nearest parking area from the distribution center dp (line 5). Customers i and j are inserted into a route by minimizing the objective function (lines 7–17). In addition, the upper-bound

capacity and the time window constraints must be checked at each step in the process (lines 13, 27 and 36). If one of two points is on a route, the unscheduled point is added to that route such that $nx(i) = j$ (lines 22–39). Lines 41–43 remove points violating constraint (13) from its route.

- **An Adapted Sweep Algorithm (vSW)**: The SW of Solomon (1987) is adapted for use in the MTDLC-VR problem. The adaptation is shown in Algorithm A4 (see the Appendix). In the clustering phase, each vehicle is scheduled to the nearest customer from a distribution center at lines 4–5, called a *seed* customer. To build a giant tour, we sequentially insert the unscheduled customers into the current route by selecting at each iteration the customer with the smallest angle formed by the ray from the distribution center through that customer and the ray from the distribution center through the seed customer (lines 6–21). Lines 8–19 explore feasible insertions. Line 14 splits a route into trips satisfying the constraint (12). Line 16 checks constraints where the constraint (13) is relaxed. Lines 17–18 select a customer that has the smallest angle to insert. We repeat exploration until all customers are served, or there is no feasible insertion. Lines 22–23 remove violated points from its route.
- **An Adapted Greedy Insertion Algorithm (vGIA)**: We devise the vGIA for the MTDLC-VR problem specifying in Algorithm A5 (see the Appendix). At a time point, the best customer is inserted into a route sequentially so that constraints are not violated, except for the constraint (13) (lines 5–14). A route is split into trips at line 8. Lines 10–12 mark the best customer and the best position to insert. Lines 22–23 remove violated points from its route.

4.4. An adapted ALNS with splitting procedure

The ALNS heuristic framework was first introduced in Ropke and Pisinger (2006). The main idea of the ALNS algorithm is repeatedly applying removal and insertion operators. However, the selection of operators at each iteration is not easy. Therefore, it is controlled by a roulette-wheel mechanism. Initially, each operator is equally weighted. These operators are dynamically selected according to their previous and present performance. As a matter of fact, the more (reasonable) neighborhoods the ALNS heuristic makes use of, the better performance it has (Ropke & Pisinger, 2006).

This section proposes an adapted ALNS algorithm (A-ALNS) for dealing with large MTDLC-VR problem instances. The A-ALNS algorithm uses eight removal operators and eight insertion operators. We propose six operators (R2, R5, I1, I2, I5, and I6) and chooses some appropriate operators in the literature by our experience. The appropriate operators are derived from Ghilas et al. (2016) and Ropke and Pisinger (2006) and are only slightly modified to fit our problem. The insertion operators use the splitting procedure to split a giant tour into feasible trips and ensure constraints (12)–(13). Due to the complexity of the problem, it is possible that no further improvements can be found after some iterations (namely *maxStable* number). In order to avoid getting stuck in local optima, one of our key ideas is to increase the quantity of eliminated customers. It means that our algorithm attempts to overcome plateaus by making a big jump. Besides, it can be seen that the roulette-wheel principle works well with a large number of iterations. However, for large instances, the processing time of each iteration is longer due to the large searching space. Therefore, we propose an adaptive score adjusting algorithm to quickly find a successful operator, where the score of operators is updated after each iteration and relates to the change in the value of the objective function.

4.4.1. Outline of A-ALNS algorithm

In Algorithm 1, some customers are removed from the current solution for each iteration and added to unscheduled customer list $\overline{R^*}$. The number of removed customer is randomly generated in $(\theta\eta, \gamma\eta]$ at line 4, where $0 < \theta < \gamma < 1$. After that, we re-insert the unscheduled customers into routes. If a new solution is better than the current one in *maxStable* iterations, the number of eliminated customers is maintained. Conversely, if the new solution is no better than the current solution after *maxStable* iterations, the number will be increased by a random variable that has a discrete uniform distribution in $[0, \theta\eta]$ (lines 27–28). Lines 12–17 update a locally optimal solution to be an input for the next iteration. Lines 18–21 indicate that the new solution is accepted. Lines 13, 19, 25 update the score of the selected operators. We note that the rate parameters are used to divide reward points into levels. We set $\pi_1 > \pi_2 > 0 > \pi_3$, where $\pi_3 < 0$ implies that there is a penalty cost. Lines 26–29 try to overcome plateaus by making a big jump.

1 An initial solution S_{init} is generated by an adapted construction heuristic;
2 Initialize probability for each removal operator and insertion operator;
3 $S_{best} \leftarrow S_{cur} \leftarrow S_{init}$;
4 A random number $\sigma \in [\theta\eta, \gamma\eta]$ is generated;
5 $it \leftarrow 0$;
6 **while** *stop-criterion not met* **do**
7 A removal operator O_R and an insertion operator O_I are randomly selected with a probability;
8 $\langle \mathcal{R}_*, S_{new} \rangle \leftarrow$ RmOpApplying(σ, O_R, S_{cur}); // see Section 4.4.3
9 $\overline{R^*} \leftarrow \overline{R^*} \cup \mathcal{R}_*$;
10 $\langle \overline{R^*}, S_{new} \rangle \leftarrow$ InsOpApplying$(\overline{R^*}, O_I, S_{new})$; // see Section 4.4.4
11 **if** $F(S_{new}) < F(S_{cur})$ **then**
12 **if** $F(S_{new}) < F(S_{best})$ **then**
13 UpdateOperatorScores$(O_R, O_I, S_{best}, S_{new}, \pi_1)$;
14 $S_{best} \leftarrow S_{new}$;
15 Generate a random number $\sigma \in [\theta\eta, \gamma\eta]$;
16 $it \leftarrow 0$;
17 **end**
18 **else**
19 UpdateOperatorScores$(O_R, O_I, S_{cur}, S_{new}, \pi_2)$;
20 $S_{cur} \leftarrow S_{new}$;
21 **end**
22 **end**
23 **else**
24 $\overline{R^*} \leftarrow \overline{R^*} \setminus \mathcal{R}_*$;
25 UpdateOperatorScores$(O_R, O_I, S_{new}, S_{cur}, \pi_3)$;
26 **if** $it++ > maxStable$ **then**
27 A random number $\sigma^* \in [0, \theta\eta]$ is generated;
28 $\sigma \leftarrow \sigma + \sigma^*$;
29 **end**
30 **end**
31 Update probabilities using Eq. (26); // see Section 4.4.2
32 **end**

Algorithm 1: ALNS$(\theta, \gamma, \pi_1, \pi_2, \pi_3)$

4.4.2. Choosing the operators

The alternating between different removal and insertion operators gives us a more robust heuristic overall (Ropke & Pisinger, 2006). A removal operator and an insertion operator are selected at each iteration. The weight of these operators is updated similarly.

We assume that there are ν operators, and the probability of the ith operator is denoted by w_i, $i \in \{1, \dots, \nu\}$. Initially, we assign $w_i = 1/\nu, i \in \{1, \dots, \nu\}$. After that, for each iteration, operators are chosen based on a roulette-wheel mechanism. We select the ith operator ($i = 1, \dots, \nu$) with probability $w_i / \sum_{i=1}^{\nu} w_i$.

In the implementation of Ropke and Pisinger (2006), the probability is now updated as:

$$w_i = w_i(1-\psi) + \psi\frac{\delta_i}{n_i}, \tag{26}$$

where n_i is the number of times that the ith operator was used during iterations, $\psi \in [0,1]$ is the roulette wheel parameter, δ_i is the score of the ith operator (the initial value is equal to 1). The score of the selected operators are updated by Algorithm 2. In Algorithm 2, scores of operators are updated with the change rate of objective values of two solutions s_1 and s_2 (lines 2, 4). Our proposed method to compute the benefit and penalty with regard to the objective function is more obvious than methods proposed by Ghilas et al. (2016) and Ropke and Pisinger (2006). More precisely, the scores of operators depend on the improvement percentage of the objective values.

1 $\delta_R \leftarrow$ the score of O_R;
2 $\delta_R \leftarrow \delta_R + \pi^*(\frac{g_r(s_1)-g_r(s_2)}{g_r(s_2)} + \frac{g_v(s_1)-g_v(s_2)}{g_v(s_2)} + \frac{g_c(s_1)-g_c(s_2)}{g_c(s_2)})$;
3 $\delta_I \leftarrow$ the score of O_I;
4 $\delta_I \leftarrow \delta_I + \pi^*(\frac{g_r(s_1)-g_r(s_2)}{g_r(s_2)} + \frac{g_v(s_1)-g_v(s_2)}{g_v(s_2)} + \frac{g_c(s_1)-g_c(s_2)}{g_c(s_2)})$;

Algorithm 2: UpdateOperatorScores($O_R, O_I, s_1, s_2, \pi^*$)

4.4.3. Removal operators

Eight removal operators are used in our A-ALNS algorithm. They are adapted from or inspired by Nguyen et al. (2019), Ropke and Pisinger (2006) and Shaw (1998). Algorithm 3 depicts the procedure for removing customers. In algorithm 3, there are σ requests removed from its current route (lines 2–6). The set $\mathcal{R}_*$ contains these removed requests (line 5). The procedure returns a new solution and a set of removed requests (line 7).

1 $\mathcal{R}_* \leftarrow \emptyset$;
2 **for** $i = 0, \ldots, \sigma - 1$ **do**
3 Apply the removal operator O_R to find a scheduled customer $c \in C$;
4 $S_{cur} \leftarrow$ Remove customer c from the current solution S_{cur};
5 $\mathcal{R}_* \leftarrow \mathcal{R}_* \cup \{c\}$
6 **end**
7 **return** $\langle \mathcal{R}_*, S_{cur} \rangle$;

Algorithm 3: RmOpApplying(σ, O_R, S_{cur})

The description of these operators is shown below.

- **Random removal (R1)**: This operator randomly removes σ customers from the current solution.
- **Random trip removal (R2)**: This operator removes a full trip from the current solution. This trip is randomly selected from the scheduled trips. The operator is repetitively applying selection and removal until the number of removed points is greater than σ.
- **Worst removal (R3)**: This operator repeats the removing the customer with the highest cost until the number of removed customers is equal to σ. The main idea of the worst removal operator is to remove a number of high-cost points in which the cost of a point $c \in C$ is defined as follows:
  Given a solution s, where point c is served by a vehicle (i.e., $ix(c) \neq \perp$). A solution s' is obtained by removing point c from solution s. We denote the cost of point c by $cost(c) = g_c(s) - g_c(s')$.
- **Shaw removal (R4)** This operator was introduced by Shaw (1997). The goal of this operator is to remove a set of customers that are related in a predetermined way and hence are easy to interchange. The similarity of two customers $c_1 \in C$ and $c_2 \in C$ as a relatedness measure is denoted by $SR(c_1, c_2)$ that is determined as follows:

  $$SR(c_1, c_2) = \eta_1 t_{c_1,c_2} + \eta_2 |ST(c_1) - ST(c_2)| + \eta_3 b_{c_1 c_2}$$

  where η_1, η_2, η_3 are the weight parameters, $b_{c_1 c_2}$ is equal to -1 if point c_1 and point c_2 are on the same route, or equal to 1 otherwise.
- **Forbidden-removal**: We adapt operators **R1–R4** by selecting non-forbidden points and obtain the corresponding operators **R5–R8**. This selection keeps a record of the removal counts for the previous iterations. If the removal count of point $c \in C$ is bigger than a predefined number τ, point c is forbidden for selection. These operators R5–R8 only choose the non-forbidden points to remove for each iteration.

4.4.4. Insertion operators

Eight insertion operators are used in our proposed A-ALNS heuristic algorithm. These operators try to insert customers from the removal list into the existing routes. These operators are to fix a partly destroyed solution and to reduce the number of unserved customers. Algorithm 4 specifies how to use these operators. If a customer can be inserted into a route at line 2, the SP procedure splits the route into feasible trips (line 5) and the customer is removed from the set of unscheduled customers (line 7).

1 **foreach** *unscheduled customer* $c \in \overline{R^*}$ **do**
2 Apply operator O_I to insert point c into a feasible position;
3 **if** $ix(c) \neq \perp$ **then**
4 $k \leftarrow ix(c)$;
5 $s^* \leftarrow SP(s^k)$; // see Section 4.2.3
6 $S_{cur} \leftarrow \mathcal{R}(S_{cur}, s^k, s^*)$;
7 $\overline{R^*} \leftarrow \overline{R^*} \setminus \{c\}$;
8 **end**
9 **end**
10 **return** $\langle \overline{R^*}, S_{cur} \rangle$;

Algorithm 4: InsOpApplying($\overline{R^*}, O_I, S_{cur}$)

The description of these operators is shown below.

- **vGIA (I1)**: This operator repeatedly inserts an unscheduled customer into the best possible position on the best route.
- **vGIA with a noise (I2)**: This operator is similar to the operator I1. However, a new customer is inserted into the best position with a probability. It implies that the operator allows a flexible insertion with noise.
- **First possible insertion (I3)**: the unscheduled point is inserted into the first possible position.
- **Regret-H insertion (I4)**: The difference between the profit of inserting a point into the best position and the hth best position ($h = 2, \ldots, H$) is called *regret value*. This operator computes this difference as a type of look-ahead information when opting for the point to insert. The regret heuristic algorithm has been effectively utilized by Nguyen et al. (2019), Potvin and Rousseau (1993). The regret value of a customer $c \in C$ is computed as follows:

  $$reg_h(c) = \sum_{h=2}^{H} |cost_h(c) - cost_1(c)|$$

  where, $cost_h(c)$ is the cost of inserting customer c into the hth best position. This operator inserts customer c which has the maximum regret value into the best position. In this paper, we use $H = 2$.
- **Sorting-insertion**: We see that the latest allowed arrival time at a customer c minus the earliest allowed arrival time at that customer (i.e., $l(c) - e(c)$) equals the allowed serving duration of customer c. A customer with short allowed serving duration

Table 4
The detail of the found optimal solutions.

Ins	g_r	g_v	#trips	g_c	Optimal solution		
					g_r	g_v	g_c
C101.25	0	3022.5	3	191.3	0	3022.5	191.3
C101.50	0	14250	5	362.4	0	14250	362.4
C101.100	0	96100	10	827.3	0	96100	827.3
C102.25	0	3022.5	3	190.3	0	3022.5	190.3
C102.50	0	14250	5	361.4	0	14250	361.4
C102.100	0	96100	10	827.3	0	96100	827.3
C201.25	0	2665	2	214.7	0	2665	214.7
C201.50	0	10605	3	360.2	0	10605	360.2
C201.100	0	28830	3	589.1	0	28830	589.1
C202.25	0	2665	2	214.7	0	2665	214.7
C202.50	0	10605	3	360.2	0	10605	360.2
C202.100	0	28830	3	589.1	0	28830	589.1
R101.25	0	13880	8	617.1	0	13880	617.1
R101.50	0	50040	12	1044.0	0	50040	1044.0
R101.100	0	183600	20	1044.0	0	183600	1044.0
R201.25	0	6940	4	463.3	0	6940	463.3
R201.50	0	25020	6	791.9	0	25020	791.9
R201.100	0	73440	8	1143.2	0	73440	1143.2
RC101.25	0	6940	4	461.1	0	6940	461.1
RC101.50	0	13880	8	944	0	6940	944
RC101.100	0	144150	15	1619.8	0	144150	1619.8

Table 5
Comparison between MILP model and the A-ALNS algorithm (t $\leq$ 2 hours).

Ins	MILP model				A-ALNS			
	g_r	g_v	g_c	t	g_r	g_v	g_c	t
RG-1-2-2-2-6	0	1200	145	0.57	**0**	**1200**	**145**	3.69
RG-2-2-2-2-6	0	1200	118	0.34	**0**	**1200**	**118**	0.89
RG-1-2-2-3-8	0	1800	169	4325.14	**0**	**1800**	**169**	4.12
RG-2-2-2-3-8	0	1800	154	4259.2	**0**	**1800**	**154**	5.08
E_{21}-1-2-4-6-5	0	20610	9982	0.84	**0**	**20610**	**9982**	4.28
E_{21}-2-2-4-6-5	0	20610	9657	1.58	**0**	**20610**	**9657**	4.75
E_{21}-1-2-2-2-25	0	200382	21682	6982.5	**0**	**200382**	**21682**	14.34

Table 6
Comparison between MILP model and the A-ALNS algorithm (t > 2 hours).

Ins	MILP model				A-ALNS			
	g_r	g_v	g_c	t	g_r	g_v	g_c	t
RG-1-2-4-4-25	0	2400	402	7521.64	**0**	**2400**	**402**	17.03
RG-2-2-4-4-25	0	2400	391	7681.02	**0**	**2400**	**391**	19.89
RG-1-2-4-4-50	0	2400	725	13724.62	**0**	**2400**	**725**	402.12
RG-2-2-4-4-50	0	2400	697	14018.26	**0**	**2400**	**697**	544.1
E_{21}-2-2-2-2-25	0	197950	20793	7282.02	**0**	**197950**	**20793**	15.21
E_{21}-1-2-2-4-50	0	401256	38017	11021.72	**0**	**401256**	**38017**	388.62
E_{21}-2-2-2-4-50	0	401256	37882	12018.88	**0**	**401256**	**37882**	412.44
E_{21}-1-2-2-4-75	0	426272	58544	67472.13	**0**	**426272**	**58544**	1214.75
E_{21}-2-2-2-4-75	0	426272	58007	75721.47	**0**	**426272**	**58007**	1493.2

should be inserted first. Therefore, we denote the variant of operators **I1–I4** by **I5–I8**, where the unscheduled customer list is sorted by allowed serving time (short to long) before applying the insertion operators (**I1–I4**)

5. Experiments

In this section, we investigate the performance of the proposed algorithm by numerical examples. Algorithms are implemented in Java programming language. In our experiments, a computer with a Core(TM) i7-4790 CPU @ 3.660 GHz processor and 16 GB of RAM is used. For the convenience of reference, some parameters used for the MILP model and experimental results are outlined in Appendix. In our experiments, we focus on four main experiments.

5.1. Instances and setting

As far as we know, the generic problem described in this paper considers some real-life constraints that have not been reported in the literature. Thus, no former research and available data sets are found for comparison. Besides, one major reason is that it is not easy to obtain suitable benchmark instances in the literature. Therefore, to evaluate the algorithm's efficiency and its practical applicability, we experiment on several data sets from different sources described as follows.

To validate the proposed MILP model and evaluate the efficiency of the proposed heuristic algorithms, we generate some small-size problem instances that represent complete cases for the MTDLC-VR problem with symmetric distance matrix, denoted by RG-x-y-z-t-w. The notation x is either 1 or 2, where type 1 has tighter time windows than instances of type 2. The notation y represents the number of parking areas. The number of distribution centers is denoted by z. The notation t presents the number of vehicles, and w is the number of customers. The locations of distribution centers and parking areas are randomly generated in a two-dimensional coordinate area where the customers are located at. The distance from one point to another point is the Euclidean distance. Each vehicle is scheduled for a maximum of three trips per day.

As pointed out in the introduction section, our problem combines the existing problems in the literature, including CVRP, VRPTW, MTVRP, MDVRP problems, and extends them by adding real factors. Therefore, the benchmark instances representing CVRP and VRPTW problems such as the instances provided by Solomon (1987) can be modified to fit our problem (as a specific case). In our experiments, 21 benchmark instances in Solomon (1987) are used. These instances included routing and scheduling environments that differed in the type of data used to generate the percentage of customers, vehicle capacity, the tightness of time windows, and customer positions. These instances can be seen as a small case of the MTDLC-VR problem, where $|PK| = 1, |D| = 1, t_{wait}(dp) = 0, t_{unit}(dp) = 0$, $t_{unit}(c) = 0$, and $t_{wait}(c)$ is equal to service time at customer c, $\forall c \in C, \forall dp \in D$.

To validate the MILP model on real data sets, we perform on small problem instances extracted from a real data set of a big milk distribution company in Vietnam. Each instance is denoted by E_d-x-y-z-t-w, where d denotes the release date. In these instances, each vehicle is scheduled for a maximum of two trips per day. The distance from one point to another point is provided by Google Map API services. The detailed descriptions are presented by Tables A.1, A.2, A.3, A.4 in Appendix.

To evaluate the efficiency of the proposed A-ALNS algorithm and its practical applicability, we conduct numerical experiments and sensitivity analysis on real data sets of a big milk distribution company in Vietnam. There are 12 data sets collected from 2019-09-09 to 2019-09-21. Each data set is denoted by E_d, where d is the release date. There are two parking areas and four distribution centers. Distribution centers are evenly distributed on the four cardinal points. A parking area is usually located in an area not too far from some distribution centers to reach these centers in a short time. On each day, vehicles work from 7 a.m. to 5 p.m. The maximum number of trips is five for each vehicle. The max, min, and the average number of requests in a day are 12580, 2956, and 8938, respectively. The total weight of products ordered by a customer can be up to 16.1 tons. Customers are mainly located in the city center. The maximum distance from a distribution center to a customer is 15.6 km.

5.2. Experiment 1: Mathematical formulation validation

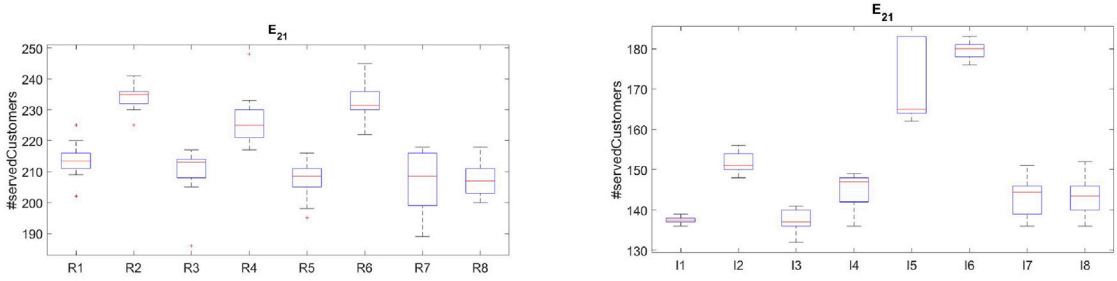
- *Description*: The first problem is to validate and analyze the performance of our MILP model for the MTDLC-VR problem.

Table 7
The efficiency comparison between construction algorithms.

Ins	#cus	vSA				vSW				vGIA				ρ
		g_r	g_v	g_c	t	g_r	g_v	g_c	t	g_r	g_v	g_c	t	(%)
E_{10}	842	48	82	1271551	**21.05**	54	87	1552524	5982.55	**35**	86	1454483	78.43	95.50
E_{11}	1149	**101**	103	1735808	**39.42**	149	104	1855500	3038.09	181	102	1757928	178.44	91.21
E_{12}	937	**141**	75	1287308	**33.57**	157	82	1463081	1156.01	157	81	1418066	89.75	84.95
E_{13}	954	**54**	89	1511288	**30.86**	198	84	1439062	934.56	152	86	1491953	86.17	94.34
E_{14}	936	96	100	1450418	**52.84**	104	102	1617522	1201.59	**70**	104	1615203	94.87	92.52
E_{15}	531	101	53	737811	**7.14**	103	57	903030	226.69	**100**	56	855863	28.39	81.17
E_{16}	785	**46**	75	1189171	**21.49**	76	84	1449256	3051.73	56	83	1382954	72.65	94.14
E_{17}	1256	133	83	1728323	**45.06**	**115**	89	1956719	6159.97	165	85	1820635	166.77	90.84
E_{18}	718	**35**	78	1196277	**26.62**	142	80	1188847	357.99	175	78	1110206	52.29	95.13
E_{19}	988	**87**	89	1518201	**35.64**	187	88	1533257	1582.18	269	86	1430748	93.72	91.19
E_{20}	814	125	52	994307	**17.00**	115	61	1256651	5103.44	**105**	57	1106481	77.65	87.10
E_{21}	267	134	14	258995	**1.99**	112	15	269870	81.98	**74**	17	309269	16.67	72.28

Table 8
The results of the A-ALNS algorithm.

Ins	#cus	A-ALNS								F_c				LS			
		min_g_r	max_g_r	avg_g_r	std_g_r	ρ_1	ρ_2	avg_g_v	avg_g_c	g_r	g_v	g_c	ρ_3	g_r	g_v	g_c	ρ_4
E_{10}	842	0	14	4	5.13	99.52	1.66	85	1863268	35	86	1454483	88.57	15	85	1573627	73.33
E_{11}	1149	42	87	67	16.98	94.17	3.92	104	2130847	101	103	1735808	33.66	87	103	1813869	22.98
E_{12}	937	15	93	46	25.51	95.09	8.32	77	1718317	141	75	1287308	67.38	108	76	1485266	57.41
E_{13}	954	22	64	37	14.3	96.12	4.4	92	1763982	54	89	1511288	31.48	48	91	1602029	22.91
E_{14}	936	22	64	37	14.3	96.05	4.49	92	1763982	70	104	1615203	47.14	63	95	1673202	41.27
E_{15}	531	6	38	19	11.78	96.42	6.03	57	1076136	100	56	855863	81	42	57	928225	54.76
E_{16}	785	9	16	12	2.19	98.47	0.89	83	1740164	46	75	1189171	73.91	25	78	1342388	52
E_{17}	1256	40	108	72	26.11	94.27	5.41	92	2225514	115	89	1956719	37.39	103	92	2062217	30.09
E_{18}	718	14	32	22	6.19	96.94	2.51	75	1452254	35	78	1196277	37.14	28	78	1338542	21.43
E_{19}	988	24	61	45	15.22	95.45	3.74	79	1774229	87	89	1518201	48.28	71	85	1557343	36.63
E_{20}	814	2	74	9	21.51	98.89	8.85	63	1563752	105	57	1106481	91.43	70	60	1321475	69.87
E_{21}	267	19	48	41	7.84	84.64	10.86	18	422084	74	17	309269	44.59	46	18	342724	10.87



(a) Removal operators (b) Insertion operators

Fig. 3. The efficiency of operators.

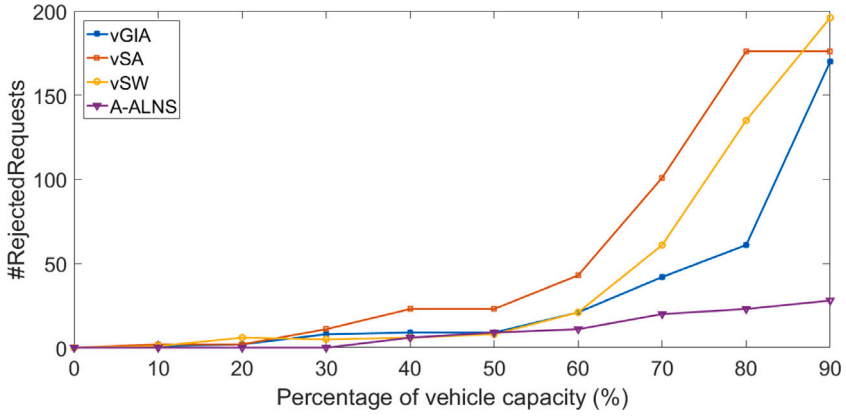


Fig. 4. The number of requests removed from the solution for violating the lower-bound capacity constraints.

- *Data*: Three data sets of different origins are used for this experiment. There are eight generated instances, eight small real instances, and 21 benchmark data sets of Solomon (1987) (followed the author's naming convention) are used. By selecting

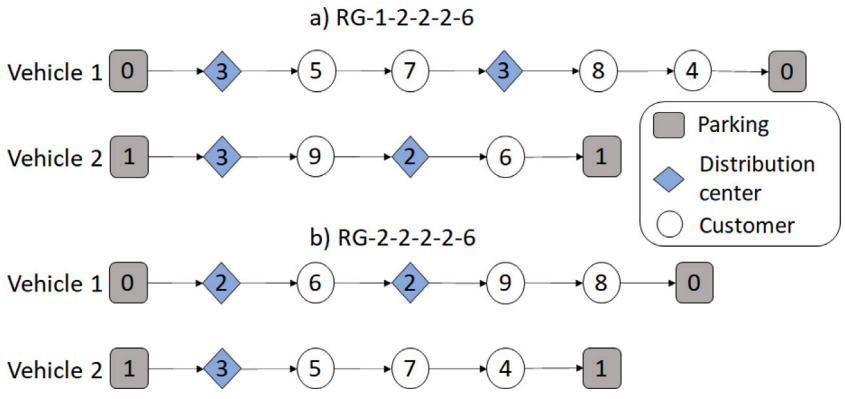


Fig. A.1. Results of solving the MIP model on random generated instances.

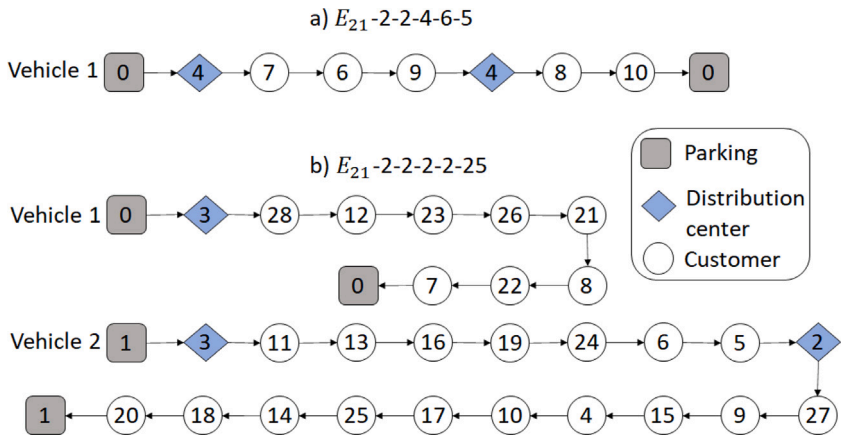


Fig. A.2. Results of solving the MIP model on real small instances.

Table A.1
Parameters of instance E_{21}-1-2-4-6-5.

Denotation	E_{21}-1-2-4-6-5
P	$\{p_0, p_1, p_2\}$
$w(p_i)$	1 $(i = 0, 1, 2)$
PK	$\{0, 1\}$
$e(i), l(i)$	25200, 61200 $(i = 0, 1)$
D	$\{2, 3, 4, 5\}$
$e(i), l(i), t_{wait}(i), t_{unit}(i)$	25200, 61200, 1800, 1.8 $(i = 2, 3, 4, 5)$
C	$\{6, 7, 8, 9, 10\}$
$e(i), l(i), t_{wait}(i), t_{unit}(i)$	30800, 36500, 900, 4.2 $(i = 6)$
	28800, 33400, 900, 4.2 $(i = 7)$
	44600, 49800, 900, 4.2 $(i = 8)$
	36200, 40500, 900, 4.2 $(i = 9)$
	48200, 52400, 900, 4.2 $(i = 10)$
$dm(i, p_j)$	123, 56, 53, 153, 231 $(\forall i = 6, \ldots, 10, j = 0)$
	123, 56, 53, 153, 231 $(\forall i = 6, \ldots, 10, j = 1)$
	123, 56, 53, 153, 231 $(\forall i = 6, \ldots, 10, j = 2)$
K	$\{k_0, k_1, k_2, k_3, k_4, k_5\}$
$p(k_i), \underline{c}(k_i), \overline{c}(k_i), q(k), f_{k_i}$	0, 780, 1430, 5, 1.2 (i = 0)
	0, 1380, 2530, 5, 1.2 (i = 1)
	0, 780, 1430, 5, 1 (i = 2)
	1, 780, 1430, 5, 1.1 (i = 3)
	1, 8580, 15730, 5, 1 (i = 4)
	1, 1380, 2530, 5, 1.2 (i = 5)
$rp(k, p)$	1 $\forall k \in K, \forall p \in P$
$rv(k, c)$	1 $\forall k \in K, \forall c \in C$
$vc(k, c)$	0 $\forall k \in K, \forall c \in C$

instances where customers can be served, $\hat{f} = 6 \times \eta \times \max_{(i,j)\in E} d_{i,j}$, $f_k = \eta \times \max_{(i,j)\in E} d_{i,j}$ (if vehicle k is an in-house vehicle) and $f_k = 2 \times \eta \times \max_{(i,j)\in E} d_{i,j}$ (if vehicle k is an outsourced vehicle) $(\forall k \in K)$, it is possible to compare solutions of the MILP model with solutions of the proposed heuristic algorithms.

Table A.2
Travel time matrix of instance E_{21}-1-2-4-6-5.

Point	0	1	2	3	4	5	6	7	8	9	10
0	0	245	2829	1175	1222	1216	1441	743	350	800	1256
1	280	0	3017	1364	1275	1405	1243	756	461	750	1107
2	2845	2998	0	1791	3502	1843	4087	3251	2983	3103	3577
3	1198	1351	1802	0	2210	151	2546	1885	1450	1476	1950
4	1195	1211	3508	2145	0	2137	1137	631	872	1950	2306
5	1247	1400	1853	151	2201	0	2595	1933	1498	1557	2030
6	1456	1265	4122	2547	1169	2588	0	979	1393	1593	1713
7	727	720	3222	1830	641	1822	981	0	436	1455	1812
8	322	392	3008	1449	871	1490	1393	436	0	1104	1487
9	824	743	3061	1434	1964	1515	1641	1472	1104	0	571
10	1232	1056	3522	1895	2277	1976	1700	1785	1455	508	0

We solve our MILP model by GUROBI Optimizer. Figs. A.1 and A.2 in the Appendix describe the found optimal solutions of two randomly generated data sets and two real small data sets, respectively. For visualization, we plot the optimal solution of the real instance E_{21}-1-2-4-6-5 on the Google map platform presenting in Fig. A.3. For Solomon's benchmark data sets, we compare our results with the best-known solutions summarized on Solomon's web page http://web.cba.neu.edu/~msolomon/problems.htm. We found that the solutions found by our model are equivalent to the best solution of Solomon's benchmark data sets in the literature. To demonstrate our results, Tables 4 and 5 show the optimal solution found and the computational time (t). These results show that our formulation is validated. Table 5 states that the A-ALNS and MILP model results are the same. In addition, these experiments

Table A.3
Parameters of instances *RG*-1-2-2-2-6 and *RG*-2-2-2-2-6.

Denotation	*RG*-1-2-2-2-6	*RG*-2-2-2-2-6
P	$\{p_0, p_1\}$	$\{p_0, p_1\}$
$w(p_i)$	1 (i=0), 2 (i = 1)	1 (i=0), 2 (i = 1)
PK	$\{0, 1\}$	$\{0, 1\}$
$e(i), l(i)$	0, 200 (i = 0, 1)	0, 200 (i = 0, 1)
D	$\{2, 3\}$	$\{2, 3\}$
$e(i), l(i), t_{wait}(i), t_{unit}(i)$	0, 200, 1, 1 (i = 2, 3)	0, 200, 1, 1 (i = 2, 3)
C	$\{4, 5, 6, 7, 8, 9\}$	$\{4, 5, 6, 7, 8, 9\}$
$e(i), l(i), t_{wait}(i), t_{unit}(i)$	60, 120, 1, 1 (i = 4)	0, 200, 1, 1 (i = 4)
	15, 90, 1, 1 (i = 5)	0, 200, 1, 1 (i = 5)
	130, 180, 1, 1 (i = 6)	0, 200, 1, 1 (i = 6)
	5, 50, 1, 1 (i = 7)	0, 200, 1, 1 (i = 7)
	30, 100, 1, 1 (i = 8)	0, 200, 1, 1 (i = 8)
	80, 130, 1, 1 (i = 9)	0, 200, 1, 1 (i = 9)
$dm(i, p_j)$	3, 5, 2, 0, 0, 10 ($\forall i = 4, \ldots, 9, j = 0$)	3, 5, 2, 0, 0, 10 ($\forall i = 4, \ldots, 9, j = 0$)
	2, 5, 3, 2, 4, 5 ($\forall i = 4, \ldots, 9, j = 1$)	2, 5, 3, 2, 4, 5 ($\forall i = 4, \ldots, 9, j = 1$)
K	$\{k_0, k_1\}$	$\{k_0, k_1\}$
$p(k_i), \underline{c}(k_i), \bar{c}(k_i), q(k), f_{k_i}$	0, 2, 40, 2, 1 (i = 0)	0, 2, 40, 2, 1 (i = 0)
	1, 3, 40, 2, 1 (i = 1)	1, 3, 40, 2, 1 (i = 1)
$rp(k_i, p_j)$	1 $\forall k_i \in K, \forall p_j \in P$	1 $\forall k_i \in K, \forall p_j \in P$
$rv(k_i, c_j)$	$rv(k_0, 6) = 0$, 1 otherwise	1 $\forall k_i \in K, \forall c_j \in C$
$vc(k_j, c_j)$	0 $\forall k_i \in K, \forall c_j \in C$	$rm(k_0, 6) = 1$, 0 otherwise

Table A.4
Travel time matrix of instances *RG*-1-2-2-2-6 and *RG*-2-2-2-2-6.

Point	0	1	2	3	4	5	6	7	8	9
0	0	3	3	6	10	4	19	15	2	10
1	3	0	93	77	8	3	4	2	9	12
2	3	93	0	5	10	12	2	5	33	6
3	6	77	5	0	9	3	99	7	9	2
4	10	8	10	9	0	12	89	3	14	15
5	4	3	21	3	12	0	95	2	33	24
6	19	4	2	99	89	95	0	100	93	90
7	15	2	5	7	2	16	100	0	33	4
8	2	9	33	9	17	27	93	12	0	8
9	10	12	6	2	31	25	90	20	8	0

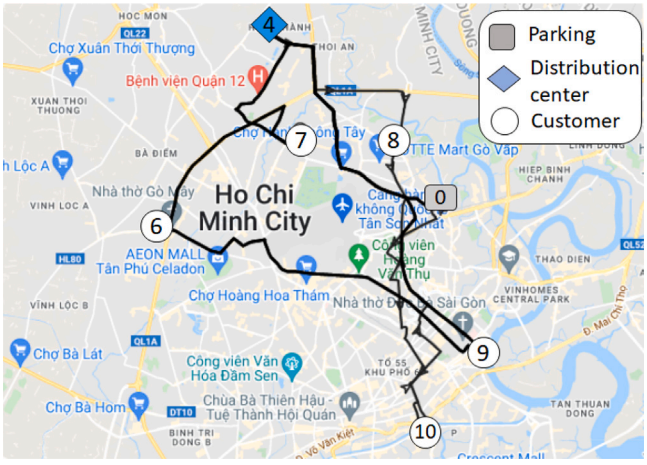


Fig. A.3. Solution visualization of instance E_{21}-1-2-4-6-5.

indicate that our proposed algorithm can find optimal solutions in small instances very fast compared to the MILP solver. Moreover, the GUROBI Optimizer cannot solve the MILP model for instances having more than 25 customers within two hours (for instances that take more than two hours to solve, the results are shown in Table 6). The main reason is that the MILP model uses many auxiliary variables to linearize the constraints. As a result, the MILP model is not practical for large problems (it does not scale if the number of variables increases), primarily for computational time reasons. It is also easy to see that the construction algorithms do not give results close to the optimal results (see Table A.5 in the Appendix).

5.3. Experiment 2: Comparison the efficiency between construction heuristics

- *Description*: In the second experiment, we compare the efficiency between proposed construction heuristics.
- *Data*: 12 real data sets ($E_{10} - E_{21}$) from one of the biggest milk distribution companies are used.

We conduct the second experiment by comparing the objective of the three variants of traditional construction algorithms on 12 real instances, as listed in Table 7. In each variant of the construction algorithm, the number of unserved customers (g_r), the number of vehicles needed (g_v, where c_k equals to 1 if vehicle k is an in-house vehicle; otherwise, equals to 1.1), the total travel distance (g_c), and the running time (t) are reported. The best-achieved solution for each instance is shown in boldface. The last column of the table presents $\rho = (\#cus - \min g_r)100/\#cus$ which is the percentage of the served customers of the best solution found by (among) three algorithms vSA, vSW and vGIA. The table shows that the proposed algorithms vSA and vGIA generate effective solutions at most instances in a short computation time. At the initialization step, the results indicate that about 72.28% to 95.84% of total customers are served over all instances. The best solution is found in less than 100 seconds in most instances.

5.4. Experiment 3: The efficiency of the A-ALNS algorithm

- *Description*: We implement the A-ALNS algorithm for solving the large instances of the MTDLC-VR problem. Also, the robustness of the A-ALNS strategy is analyzed.
- *Data*: Twelve real data sets from one of the biggest milk distribution companies are used.

5.4.1. Parameter tuning

Five sensitive parameters of the A-ALNS algorithm were tuned for the given combination of parameters as shown in Appendix (Table A.6). Some reasonable values for each parameter are given from our experience. The best-achieved values are shown in bold-face ($\theta = 0.2$, $\gamma = 0.3$, $\pi_1 = 10$, $\pi_2 = 0$, $\pi_3 = -10$). Besides, several parameters used in the A-ALNS algorithm have the setting as determined in Ropke and Pisinger (2006) as $T_{init} = 200, \eta_1 = 0.5, \eta_2 = 0.2, \eta_3 = 0.1, \tau = 0.1\eta$. The other parameters are selected based on some other research and our experience.

Table A.5
Comparison between MIP model and construction algorithms.

Ins	MIP model				vGIA				vSA				vSW			
	g_r	g_v	g_c	t	g_r	g_v	g_c	t	g_r	g_v	g_c	t	g_r	g_v	g_c	t
RG-1-2-2-2-6	0	1200	145	0.574	0	1200	227	0.014	0	1200	208	0.018	3	1200	130	0.016
RG-2-2-2-2-6	0	1200	118	0.34	3	1200	117	0.009	4	600	26	0.03	2	1200	153	0.017
RG-1-2-2-3-8	0	1800	169	4325.14	2	1200	132	0.06	1	1200	147	0.073	2	1200	132	0.007
RG-2-2-2-3-8	0	1800	154	4259.2	2	1200	132	0.01	2	1200	132	0.024	2	1200	132	0.011
E_{21}-1-2-4-6-5	0	20610	9982	0.841	2	20610	5251	0.018	2	20610	5223	0.022	2	20610	5251	0.013
E_{21}-2-2-4-6-5	0	20610	9657	1.58	1	20610	8224	0.012	2	20610	7683	0.019	2	20610	7683	0.103
E_{21}-1-2-2-2-25	0	200382	21682	6982.5	0	200382	25367	0.17	0	21124	22458	0.245	0	200382	22458	0.32
E_{21}-2-2-2-2-25	0	197950	20793	7282.02	0	197950	23890	0.081	0	197950	30231	0.156	0	197950	31441	0.25

Table A.6
Results of parameter tuning.

Iters	θ	γ	π_1	π_2	π_3	g_c	g_r	g_v	θ	γ	π_1	π_2	π_3	g_c	g_r	g_v
10000	0.1	0.3	3	0	−1	445485	45	19	0.2	0.3	3	0	−1	420380	39	18
10000	0.1	0.3	3	0	−3	440172	38	19	0.2	0.3	3	0	−3	423817	47	18
10000	0.1	0.3	3	0	−10	379908	50	18	0.2	0.3	3	0	−10	421053	40	19
10000	0.1	0.3	3	1	−1	459498	21	20	0.2	0.3	3	1	−1	389877	49	19
10000	0.1	0.3	3	1	−3	455354	29	20	0.2	0.3	3	1	−3	429510	30	19
10000	0.1	0.3	3	1	−10	425414	45	19	0.2	0.3	3	1	−10	418997	41	19
10000	0.1	0.3	5	0	−1	417409	41	19	0.2	0.3	5	0	−1	410687	46	19
10000	0.1	0.3	5	0	−3	406200	50	19	0.2	0.3	5	0	−3	421139	47	19
10000	0.1	0.3	5	0	−10	432016	36	19	0.2	0.3	5	0	−10	407769	46	19
10000	0.1	0.3	5	1	−1	460105	32	20	0.2	0.3	5	1	−1	426735	42	19
10000	0.1	0.3	5	1	−3	437340	39	19	0.2	0.3	5	1	−3	445484	25	19
10000	0.1	0.3	5	1	−10	424521	38	19	0.2	0.3	5	1	−10	421118	39	19
10000	0.1	0.3	10	0	−1	422010	26	20	0.2	0.3	10	0	−1	470880	22	20
10000	0.1	0.3	10	0	−3	421454	36	19	0.2	0.3	10	0	−3	416284	47	19
10000	0.1	0.3	10	0	−10	421531	40	20	**0.2**	**0.3**	**10**	**0**	**−10**	**452481**	**14**	**20**
10000	0.1	0.3	10	1	−1	404890	38	19	0.2	0.3	10	1	−1	432293	36	19
10000	0.1	0.3	10	1	−3	410203	38	19	0.2	0.3	10	1	−3	430403	40	19
10000	0.1	0.3	10	1	−10	430634	43	19	0.2	0.3	10	1	−10	403562	49	18
10000	0.1	0.4	3	0	−1	416768	46	19	0.2	0.4	3	0	−1	415177	39	19
10000	0.1	0.4	3	0	−3	421942	45	19	0.2	0.4	3	0	−3	468029	21	20
10000	0.1	0.4	3	0	−10	422814	29	19	0.2	0.4	3	0	−10	435915	40	19
10000	0.1	0.4	3	1	−1	430364	40	19	0.2	0.4	3	1	−1	426893	35	19
10000	0.1	0.4	3	1	−3	427418	46	19	0.2	0.4	3	1	−3	444074	29	20
10000	0.1	0.4	3	1	−10	412479	33	19	0.2	0.4	3	1	−10	440920	31	19
10000	0.1	0.4	5	0	−1	394850	43	18	0.2	0.4	5	0	−1	417227	44	19
10000	0.1	0.4	5	0	−3	454826	48	19	0.2	0.4	5	0	−3	414661	39	19
10000	0.1	0.4	5	0	−10	433225	43	19	0.2	0.4	5	0	−10	390104	44	18
10000	0.1	0.4	5	1	−1	433284	46	19	0.2	0.4	5	1	−1	412700	31	19
10000	0.1	0.4	5	1	−3	430165	24	19	0.2	0.4	5	1	−3	425162	30	19
10000	0.1	0.4	5	1	−10	439430	42	19	0.2	0.4	5	1	−10	442049	29	20
10000	0.1	0.4	10	0	−1	418130	35	19	0.2	0.4	10	0	−1	379094	51	18
10000	0.1	0.4	10	0	−3	370640	51	18	0.2	0.4	10	0	−3	427600	44	19
10000	0.1	0.4	10	0	−10	428677	33	19	0.2	0.4	10	0	−10	431843	38	19
10000	0.1	0.4	10	1	−1	402830	48	19	0.2	0.4	10	1	−1	458796	25	20
10000	0.1	0.4	10	1	−3	454653	30	20	0.2	0.4	10	1	−3	459346	27	20
10000	0.1	0.4	10	1	−10	441160	17	20	0.2	0.4	10	1	−10	453280	22	20

5.4.2. The efficiency of removal and insertion operators

This experiment aims to evaluate the efficiency of both removal operators and insertion operators used in the A-ALNS algorithm. Each operator is used independently to evaluate its efficiency. The minimum, maximum and average number of served customers are determined after running ten times. Fig. 3(a) displays the distribution of the number of served customers. Eight box plots correspond to the result of eight removal operators. We found that operators R1 and R2 have less dispersion than others. Operator R2 and its adaptation R6 obtain the best result. The result of R4 is nearly a normal distribution. Operators using forbidden records (i.e., R5, R6, R7, R8) have worse and more scattered data. Moreover, these operators have outliers within the results. They illustrate the effects of the forbidden record in different scenarios.

Second, insertion operators are also evaluated in the same way. Fig. 3(b) specifies that operators I1, I2 and I6 have small dispersion. Although their results are not symmetric, they have small standard deviations. Besides, the evaluation also presents that the independent use of operators is not effective due to the limitation of searching space.

5.4.3. Robustness of the A-ALNS strategy

In this section, we conduct experiments to assess the robustness of the A-ALNS strategy. A robustness strategy could provide solutions with great precision fast and stably. We implement the proposed A-ALNS algorithm with the JAVA language programming for solving the large instances of the MTDLC-VR problem. The effective construction vGIA algorithm is used for generating the initial solution. After 10000 iterations or 2 h of an execution (the time limit for planning in actual operation that we have surveyed), the best solution is reported. The average values reported for the objective function and runtime of the A-ALNS are determined after running 10 times. The results of 12 realistic instances for 267 to 1256 customers are listed as shown in Table 8. The percentage of the number of served customers is denoted by $\rho_1 = (\#cus - avg\_g_r)100/\#cus$. The dispersion of the results of different runs is presented by the standard deviation std_g_r and $\rho_2 = (max\_g_r - min\_g_r)100/\#cus$. It is not possible to have solutions made by the company for the comparison due to some confidential policies. Therefore, in a natural way, we consider manual solutions created by one of the widely-used construction algorithms, for example, vGIA, vSA, vSW. We compare the obtained solution by the A-ALNS algorithm

with the best solution found by construction heuristics (see Table 7). The improvement rate between the average number of unserved customers of the A-ALNS algorithm and the number of unserved customers of the best solution found by construction heuristics (namely F_c) is computed by $\rho_3 = (g_r - avg\_g_r)100/g_r$. We also conduct experiments to compare our proposed algorithm and the local search algorithm of Pham et al. (2017) (LS), the most recent algorithm for the general capacitated vehicle routing problem. The LS strategy is one of appropriate strategies for real-world applications due to its short computational time at each searching iteration and the ability to provide a complete solution at any give time. The LS strategy proposed by Pham et al. (2017) can solve the general CVRP in both offline and online manners using two kinds of neighborhoods. In order to compare the efficiency of the adaptive large neighborhood search with that of the LS, we replace the large neighborhood search in Algorithm 1 by the LS at each searching iteration. We show the improvement rate between the avg_g_r of the A-ALNS algorithm and the g_r of the LS algorithm by $\rho_4 = (g_r - avg\_g_r)100/g_r$.

$acmW \leftarrow 0$;
if $nx(s_1^k) \in C$ **then**
  $dp^* \leftarrow \arg_{dp \in D^*} \min(d_{s_1^k,dp} + d_{dp,s_2^k})$;
  $s^k \leftarrow I(s^k, dp^*, s_1^k)$;
end
for $i = 3$ *to* $\ln(k) - 1$ **do**
  if $acmW + \sum_{p \in P} dm_w(s_i^k, p) > \bar{c}(k)$ **then**
    if $ad(s^k, s_i^k) \geq q(k)$ **then**
      $s^k \leftarrow R(s^k, s_i^k)$;
    end
    else
      $dp^* \leftarrow \arg_{dp \in D^*} \min(d_{s_{i-1}^k,dp} + d_{dp,s_i^k})$;
      $s^k \leftarrow I(s^k, dp^*, s_{i-1}^k)$;
      $acmW \leftarrow 0$;
    end
  end
  else
    $acmW \leftarrow acmW + \sum_{p \in P} dm_w(s_i^k, p)$;
  end
end
return s^k;

Algorithm A.1: SP(s^k)

$G \leftarrow \{\emptyset\}$;
$acmW \leftarrow 0$;
for $i = 3$ *to* $\ln(k) - 1$ **do**
  $acmW \leftarrow acmW + \sum_{p \in P} dm_w(s_i^k, p)$;
  $G \leftarrow G \cup \{s_i^k\}$;
  if $s_i^k \in D^* \cup PK^*$ **then**
    if $acmW < \underline{c}(k)$ **then**
      foreach $e \in G$ **do**
        $s^k \leftarrow R(s^k, e)$;
      end
    end
    $acmW \leftarrow 0$;
  end
end
return s^k;

Algorithm A.2: RCV(s^k)

We found that about 84.64% to 99.52% of total customers are served at most instances where std_g_r is less than or equal to 26.11, the range of ρ_2 is $[0.89, 10.86]$. We see that the existence of the lower-bound capacity constraints makes requests with too little demand and a tight time window to be rejected. The results also reveal that for most instances, the number of unserved customers of the A-ALNS was 31.48% to 91.43% lower than that of constructive algorithms. The A-ALNS algorithm continued to outperform the LS algorithm. The total unserved customers per day of the A-ALNS is significantly improved from the LS algorithm, with the improvement rate remaining extremely stable from 10.87% to 73.33%. For the large instances of the MTDLC-VR problem which has complex constraints, this outcome is explained by the balance mechanism between intensification and diversification of the A-ALNS algorithm is better than that of the LS algorithm.

In general, our proposed algorithm generates good quality solutions for larger instances of this problem within a reasonable computational time; this task that used to take one day is decreased to just two hours. Therefore, the A-ALNS strategy is very robust, and the performance of the A-ALNS strategy should be acceptable in applications as the most appropriate algorithm for solving large instances.

$s \leftarrow$ an empty solution;
$\mathcal{L} \leftarrow$ the saving list are computed using Eq. (25) and sorted non-increasing order;
foreach *element* $M_{i,j}^{dp}$ *in list* $\mathcal{L}$ *from top to bottom* **do**
  if $ix(i) = \perp$ *and* $ix(j) = \perp$ **then**
    $pk^* \leftarrow \arg_{pk \in PK^*} \min d_{pk,dp}$;
    $s^* \leftarrow s$; $flag \leftarrow 0$;
    foreach *route* s^k *of solution* s **do**
      if $p(k) = pk^*$ **then**
        $\hat{s} \leftarrow I(s^k, i, s_{\ln(k)-1}^k)$;
        $\hat{s} \leftarrow I(\hat{s}, j, i)$;
        $\hat{s} \leftarrow SP(\hat{s})$;
        $\bar{s} \leftarrow \mathcal{R}(s, s^k, \hat{s})$;
        if *the solution* $\bar{s}$ *has no violation and* $F(\bar{s}) \leq F(s)$ **then**
          $s^* \leftarrow \bar{s}$; $flag \leftarrow 1$;
        end
      end
    end
    if $flag = 1$ **then**
      $s \leftarrow s^*$;
    end
  end
  else if $ix(i) = \perp$ *and* $ix(j) \neq \perp$ *and* $pr(j) \in D^*$ **then**
    $\hat{s} \leftarrow I(s^{ix(j)}, i, pr(j))$;
    $\hat{s} \leftarrow SP(\hat{s})$;
    $\bar{s} \leftarrow \mathcal{R}(s, s^{ix(j)}, \hat{s})$;
    if *the solution* $\bar{s}$ *has no violation* **then**
      $s \leftarrow \bar{s}$;
    end
  end
  else if $ix(i) \neq \perp$ *and* $ix(j) = \perp$ *and* $nx(i) \in D^* \cup PK^*$ **then**
    $\hat{s} \leftarrow I(s^{ix(i)}, j, i)$;
    $\hat{s} \leftarrow SP(\hat{s})$;
    $\bar{s} \leftarrow \mathcal{R}(s, s^{ix(i)}, \hat{s})$;
    if *the solution* $\bar{s}$ *has no violation* **then**
      $s \leftarrow \bar{s}$;
    end
  end
end
foreach *route* s^k *of solution* s **do**
  $s^* \leftarrow RCV(s^k)$;
  $s \leftarrow \mathcal{R}(s, s^k, s^*)$;
end
return s;

Algorithm A.3: vSA

5.5. Experiment 4: Sensitivity analysis for the lower-bound capacity constraint

- *Description*: In this experiment, we analyze the impact of the new lower-bound capacity constraint in the MTDLC-VR problem. We change the value of the lower boundary of vehicle capacity and count the number of requests that violate the constraint.
- *Data*: We experiment on the real data set E_{21} which has the lower boundary of vehicle capacity changed.

For all vehicle $k \in K$, we denote $\underline{c}(k) = \xi * \bar{c}(k)$, where ξ varies from 10 percent to 90 percent. Fig. 4 presents the number of requests that violate the lower-bound capacity constraint found by four algorithms. The number of requests increases rapidly when the lower boundary of vehicle capacity is increased. We see that the curves have a low-slope from $\underline{c}(k) = 10\%$ to $\underline{c}(k) = 60\%$. Conversely, it increases rapidly from $\underline{c}(k) = 60\%$. It also indicates that the lower-bound constraint has a great influence on the objective function value. Therefore, each manager must balance the objective of increasing the minimum vehicle occupancy rate and the maximum number of customers served. A value of rate between 40% and 50% is acceptable. The results also show that the vGIA algorithm is still better than other construction algorithms and the A-ALNS algorithm has the lowest number of requests violated.

$s \leftarrow$ an empty solution;
for *route* s^k *of solution* s **do**
  $dp^* \leftarrow \arg_{dp \in D^*} \min(d_{s_1^k,dp} + d_{dp,s_2^k})$;
  $c_s \leftarrow \arg_{c \in C \wedge ix(c)=\perp} \min(d_{dp^*,c} + d_{c,dp^*})$;
  $s^* \leftarrow I(s^k, c_s, s^k_{\ln(k)-1})$; $flag \leftarrow 1$;
  while *flag* $= 1$ **do**
    $deg_{min} \leftarrow +\infty$; $flag \leftarrow 0$;
    foreach *the unscheduled customer* $c \in C$ **do**
      $deg \leftarrow$ compute the polar angle by the cosine formula in Pickover (2009);
      $s^* \leftarrow I(s^k, c, s^k_{\ln(k)-1})$;
      if *the time window constraint for that vehicle is reached* **then**
        $s^* \leftarrow$ get suitably sequencing customers by applying TSP problem;
      end
      $s^* \leftarrow SP(s^*)$;
      $\bar{s} \leftarrow \mathcal{R}(s, s^k, s^*)$;
      if *the solution* $\bar{s}$ *has no violation and* $deg < deg_{min}$ **then**
        $s \leftarrow \bar{s}$;
        $deg_{min} \leftarrow deg$; $flag \leftarrow 1$;
      end
    end
  end
  $s^* \leftarrow RCV(s^k)$;
  $s \leftarrow \mathcal{R}(s, s^k, s^*)$;
end
return s;

Algorithm A.4: vSW

6. Conclusion

In this paper, we consider a new scheduling problem of dairy product transportation, called the multi-trip multi-distribution center vehicle routing problem with lower-bound capacity constraints. This paper fills a gap in the literature on the problem by combining some real-world factors, some of which exist in literature while others have not been investigated. We formulated the considered problem by a mixed-integer linear programming (MILP) model and analyzed the challenges of the lower-bound capacity constraints that have not been investigated in the literature. The MILP model was tested by small real instances, small randomly generated instances, and some benchmark instances provided by Solomon (1987). With the presence of the lower-bound capacity constraints, widely-used construction algorithms like SA, SW, GIA are not suitable. Hence, we proposed adapted versions of these algorithms to handle the lower-bound capacity constraints. The solutions computed by these constructions algorithms are then improved by iteratively removing and re-inserting points in an adaptive large neighborhood search framework (this proposed algorithm is called A-ALNS algorithm). To evaluate the efficiency of the proposed A-ALNS algorithm and its practical applicability, we conduct numerical experiments and sensitivity analysis on real data sets of a big milk distribution company in Vietnam. Experimental results on real instances show the effectiveness of our proposed algorithm. The proposed A-ALNS algorithm can improve substantially the quality of the solution computed by construction algorithms and the local search strategy in Pham et al. (2017). The application of the model led to the rapidity in generating the solution; this task that used to take one day is decreased to just two hours. Furthermore, the problem requirements were described by one of the biggest dairy distribution companies in Vietnam, and the proposed algorithm is evaluated on the real data sets. Hence, the performance of the A-ALNS strategy can be acceptable in applications as the most appropriate algorithm for solving large instances of last-mile distribution problems such as perishable product distribution, e-commerce package delivery, and fuel oil distribution problems.

$s \leftarrow$ an empty solution;
for *route* s^k *of solution* s **do**
  while *true* **do**
    $s^* \leftarrow s$; $flag \leftarrow 0$;
    foreach *unscheduled customer* $c \in C$ **do**
      for $i = 1, \ldots, \ln(k) - 1$ **do**
        $\hat{s} \leftarrow I(s^k, c, s^k_i)$;
        $\hat{s} \leftarrow SP(\hat{s})$;
        $\bar{s} \leftarrow \mathcal{R}(s, s^k, \hat{s})$;
        if *the solution* $\bar{s}$ *has no violation and* $F(\bar{s}) \leq F(s)$ **then**
          $s^* \leftarrow \bar{s}$; $flag \leftarrow 1$;
        end
      end
    end
    if $flag = 1$ **then**
      $s \leftarrow s^*$;
    end
    else
      break;
    end
  end
  $s^* \leftarrow RCV(s^k)$;
  $s \leftarrow \mathcal{R}(s, s^k, s^*)$;
end
return s;

Algorithm A.5: vGIA

For future works, the online scenario of the MTDLC-VR problem will be investigated in which requests are not known beforehand and revealed online during the execution of the schedule. Moreover, a challenging area for future research is to extend the problem more flexibly and realistically with a stochastic environment (e.g., stochastic demands and stochastic travel time).

CRediT authorship contribution statement

Van Son Nguyen: Conceptualization, Methodology, Formal analysis, Writing – original draft. **Quang Dung Pham:** Validation, Writing – review & editing, Supervision. **Thanh Hoang Nguyen:** Software, Investigation. **Quoc Trung Bui:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Appendix

The detail of proposed algorithms, results of parameter tuning, and the generated data description are described as follows.

References

Abdelmaguid, T., Dessouky, M., & Ordóñez, F. (2009). Heuristic approaches for the inventory-routing problem with backlogging. *Computers & Industrial Engineering*, 1519–1534.

Akpinar, S. (2016). Hybrid large neighbourhood search algorithm for capacitated vehicle routing problem. *Expert Systems with Applications*, *61*, 28–38.

Alinaghian, M., & Shokouhi, N. (2018). Multi-depot multi-compartment vehicle routing problem, solved by a hybrid adaptive large neighborhood search. *Omega*, *76*, 85–99. http://dx.doi.org/10.1016/j.omega.2017.05.002.

Altabeeb, A. M., Mohsen, A. M., & Ghallab, A. (2019). An improved hybrid firefly algorithm for capacitated vehicle routing problem. *Applied Soft Computing*, *84*, Article 105728.

Babaee Tirkolaee, E., Goli, A., Bakhsi, M., & Mahdavi, I. (2017). A robust multi-trip vehicle routing problem of perishable products with intermediate depots and time windows. *Numerical Algebra, Control and Optimization*, *7*, 417–433.

Baradaran, V., Shafaei, A., & Hosseinian, A. H. (2019). Stochastic vehicle routing problem with heterogeneous vehicles and multiple prioritized time windows: Mathematical modeling and solution approach. *Computers & Industrial Engineering*, *131*, 187–199.

Beasley, J. (1983). Route first—Cluster second methods for vehicle routing. *Omega*, *11*(4), 403–408. http://dx.doi.org/10.1016/0305-0483(83)90033-6.

Bettinelli, A., Ceselli, A., & Righini, G. (2011). A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows. *Transportation Research Part C (Emerging Technologies)*, *19*(5), 723–740, Freight Transportation and Logistics (selected papers from ODYSSEUS 2009 - the 4th International Workshop on Freight Transportation and Logistics).

Bezerra, S. N., Souza, S. R. D., & Souza, M. J. F. (2018). A GVNS algorithm for solving the multi-depot vehicle routing problem. *Electronic Notes in Discrete Mathematics*, *66*, 167–174. http://dx.doi.org/10.1016/j.endm.2018.03.022.

Bouzid, M. C., Aït Haddadene, H., & Salhi, S. (2017). An integration of Lagrangian split and VNS: The case of the capacitated vehicle routing problem. *Computers & Operations Research*, *78*, 513–525.

Çatay, B. (2010). A new saving-based ant algorithm for the vehicle routing problem with simultaneous pickup and delivery. *Expert Systems with Applications*, *37*(10), 6809–6817.

Cattaruzza, D., Absi, N., & Feillet, D. (2018). Vehicle routing problems with multiple trips. *Annals of Operations Research*, *271*, 127–159.

Chen, C., Demir, E., & 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.

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, *12*(4), 568–581. http://dx.doi.org/10.1287/opre.12.4.568.

Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, *6*(1), 80–91. http://dx.doi.org/10.1287/mnsc.6.1.80.

Du, J., Li, X., Yu, L., Dan, R., & Zhou, J. (2017). Multi-depot vehicle routing problem for hazardous materials transportation: A fuzzy bilevel programming. *Information Sciences*, *399*, 201–218.

Fleischmann, B. (1990). *The vehicle routing problem with multiple use of vehicles*. Hamburg, Germany: Facbereich Wirtschaftswissenschafte, Universitat Hamburg, Working Paper.

Forbes (2010). Asia's 200 best under a billion.

François, V., Arda, Y., Crama, Y., & Laporte, G. (2016). Large neighborhood search for multi-trip vehicle routing. *European Journal of Operational Research*, *255*, 422–441.

Ganepola, D. D., Jayarathna, N. D., & Madhushani, G. (2018). An intelligent cost optimized central warehouse and redistribution root plan with truck allocation system in colombo region for lion brewery ceylon PLC. *Journal of Sustainable Development of Transport and Logistics*.

Ghilas, V., Demir, E., & Woensel, T. V. (2016). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. *Computers & Operations Research*, *72*, 12–30. http://dx.doi.org/10.1016/j.cor.2016.01.018.

Gillett, B. E., & Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, *22*(2), 340–349. http://dx.doi.org/10.1287/opre.22.2.340.

Granada-Echeverri, M., Bolaños, R., & Escobar, J. (2018). A metaheuristic algorithm for the MultiDepot vehicle routing problem with heterogeneous fleet. *International Journal of Industrial Engineering Computations*, *9*, 461–478. http://dx.doi.org/10.5267/j.ijiec.2017.11.005.

Gulczynski, D., Golden, B., & Wasil, E. (2011). The multi-depot vehicle routing problem: An integer programming-based heuristic and computational results. *Computers & Industrial Engineering*, *61*, 794–804. http://dx.doi.org/10.1016/j.cie.2011.05.012.

Hernandez, F., Feillet, D., Giroudeau, R., & Naud, O. (2016). Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows. *European Journal of Operational Research*, *249*(2), 551–559.

Ioannou, G., Kritikos, M., & Prastacos, G. (2001). A greedy look-ahead heuristic for the vehicle routing problem with time windows. *Journal of the Operational Research Society*, *52*(5), 523–537. http://dx.doi.org/10.1057/palgrave.jors.2601113.

Karp, R. M. (2009). Reducibility among combinatorial problems. *50 Years of Integer Programming 1958-2008*, 219–241. http://dx.doi.org/10.1007/978-3-540-68279-0_8.

Konstantakopoulos, G. D., Gayialis, S., & Kechagias, E. (2020). Vehicle routing problem and related algorithms for logistics distribution: A literature review and classification. *Operational Research*, 1–30.

Kramer, R., Cordeau, J. -F., & Iori, M. (2019). Rich vehicle routing with auxiliary depots and anticipated deliveries: An application to pharmaceutical distribution. *Transportation Research Part E: Logistics and Transportation Review*, *129*, 162–174.

Lahyani, R., Gouguenheim, A., & Coelho, L. C. (2019). A hybrid adaptive large neighbourhood search for multi-depot open vehicle routing problems. *International Journal of Productions Research*, *57*, 6963–6976.

Lalla-Ruiz, E., Expósito-Izquierdo, C., Taheripour, S., & VoB, S. (2016). An improved formulation for the multi-depot open vehicle routing problem. *OR Spectrum*, *38*(1), 175–187. http://dx.doi.org/10.1007/s00291-015-0408-9.

Laporte, G., Gendreau, M., Potvin, J. -Y., & Semet, F. (2000). Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*, *7*(4–5), 285–300. http://dx.doi.org/10.1111/j.1475-3995.2000.tb00200.x.

Li, H., Yuan, J., Lv, T., & Chang, X. (2016). The two-echelon time-constrained vehicle routing problem in linehaul-delivery systems considering carbon dioxide emissions. *Transportation Research Part D: Transport and Environment*, *49*, 231–245. http://dx.doi.org/10.1016/j.trd.2016.10.002.

Liang, Y., Liu, F., Lim, A., & Zhang, D. (2020). An integrated route, temperature and humidity planning problem for the distribution of perishable products. *Computers & Industrial Engineering*, *147*, Article 106623.

Liu, R., Jiang, Z., & Geng, N. (2014). A hybrid genetic algorithm for the multi-depot open vehicle routing problem. *OR Spectrum*, *36*(2), 401–421. http://dx.doi.org/10.1007/s00291-012-0289-0.

Louveaux, F. V., & Salazar-González, J. -J. (2016). Solving the single vehicle routing problem with variable capacity. *Transportation Science*, *50*(2), 708–719. http://dx.doi.org/10.1287/trsc.2014.0556.

Masmoudi, M. A., Hosny, M., Braekers, K., & Dammak, A. (2016). Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. *Transportation Research Part E: Logistics and Transportation Review*, *96*, 60–80.

Mirzaei, S., & Seifi, A. (2015). Considering lost sale in inventory routing problems for perishable goods. *Computers & Industrial Engineering*, *87*, 213–227.

Montoya-Torres, J. R., Franco, J. L., Isaza, S. N., Jiménez, H. F., & Herazo-Padilla, N. (2015). A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering*, *79*, 115–129. http://dx.doi.org/10.1016/j.cie.2014.10.029.

Na, B., Jun, Y., & Kim, B. -I. (2011). Some extensions to the sweep algorithm. *International Journal of Advanced Manufacturing Technology*, *56*(9–12), 1057–1067. http://dx.doi.org/10.1007/s00170-011-3240-7.

Nguyen, V. S., Pham, Q. D., & Phan, A. T. (2019). An adaptive large neighborhood search solving large people and parcel share-a-ride problem. In *2019 6th NAFOSTED conference on information and computer science*. http://dx.doi.org/10.1109/nics48868.2019.9023893.

Olivera, A., & Viera, O. (2007). Adaptive memory programming for the vehicle routing problem with multiple trips. *Computers & Operations Research*, *34*(1), 28–47.

Pan, B., Zhang, Z., & Lim, A. (2021). Multi-trip time-dependent vehicle routing problem with time windows. *European Journal of Operational Research*, *291*(1), 218–231.

Paradiso, R., Roberti, R., Laganá, D., & Dullaert, W. (2020). An exact solution framework for multitrip vehicle-routing problems with time windows. *Operations Research*, *68*(1), 180–198.

Pham, Q. D., Le, K. T., Nguyen, H. T., Pham, V. D., & Bui, Q. T. (2017). A constraint-based local search for offline and online general vehicle routing. *International Journal on Artificial Intelligence Tools*, *26*(02), Article 1750004. http://dx.doi.org/10.1142/s021821301750004x.

Pichpibul, T., & Kawtummachai, R. (2013). A heuristic approach based on clarke-wright algorithm for open vehicle routing problem. *The Scientific World Journal*, *2013*, 1–11. http://dx.doi.org/10.1155/2013/874349.

Potvin, J. -Y., & Rousseau, J. -M. (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, *66*(3), 331–340. http://dx.doi.org/10.1016/0377-2217(93)90221-8.

Prins, C., Lacomme, P., & Prodhon, C. (2014). Order-first split-second methods for vehicle routing problems: A review. *Transportation Research Part C (Emerging Technologies)*, *40*, 179–200. http://dx.doi.org/10.1016/j.trc.2014.01.011.

Ribeiro, G., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, *39*, 728–735.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*(4), 455–472. http://dx.doi.org/10.1287/trsc.1050.0135.

Shaw, P. (1997). *A new local search algorithm providing high quality solutions to vehicle routing problems*: *Technical report*, Scotland: Department of Computer Science, University of Strathclyde.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming — CP98 Lecture Notes in Computer Science* (pp. 417–431). http://dx.doi.org/10.1007/3-540-49481-2_30.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, *35*(2), 254–265. http://dx.doi.org/10.1287/opre.35.2.254.

Song, B. D., & Ko, Y. D. (2016). A vehicle routing problem of both refrigerated- and general-type vehicles for perishable food products delivery. *Journal of Food Engineering*, *169*, 61–71.

Tarantilis, C. D., & Kiranoudis, C. T. (2002). Distribution of fresh meat. *Journal of Food Engineering*, *51*(1), 85–91. http://dx.doi.org/10.1016/S0260-8774(01)00040-1.

Toth, P., & Vigo, D. (2014). *Vehicle routing*. Society for Industrial and Applied Mathematics, http://dx.doi.org/10.1137/1.9781611973594.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2014). Implicit depot assignments and rotations in vehicle routing heuristics. *European Journal of Operational Research*, *237*(1), 15–28.

Wang, Z. (2018). Delivering meals for multiple suppliers: Exclusive or sharing logistics service. *Transportation Research Part E: Logistics and Transportation Review*, *118*, 496–512.

Zhen, L., Ma, C., Wang, K., Xiao, L., & Zhang, W. (2020). Multi-depot multi-trip vehicle routing problem with time windows and release dates. *Transportation Research Part E: Logistics and Transportation Review*, *135*, Article 101866. http://dx.doi.org/10.1016/j.tre.2020.101866.

Zou, H., & Dessouky, M. (2018). A look-ahead partial routing framework for the stochastic and dynamic vehicle routing problem. *Journal on Vehicle Routing Algorithms*, *1*, 73–88. http://dx.doi.org/10.1007/s41604-018-0006-5.