Computers & Operations Research 69 (2016) 109–125

Contents lists available at ScienceDirect

Computers & Operations Research

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

An Adaptive Large Neighborhood Search for an E-grocery Delivery Routing Problem

Uğur Emeç [a], Bülent Çatay [a], Burcin Bozkaya [b,*]

[a] Sabanci University, Faculty of Engineering and Natural Sciences, Tuzla, 34956 Istanbul, Turkey
[b] Sabanci University, School of Management, Tuzla, 34956 Istanbul, Turkey

ARTICLE INFO

Available online 17 December 2015

Keywords:
E-grocery Delivery Routing Problem
Home Delivery Services
Vehicle Routing Problem with Time Windows
Premium Good
Premium Customer
Adaptive Large Neighborhood Search

ABSTRACT

Online shopping has become ever more indispensable to many people with busy schedules who have a growing need for services ranging for a wide variety of goods, which include standard (or “staple”) goods as well as “premium” goods, i.e. goods such as organic food, specialty gifts, etc. that offer higher value to consumers and higher profit margins to retailers. In this paper, we introduce a new mathematical programming formulation and present an efficient solution approach for planning the delivery services of online groceries to fulfill this diverse consumer demand without incurring additional inventory costs. We refer to our proposed model as the E-grocery Delivery Routing Problem (EDRP) as it generically represents a family of problems that an online grocery is likely to face. The EDRP is based on a distribution network where premium goods are acquired from a set of external vendors at multiple locations in the supply network and delivered to customers in a single visit. To solve this problem, we develop an improved Adaptive Large Neighborhood Search (ALNS) heuristic by introducing new removal, insertion, and vendor selection/allocation mechanisms. We validate the performance of the proposed ALNS heuristic through an extensive computational study using both the well-known Vehicle Routing Problem with Time Windows instances of Solomon and a set of new benchmark instances generated for the EDRP. The results suggest that the proposed solution methodology is effective in obtaining high quality solutions fast.

1. Introduction

The habits of consumers shopping online have rapidly changed in the last decade as a result of remarkable developments in e-commerce and the constant search of online retailers for new and more profitable business models offering more flexibility and alternative shopping experience for consumers. Many leading and visionary online retailers look for up-and-coming business strategies to diversify their in-stock or in-store “standard” (or “staple”) product offerings with outsourced “premium” products, i.e. products such as organic or dietary food, specialty gifts, specialty wine, etc., that offer higher satisfaction value to consumers and higher profit margins to retailers. For instance, a popular online retailer AmazonFresh (fresh.amazon.com, 2014) offers a variety of grocery items like wine, pumpkin pie, vegetables, meat, seafood, etc. in addition to items from the main Amazon.com storefront. Another online retailer Peapod (www.peapod.com, 2014) provides groceries and vegetables via a centralized business model using warehouses and warerooms, where warerooms are the dedicated areas attached to a subsidiary international food provider. The main premise in all these new practices is to extend the traditional online offering to include additional high-value items for consumers' choice.

In this paper, we revisit this very idea introduced in two earlier studies by Bozkaya et al. [10] and Yanik et al. [47], and formally introduce the generalized problem as the E-grocery Delivery Routing Problem (EDRP). The problem is motivated by a large Turkish supermarket chain offering home delivery services to online shoppers. The company aims at gaining competitive advantage through a new business model that provides extended service to its customers. In this model, an online supermarket that normally uses its brick-and-mortar stores to fulfill online customer orders may now additionally collaborate with external vendors that offer an extended inventory of diverse consumer items. Hence, the shopping basket of an online customer is fulfilled either from the store or from an external vendor, depending on the items ordered. Within the same context, Campbell and Savelsbergh [11]

* Corresponding author. Tel.: +90 216 483 9687.
E-mail address: bbozkaya@sabanciuniv.edu (B. Bozkaya).

http://dx.doi.org/10.1016/j.cor.2015.11.008
0305-0548/© 2015 Elsevier Ltd. All rights reserved.

give the more restricted partnership model of Staples that offers office supplies and a grocery delivery service in Canada.

In a similar setting, Instacart (www.instacart.com, 2014), an independent grocery delivery service operating in several major cities in the U.S. (e.g. Seattle, San Francisco, Los Angeles, Austin, Maryland, Philadelphia) allows its customers to combine items from multiple groceries in their area (such as Whole Foods Market, Costo, Kroger, Safeway and other retailers) into one order and have them all delivered in one order on the same day, even within an hour. AmazonFresh offers same-day delivery of fresh grocery and local products from bakery to ethnic foods to gourmet meals or organic food from the neighborhood shops, restaurants and bakeries in Northern California. In this new business paradigm, Amazon has recently expanded its delivery network and started operating its own truck fleet in San Francisco area [9].

Such online shopping and delivery service is obviously not limited to the groceries and may be extended to other retail sectors. Our main contribution in this paper includes an alternative mathematical model and an effective solution procedure for the underlying delivery problem with a set of additional business rules. We leave the topic of profit sharing amongst the e-grocery and the collaborating vendors to a future study, and focus on the logistics of the distribution operation, i.e. the cost minimization aspect.

The EDRP that we tackle is based on a distribution network, which consists of (1) a depot, i.e. a store of the online grocery that supplies "standard" products, (2) vendors, i.e. external stores that supply their own set of "premium" products not available at the depot, (3) "regular" customers, i.e. customers that only purchase standard products and (4) "premium" customers, i.e. customers that *additionally* purchase premium products. Routing regular customers only is a straightforward special case of the EDRP in the form of VRP (Vehicle Routing Problem) with or without time windows. On the other hand, routing premium customers present an additional challenge as two additional sets of decisions are to be made simultaneously: (i) allocation of vendor(s) to each premium customer so as to satisfy the customer's order of premium product(s), (ii) routing of regular and premium customers, and their respective vendor set while preserving feasibility concerns such as precedence, vehicle capacity, time windows. Under these circumstances, the delivery of goods to each premium customer takes place only after the entire set of premium products are collected and combined with the standard products already loaded at the depot. Transfers between vehicles are not allowed and the minimum total distance solution is sought in the presence of precedence, time windows, and capacity constraints.

The main contributions of this study can be summarized as follows:

- We propose an alternative generalized mathematical model, which focuses on the distribution aspect of the problem and attempts to minimize the total transportation costs subject to additional business rules.
- We propose an effective Adaptive Large Neighborhood Search (ALNS) heuristic to solve the EDRP. The proposed ALNS includes new selection/allocation mechanisms for EDRP to tackle a more complex problem structure due to inclusion of external vendors. It also improves some of the existing removal/insertion mechanisms.
- We validate the performance of the proposed ALNS using the Vehicle Routing Problem with Time Windows (VRPTW) instances of Solomon [43] and improve the best-known solutions for five real-numbered problems.
- We randomly generate a large set of EDRP instances based on Solomon's data and report benchmark results for future studies.

The remainder of this paper is organized as follows. In Section 2, we review the related literature. In Section 3, we describe the EDRP and formulate a 0–1 mixed-integer linear programming model. In Section 4, we provide the details of our proposed ALNS approach. In Section 5, we present an experimental design and present the computational results. Finally, we provide our concluding remarks in Section 6.

2. Literature Review

The essence of the EDRP, namely the concept of using multiple sourcing and consolidation points to fulfill premium orders of consumers in e-grocery settings is first identified and discussed by Bozkaya et al. [10]. In a succeeding study, Yanik et al. [47] investigate the role of premium product offerings in creating critical mass and profit, and propose a hybrid metaheuristic approach that employs a genetic algorithm for vendor selection-allocation phase followed by a modified savings algorithm for the vehicle routing phase. The proposed genetic algorithm guides the search for optimal vendor pick-up location assignments. The authors also show possible profit opportunities of the new business model on a case study using a real dataset.

VRP with intermediate facilities (VRP-IF) and VRP with satellite facilities (VRP-SF) are the two VRP variants that are closely related to EDRP. For instance, Angelelli and Speranza [3] study periodic VRP for a collection problem where the vehicles renew their capacity at certain intermediate facilities. Sevilla and Blas [41] take into account time windows in VRP-IF and propose an algorithm based on neural networks and an ant colony system to solve the problem. Tarantilis et al. [44] address the case where vehicles start their trips from a central depot with intermediate depots again acting as replenishment stations. They propose a three-step algorithmic approach where an initial solution is first obtained by a cost-saving construction heuristic, followed by a tabu search within a variable neighborhood search improvement framework. Finally, they apply a guided local search to eliminate low-quality features from the final solution produced. Polacek et al. [33] develop a simple and robust variable neighborhood search algorithm to solve the capacitated arc routing problem with intermediate facilities. Liu et al. [21] consider a waste collection problem in the presence of intermediate facilities where the vehicles are unloaded when they are full, and propose an improved ant colony system algorithm to solve it. Bard et al. [6,7] consider satellite facilities to replenish vehicles during a route. They present a branch-and-cut methodology for solving the VRP-SF subject to capacity and route time constraints.

The EDRP can also be viewed as a variant of VRP with pick-ups and deliveries (VRPPD) in which goods are transported from pickup points to delivery points where all pickups must be made before the deliveries. Recent studies offer a variety of approaches to solve the VRPPD, including tabu search [12,23,24,27,4,45], genetic algorithms [16,35], simulated annealing [17,20], ant colony optimization [15] and hybrid heuristics [8]. We refer the interested reader to the extensive reviews by Parragh et al. [28,29] of the problem classifications, formulations, exact and metaheuristic solution approaches.

In terms of effectiveness and flexibility, the approach introduced by Ropke and Pisinger [39] is one of the best methods at hand for solving the VRPPD as well as a large class of VRP variants [31,40]. Their proposed metaheuristic employs ALNS as an extension of the Large Neighborhood Search (LNS) framework put forward by Shaw [42]. This ALNS aims to improve an initial feasible solution progressively by means of multiple removal (destroy) and insertion (repair) mechanisms competing in an adaptive environment to diversify and intensify the search. ALNS has been successfully implemented for solving many VRP variants including pickup and delivery problems with transshipment [34] and with vehicle transfers [22], cumulative capacitated VRP [36], pollution-routing problem [14], two-echelon VRP [18], VRP with multiple routes [5], periodic inventory routing problem [2] and production routing problem [1]. Due to its flexibility, the ALNS framework has been also applied to a wide range of problems such as the resource-constrained project scheduling problem [25], technician task scheduling in telecommunications industry [13], lot-sizing with setup times [26], consultation timetabling problem at Danish high schools [19]. We further refer the interested reader to Pisinger and Ropke [32] for a recent survey on large neighborhood search, its variants and extensions like the ALNS framework.

In this study, we propose an ALNS approach that modifies and improves some of the existing removal and insertion procedures in the literature. Also, we introduce novel removal/insertion and vendor selection/allocation procedures specific to the EDRP. We further contribute by introducing additional adaptive scoring mechanisms for self-adjustment of some removal/insertion procedures and multiple initial solutions during the search.

3. Problem Description

3.1. Notation

Let $G=(N,A)$ be an undirected complete graph. $N=0\cup F$ is the set of nodes where "0" is the depot and $F=V\cup C$ represents the collection of external vendors V and customers C. We further define $C=C^R\cup C^P$ where C^R and C^P are the sets of regular and premium customers, respectively. A regular customer only orders regular products available at the depot and a premium customer orders both regular and premium products. The external vendors supplying premium products need not be identical in their product offerings and they collectively provide a set of premium products denoted by P. The premium product supply range of the external vendors is represented by a binary matrix $G=\left[g_{vp}\in\{0,1\}\right]_{|V|\times|P|}$, i.e. $g_{vp}=1$ if vendor $v\in V$ supplies premium product $p\in P$. The demand indicator of the premium customers is also represented by a binary matrix $B=\left[b_{cp}\in\{0,1\}\right]_{|C^P|\times|P|}$, i.e. $b_{cp}=1$ if premium customer $c\in C^P$ requests a premium product $p\in P$. Furthermore, premium order quantities are represented by the matrix $Q=\left[q_{cp}\in\mathbb{Z}\right]_{|C^P|\times|P|}$. Each premium product $p\in P$ is associated with a unit volume of w_p and the total volume of the premium products ordered by each customer $c\in C^P$ is derived from $\sum_{p\in P}q_{cp}w_p$. The standard product volume demanded by all customers $c\in C$ is denoted by d_c.

We assume that the depot and the external vendors have unlimited supply capacities. Also, an order for premium product $p\in P$ can only be supplied by one of the eligible external vendors. The fleet consists of K of homogenous vehicles with capacity $\mathcal{U}$ and each vehicle $k\in K$ departs from and returns to the depot serving one route. Furthermore, we assume a multiple pick-up but single delivery to the premium customers. That is, a premium customer $c\in C^P$ can only be visited after all of his premium orders are collected and combined with his standard products in the same vehicle. Each node $i\in F$ is further associated with a service time of s_i, which represents loading time for external vendors and unloading time for customers. Each arc (i,j) is associated with a travel time t_{ij} and a travel cost c_{ij}. Finally, the service time window at each node $i\in N$ is represented by $[e_i,l_i]$.

3.2. An Illustrative Example

To illustrate the EDRP on a small example, consider the problem setting in Fig. 1. In Fig. 1a, the circles represent regular and premium customer nodes, the hexagons represent the external vendor nodes, and the triangle represents the depot node. Attached to each node are its (x,y) coordinates in the Euclidean space. There are four vendors with $V=\{\text{V1},\ldots,\text{V4}\}$, six regular customers with $C^R=\{\text{C1},\ldots,\text{C6}\}$, and three premium customers with $C^P=\{\text{C7},\text{C8},\text{C9}\}$. Three premium products with $P=\{1,2,3\}$ have volumes of $w_{\{1,2,3\}}=\{5,4,6\}$ per unit. The demand for the standard products is $d_{\{\text{C1},\ldots,\text{C9}\}}=\{10,30,20,10,40,20,5,10,8\}$. The external vendors supply the following premium products: $G=[[1,0,0],[0,1,0],[0,1,0],[0,0,1]]$. The premium demand indicator matrix and premium order quantity matrix of the premium customers are $B=[[1,0,1],[1,1,0],[0,0,1]]$ and $Q=[[3,0,2],[1,5,0],[0,0,10]]$, respectively. There are two identical vehicles, i.e. $K=\{1,2\}$, with a capacity of $\mathcal{U}=100$. For simplicity, service times are set equal to zero and the earliest and latest service times are $[e_i,l_i]=[0,200]$.

The optimal solution of the above problem setting is illustrated in Fig. 1b. Vehicle 1 follows the route D → V4 → C1 → V1 → C7 → V2 → C8 → C3 → C2 → D traveling a total distance of 170 whereas Vehicle 2 follows the route D → C4 → C5 → V4 → C6 → C9 → D traveling a total distance of 113.50. The optimal distribution cost is then 283.50.

3.3. Mathematical Model

The proposed mathematical model includes the following binary decision variables: if vehicle $k\in K$ travels from node $i\in N$ to node $j\in N$, $x_{ijk}=1$, otherwise $x_{ijk}=0$. If a premium customer $c\in C^P$

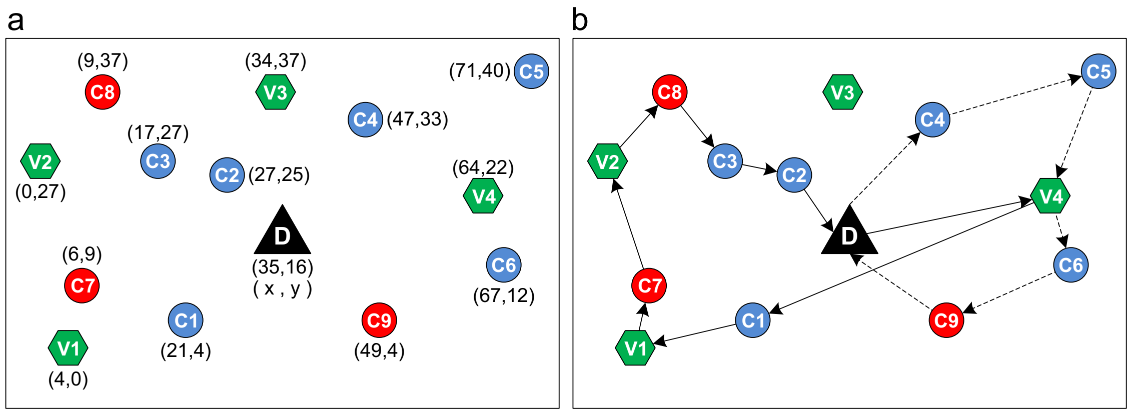

Fig. 1. An illustrative example for the EDRP: (a) Problem setting and (b) optimal solution with the corresponding routes.

is served by vendor $v \in V$ using vehicle $k \in K$, $z_{cvk}=1$, otherwise $z_{cvk}=0$. If premium order $p \in P$ of customer $c \in C^P$ is supplied by vendor $v \in V$ using vehicle $k \in K$, $r_{cvpk}=1$, otherwise $r_{cvpk}=0$. The continuous variables T_{ik} represent the arrival time of vehicle $k \in K$ at node $i \in N$ and L_{ik} represent the load of vehicle $k \in K$ as it leaves node $i \in N$. Finally, M_1 and M_2 are sufficiently large constants associated with time and load related constraints.

The mixed integer linear programming formulation of EDRP is then as follows:

$$\text{Minimize} z = \sum_{k \in K}\sum_{i \in N}\sum_{j \in N} c_{ij}x_{ijk} \tag{1}$$

Subjectto :

$$\sum_{j \in F} x_{0jk} \le 1, \forall k \in K \tag{2}$$

$$\sum_{k \in K}\sum_{j \in N} x_{jck} = 1, \forall c \in C \tag{3}$$

$$\sum_{j \in N} x_{ijk} = \sum_{j \in N} x_{jik}, \forall i \in N, k \in K \tag{4}$$

$$|C^P| \bullet \sum_{j \in N} x_{jvk} \ge \sum_{c \in C^P} z_{cvk}, \forall v \in V, k \in K \tag{5}$$

$$|V| \bullet \sum_{j \in N} x_{jck} \ge \sum_{v \in V} z_{cvk}, \forall c \in C^P, k \in K \tag{6}$$

$$T_{vk} + s_v + t_{vc} \le T_{ck} + M_1(1 - z_{cvk}), \forall k \in K, v \in V, c \in C^P \tag{7}$$

$$T_{ik} + s_i + t_{ij} \le T_{jk} + M_1(1 - x_{ijk}), \forall k \in K, i \in N, j \in F \tag{8}$$

$$e_i \le T_{ik} \le l_i, \forall i \in N, k \in K \tag{9}$$

$$T_{ik} + x_{i0k}(t_{i0} + s_i) \le M_1(1 - x_{i0k}) + l_0, \forall i \in N, k \in K \tag{10}$$

$$z_{cvk} \ge r_{cvpk}, \forall c \in C^P, v \in V, p \in P, k \in K \tag{11}$$

$$\sum_{k \in K}\sum_{v \in V} g_{vp} r_{cvpk} = b_{cp}, \forall c \in C^P, p \in P \tag{12}$$

$$L_{0k} = \sum_{c \in C}\sum_{j \in N} d_c x_{jck}, \forall k \in K \tag{13}$$

$$L_{jk} - (d_c + \sum_{p \in P} q_{cp} w_p) \le L_{ck} + M_2(1 - x_{jck}), \forall c \in C^P, j \in N, k \in K \tag{14}$$

$$L_{jk} - d_c \le L_{ck} + M_2(1 - x_{jck}), \forall c \in C^R, j \in N, k \in K \tag{15}$$

$$L_{jk} + \sum_{c \in C^P}\sum_{p \in P} r_{cvpk} q_{cp} w_p \le L_{vk} + M_2(1 - x_{jvk}), \forall v \in V, j \in N, k \in K \tag{16}$$

$$L_{ik} \le \mathcal{U} \sum_{j \in N} x_{ijk}, \forall i \in N, k \in K \tag{17}$$

$$x_{ijk} \in \{0,1\}, \forall i,j \in N, i \ne j, k \in K \tag{18}$$

$$y_k \in \{0,1\}, \forall k \in K \tag{19}$$

$$z_{cvk} \in \{0,1\}, \forall c \in C^P, v \in V, k \in K \tag{20}$$

$$r_{cvpk} \in \{0,1\}, \forall c \in C^P, v \in V, p \in P, k \in K \tag{21}$$

The objective function minimizes the total distribution costs associated with the distance traveled. Constraints (2) ensure that a vehicle starts its route at the depot. Constraints (2) along with constraints (8), which are essential for the sub-tour elimination, also ensure that a vehicle cannot appear in any route if it is not used. Constraints (3) make sure that each customer is visited once and constraints (4) are the flow balance constraints. Furthermore, constraints (4) together with constraints (2), (8), and (18) ensure that each route terminates at the depot. Constraints (5) and (6) ensure consistent assignment of vendors and premium customers to the vehicles routes and schedules. Premium product orders of premium customers are collected from appropriate set of vendors before final delivery time as a result of constraints (7). Moreover, the time consistency between consecutive stops in each route is ensured via constraints (8) and constraints (9) guarantee that each customer is served within a prespecified time window. These constraints also eliminate the sub-tours. Constraints (10) ensure that all vehicles return to the depot within the corresponding time window. The supply of different premium product orders of customers by exactly one of the proper vendors is ensured by constraints (12). Constraints (13) make sure that the initial load of each vehicle equals to the total standard product volume of all customers assigned to that vehicle. Constraints (14) and (15) ensure that the load balance of each vehicle is consistent throughout its route after each delivery to premium customers and regular customers assigned to that vehicle, respectively. Similarly, constraints (16) keep track of the premium product load after collection at each vendor. Constraints (17) guarantee that the vehicle capacities are not violated. Finally, constraints (18)–(21) define the binary decision variables. Note that since VRP with Time Windows, a special case of EDRP formulated above, is NP-Hard [46], EDRP is also NP-Hard.

4. Proposed Methodology

To solve the EDRP described above, we propose an Adaptive Large Neighborhood Search (ALNS) framework. In what follows, we first provide a general framework of our methodology, followed by details on the specific algorithmic components.

4.1. ALNS for the EDRP

Our ALNS approach consists of three main sets of algorithms: (i) $\mathcal{A}_R$: Removal, (ii) $\mathcal{A}_V$: Vendor selection/allocation, and (iii) $\mathcal{A}_I$: Insertion algorithms. It combines the strengths of the ALNS heuristics previously put forward by Ropke and Pisinger [39,40], Pisinger and Ropke [31], and Demir et al. [14] by modifying some of the existing removal and insertion algorithms as well as introducing new removal, insertion and vendor selection/allocation algorithms specific to EDRP. It also introduces additional scoring mechanisms for self-adjustment of some removal/insertion mechanisms and produces multiple initial solutions during the search. The main aspects of the proposed ALNS approach are described as follows.

4.1.1. General flow

Let S_i be the current feasible solution on hand at the beginning of iteration i and S_i^- be a partial feasible solution. At iteration i, a removal algorithm $r \in \mathcal{A}_R$, a vendor selection/allocation algorithm $v \in \mathcal{A}_V$ and an insertion algorithm $i \in \mathcal{A}_I$ are dynamically and independently selected. Next, S_i^- is obtained by removing n_i regular and premium customers from S_i by using the selected removal algorithm. Then, a new proper vendor set is selected and appended to each removed premium customer via the selected vendor selection/allocation algorithm v. Finally, each removed customer is inserted into S_i^- with its respective vendor set, if any, by means of the selected insertion algorithm i. At the end of the iteration, we obtain a temporary feasible solution S_i^+ which can be discarded or made the new feasible solution by setting $S_{i+1} \leftarrow S_i^+$.

4.1.2. Large neighborhood

The neighborhood size is determined by n_i and it has a substantial effect on the performance of the ALNS algorithm. If $n_i \ll |C|$, the effect of the large neighborhood structure is lost and the search space may not be explored in an efficient way. On the other hand, if $n_i \gg |C|$, repairing S_i^- may be very time-consuming and/or the resulting S_i^+ may be of poor quality [32].

4.1.3. Adaptive scoring

The entire search procedure of $\mathcal{N}$ iterations is divided into Δ “segments” and $\mathcal{N}_S = \lfloor \mathcal{N}/\Delta \rfloor$ represents the number of sequential iterations in a segment. Each algorithm $a \in \mathcal{A}_R \cup \mathcal{A}_V \cup \mathcal{A}_I$ is associated with a score π_a, which shows how well an algorithm has performed during a segment. All π_a values are initialized to zero at the beginning of each segment. If a new global best solution S^* is found in an iteration of a segment, the π_a scores of r, v and i are increased by σ_1, as neither of the three algorithms is responsible for the improvement by itself. If S_i^+ is better than S_i, the π_a scores of r, v and i are increased by σ_2. On the other hand, if S_i^+ is accepted even though it is worse than S_i, the π_a scores of r, v and i are incremented by σ_3. Note that $\sigma_1, \sigma_2, \sigma_3 > 0$.

4.1.4. Adaptive weight adjustment

Let η_a^s and τ_a^s represent the adaptive weight of algorithm a and the number of times algorithm a has been selected during segment $s = 1, \ldots, \Delta$, respectively. If $s = 1$, then $\eta_a^s = 1$, i.e. initially all algorithms have the same weight. After $\mathcal{N}_S$ iterations, i.e. at the beginning of segment $s+1$, η_a^{s+1} value is updated for each algorithm a according to their π_a scores obtained during the previous segments as follows:

$$\eta_a^{s+1} = \begin{cases} \eta_a^s \&, \tau_a^s = 0 \\ (1-\rho)\eta_a^s + \rho \frac{\pi_a}{\tau_a^s}, \tau_a^s > 0 \end{cases} \tag{22}$$

where $\rho \in [0,1]$ is a parameter called as *reaction factor* that controls how quickly the adaptive weight adjustment mechanism reacts to changes in the effectiveness of the algorithms.

4.1.5. Adaptive selection

r, v and i are independently and individually selected by means of a roulette wheel mechanism based on their past performances which are dynamically updated with respect to their adaptive weights using (22). Given m algorithms (e.g. $m = |\mathcal{A}_R|$), let P_a^s be the probability of selecting algorithm a (e.g. $a \in \mathcal{A}_R$ during segments. Then P_a^s is given by:

$$P_a^s = \eta_a^s / \sum_{l=1,\ldots,m} \eta_l^s \tag{23}$$

4.1.6. Initial solutions

Each starting feasible solution of a segment is generated by either the *Greedy Heuristic* or *Regret-2 Heuristic* described in Appendix A. Initially (for $s = 1$), either heuristic has an equal chance of being selected. The probability of selection is calculated at the beginning of each succeeding segment using (22) and (23), but taking into account only σ_1 and σ_2 while updating π_a values for these two algorithms, and using $m = 2$ as the basis of selection. A new segment with the initial solution S_1 is started if there is no improvement in S^* in a segment for $\mathcal{N}_{IWI}$ iterations (number of iterations without improvement).

4.1.7. Acceptance and stopping criteria

A Simulated Annealing (SA) local search framework is used at the master level of our proposed ALNS heuristic. Let $z(S)$ denote the cost of a feasible solution S given by (1), and $T > 0$ be the current *temperature* of the process, initially set to T_{start}. A solution S_i^+ is always accepted over an existing solution S_i if $z(S_i^+) < z(S_i)$; otherwise it is accepted with probability $\exp\left(-\frac{[z(S_i^+) - z(S_i)]}{T}\right)$. Similar to Ropke and Pisinger [39], we set T_{start} such that S_i^+ is accepted with a probability of 0.5, if it is μ percent worse than S_i. We refer to μ as the *start temperature control parameter*. Furthermore, the current temperature T is reduced gradually in every iteration
using the expression $T = \varepsilon T$, where $0 < \varepsilon < 1$ is a fixed *cooling rate*. The ALNS terminates after a total of $\mathcal{N}$ iterations and reports S^*.

4.1.8. Applying noise

Some insertion heuristics tend to make locally best moves and can result the algorithm in getting stuck in a local optimum [39]. So, we introduce a noise term to the objective function to avoid this situation. We use two classes of insertion heuristics, those that are *clean*, which consider the true objective function for insertion purposes, and those that are *noise-imposed*, which consider $noisedcost = truecost + \mathbf{3}\,\partial \max_{i,j \in N}\{c_{ij}\}$ where $\mathbf{3}$ is a noise parameter used for diversification and $\partial \in [-1, 1]$ is a random number. We again employ a roulette wheel procedure similar to (23) after an insertion heuristic is selected, and track the performance of clean and noise-imposed heuristics.

We now describe the removal, vendor selection/allocation, and insertion algorithms.

4.2. Removal Algorithms

The destroy mechanism of the proposed ALNS framework uses one of the removal algorithms in $\mathcal{A}_R$ given in this section. Here, an input solution S_i in iteration i is destroyed to produce S_i^- by removing n_i customers, which are added to a removal list $\mathcal{L}$. The pseudo-code of the general removal procedure is presented in Algorithm 1. The parameter θ denotes the maximum number of trial iterations, each of which attempts to remove a subset of customers using a removal algorithm r. After each removal, the routes are updated. Each iteration of θ considers the resulting route structure from the previous iteration. The procedure is repeated until a total of n_i customers have been removed. If a premium customer is removed, its associated vendors are also removed unless a vendor serves another (non-removed) premium customer.

For the destroy phase of our ALNS heuristic, we adaptively employ a total of 13 algorithms. The first 10 (*Random Removal*, *Worst-Distance Removal*, *Worst-Time Removal*, *Shaw Removal*, *Proximity-Based Removal*, *Time-Based Removal*, *Demand-Based Removal*, *Historical Node Removal*, *Neighborhood Removal*, *Node Neighborhood Removal*) are adopted from Ropke and Pisinger

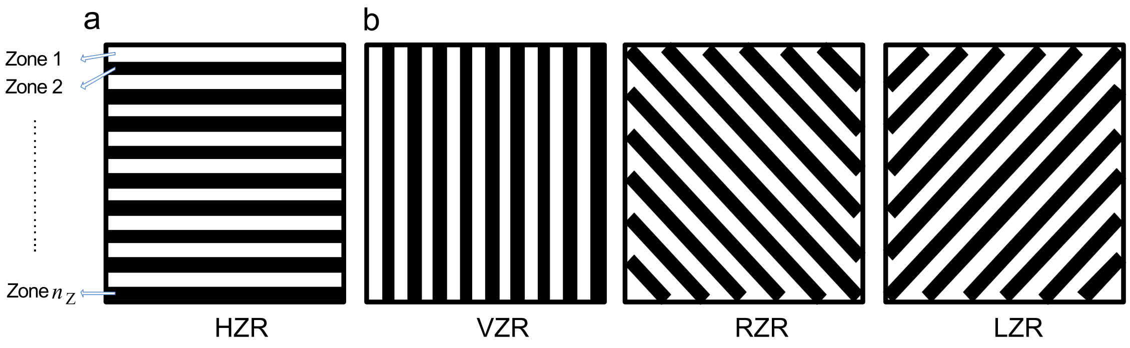

Fig. 2. Zone removal algorithm variants.

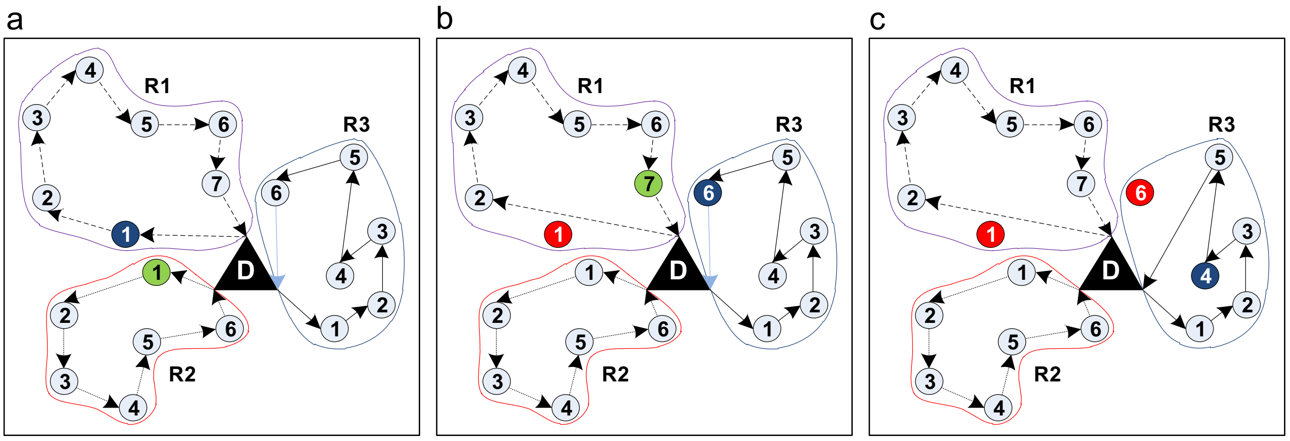

Fig. 3. Route Neighborhood Removal Algorithm. (a) $R_l = R2$, (b) $R_l = R1$ and (c) $R_l = R3$.

[39,40], Pisinger and Ropke [31], and Demir et al. [14]. Hence we provide their descriptions in Appendix A. Here we describe *Route Removal (RoR)* and *Zone Removal (ZR)*, which involve some improvements over the existing algorithms, and *Route Neighborhood Removal (RNR)*, which is a new algorithm we propose.

4.2.1. Route Removal (RoR)

This algorithm removes one or more complete routes from S_i. Let $\mathfrak{R} \subseteq \mathbb{R}$ be a set of randomly selected and removed routes, where $n^{\mathfrak{R}} = \sum_l n^{\mathcal{R}_l}$ is the total number of customers on these routes and the removal of the last randomly selected route in $\mathfrak{R}$ causes $n^{\mathfrak{R}}$ to become greater than or equal to n_i for the first time. We further distinguish the *Single Route Removal* (*SRoR*) case where $\mathfrak{R} = \{\mathcal{R}_1\}$ with $n^{\mathfrak{R}} \geq n^{\mathcal{R}_1}$, and the *Multiple Route Removal* (*MRoR*) case where $|\mathfrak{R}| > 1$. The algorithm alternatingly seeks to use one of these two removal schemes using the roulette wheel mechanism similar to (23).

```
    Algorithm 1. The general structure of a removal algorithm
    Input: $r \in \mathcal{A}_R, S_i$, and maximal number of iterations $\theta$
1   Initialize$\mathcal{L} \longleftarrow$
2   While $\theta > 0$ do
3       Apply $r$to find a subset $\Psi \subseteq C$of customers for removal
4       $\mathcal{L} \longleftarrow \mathcal{L} \cup \Psi$
5       Remove each customer$c \in \Psi from S_i$
6       If$c \in C^P$then
7       Remove external vendor$v \in V$that servers customer$c$
  (if service feasiblity of other premium customers not violated)
8           End if
9           $\theta = \theta - 1$
10  End while
11  Return $S_i^-$ and$\mathcal{L}$
```

4.2.2. Zone Removal (ZR)

This algorithm improves the zone removal operator used by Demir et al. [14]. It operates on a region with Cartesian coordinate system extracted from the underlying network. The corner points of this region are computed during initial preprocessing and the area is then divided into smaller *zones* with respect to a predefined *zone direction*, from which customers are to be removed. Based on the zone direction, the *ZR* algorithm is performed using one of the following four methods: (i) *Horizontal Zone Removal* (*HZR*), (ii) *Vertical Zone Removal* (*VZR*), (iii) *Right-Sided Zone Removal* (*RZR*), and (iv) *Left-Sided Zone Removal* (*LZR*), which are illustrated in Fig. 2. Using a roulette wheel mechanism similar to (23), the *ZR* algorithm chooses one of these four to execute. Let $\mathbb{Z}$ be the chosen set of zones, and let $\mathcal{Z} \subseteq \mathbb{Z}$ be the set of randomly selected zones from which customers will be removed, where $n^{\mathcal{Z}} = \sum_l n^{Z_l}$ is the total number of customers in these zones and the removal of the customers in the last randomly selected zone $\overline{\mathcal{Z}}$ in $\mathcal{Z}$ causes $n^{\mathcal{Z}}$ to become greater than or equal to n_i for the first time. The ZR algorithm iteratively removes all customers in each zone $Z \in \mathcal{Z}\{\overline{\mathcal{Z}}\}$. Customers in zone $\overline{\mathcal{Z}}$ are sorted in non-decreasing order of their distance to the geographic centroid of $\overline{\mathcal{Z}}$ and the closest $\left(n_i - \sum_{Z_k \neq \overline{\mathcal{Z}}} n^{Z_k}\right)$ customers are removed from zone $\overline{\mathcal{Z}}$.

4.3. Route Neighborhood Removal (RNR)

This algorithm integrates some feasibility concerns into a distance-based procedure to increase the efficiency of the removal process in connection with the insertion algorithm that follows. Let $\mathbb{R}$ be the unordered set of all routes in S_i, and $\mathcal{R}_l, \mathcal{R}_j \in \mathbb{R}$ be two different routes with their ordered list of stops $\mathcal{R}_l = \left\{l_1, \ldots, l_{|\mathcal{R}_l|}\right\}$

and $\mathcal{R}_j = \left\{j_1, \ldots, j_{|\mathcal{R}_j|}\right\}$. Suppose we select a customer $l_m \in \mathcal{R}_l$ and a node $j_n \in \mathcal{R}_j$ to create an *eligible node pair* (l_m, j_n) such that the modified route $\mathcal{R}_l^* = \{l_1, \ldots, l_m, j_n, l_{m+1}, \ldots, l_{|\mathcal{R}_l^*|}\}$ still ensures time window as well as load feasibility. At each removal iteration, the RNR algorithm randomly selects a route $\mathcal{R}_l \in \mathbb{R}$ and seeks for an *eligible node pair* (l_m, j_n) that minimizes $(c_{l_m, j_n} + c_{j_n, l_{m+1}})$. If the algorithm is able to find such a pair, then j_n is removed from $\mathcal{R}_j$ and added to the removal list $\mathcal{L}$. Otherwise, a customer $l_r \in \mathcal{R}_l$ is randomly selected and removed from $\mathcal{R}_l$. This algorithm executes $\theta = n_i$ iterations while updating routes at the end of each iteration. A node-to-node distance matrix is generated during the initial preprocessing to increase efficiency.

Fig. 3 illustrates the working mechanism of the *RNR* algorithm. There are three routes $\mathbb{R} = R1, R2, R3$. In Fig. 3a, route $R2$ is randomly selected as R_l. Then, an eligible node pair (l_m, j_n) where $l_m \in R2$ and $j_n \in R1$ or $j_n \in R3$ is searched. Such a pair is found for $l_1 = 1$ and $j_1 = 1$ when $\mathcal{R}_j = R1$. Then, $j_1 = 1$ is removed from $R1$ and the solution is updated as shown in Fig. 3b. In the same figure $R1$ is randomly selected as R_l and the pair $l_6 = 7$ and $j_6 = 6$ for $\mathcal{R}_j = R3$ is identified. Then, $j_6 = 6$ is removed from $R3$ and the solution is updated as shown in Fig. 3c. Finally in Fig. 3c, $R3$ is randomly selected as R_l and no eligible pair of nodes exists. As a result, $l_4 = 4$ is randomly selected for removal.

4.4. Vendor Selection/Allocation Algorithms

The algorithms described in this section are crucial auxiliary actors employed in the repair phase of our proposed ALNS framework, and aim to allocate a new vendor set for a customer removed in the destroy phase. Each algorithm operates on the partial feasible solution S_i^- obtained after the removal phase. Let $V_j \subseteq V$ be a set of vendors who can supply premium products $P_j = \left\{p_1, \ldots, p_{|P_j|}\right\} \subseteq P$ ordered by premium customer $j \in C^P$. Also, let $P_v \subseteq P$ denote a set of premium products offered solely by vendor $v \in V$. Finally, let O_j^p be a sorted list of all eligible vendors $v \in V_j$ that can supply premium product $p \in P_j$, in non-decreasing order of their distance to node $j \in C^P$. These O_j^p lists are generated during initial preprocessing to increase the efficiency of distance-based vendor selection/allocation algorithms. We now describe six new vendor selection/allocation algorithms we developed and used in our study.

Algorithm 2. The general structure of the *NNVS* algorithm

```
Input: $j \in \left(C^P \cap \mathcal{L}\right), P_j, P_v$ for each $v \in V$, and $O_j^p$ for each $p \in P_j$
1  Initialize $V_j \longleftarrow \varnothing$ and $d(V_j) = \infty$
2  For each $p \in P_j$ do
3  Initialize $\mathcal{H}$, set of premium products not yet supplied,
   $\mathcal{H} \longleftarrow P_j$
4  Initialize temporary output $\Upsilon_j$ for $V_j$, $\Upsilon_j \longleftarrow \varnothing$
5  Select the closest vendor $v$ supplying $p$ as $v = O_j^p[0]$
6     $\Upsilon_j \longleftarrow \Upsilon_j \cup \{v\}$
7     $\mathcal{H} \longleftarrow \mathcal{H}(\mathcal{H} \cap P_v)$
8  While $\mathcal{H} \neq$ do
9  Select an unsupplied premium product $p_u \in \mathcal{H}$
10 Select vendor $v = O_j^{p_u}[0]$
11    $\Upsilon_j \longleftarrow \Upsilon_j \cup \{v\}$
12    $\mathcal{H} \longleftarrow \mathcal{H}(\mathcal{H} \cap P_v)$
13 End while
14    If $d(\Upsilon_j) < d(V_j)$ then
15       $V_j \longleftarrow \Upsilon_j$
16    End if
17 End for
18 Return $V_j$
```

4.4.1. Node Neighborhood Vendor Selection (NNVS)

This algorithm ensures that each removed premium customer receives service from the closest eligible vendors in the temporary feasible solution S_i^+. The algorithm generates a new feasible set V_j for each customer $j \in (C^P \cap \mathcal{L})$ such that $d(V_j) = max_{v \in V_j}\{c_{vj}\}$, is minimized. A pseudo-code of the NNVS heuristic is presented in Algorithm 2. The algorithm starts with a set of input parameters mostly related to the target customer and Steps 3-15 are repeated for each $p \in P_j$ to find a new feasible V_j that has the least $d(V_j)$ amount for the target customer.

4.4.2. Route Neighborhood Vendor Selection (RNVS)

This algorithm considers the existing routes in S_i^- while generating a new feasible V_j for each removed premium customer $j \in (C^P \cap \mathcal{L})$. Let $\mathbb{R}$ be the set of all routes in S_i^- and $\mathcal{R}_l = \left\{l_1, \ldots, l_{|\mathcal{R}_l|}\right\} \in \mathbb{R}$ be a route with unsorted nodes. The RNVS algorithm aims to identify a new vendor set V_j whose elements are located around $\mathcal{R}_l$ such that $d(V_j) = max_{v \in V_j}\left\{min_{k \in \mathcal{R}_l}\{c_{vk}\}\right\}$ is minimized. Let $v_k = O_j^p[0]$ be the closest vendor to node $k \in \mathcal{R}_l$, which supplies $p \in P_j$. RNVS takes $\mathcal{R}_l$ and O_j^p for each $p \in P_j$ and $k \in \mathcal{R}_l$ as additional inputs and only differs in Steps 5 and 10 of the pseudo-code in Algorithm 2. In those steps, the closest vendor v is selected as $v = argmin_{v_k; k \in \mathcal{R}_l}\{c_{v_k j}\}$ by the RNVS algorithm. Note that RNVS has a potential to produce different results for each selected route $\mathcal{R}_l \in \mathbb{R}$.

4.4.3. Node Neighborhood Vendor Selection with Noise (NNVSN)

This algorithm is an adaptation of NNVS and takes into account a degree of freedom in the closest vendor allocation processes to avoid the selection of the same vendor sets repeatedly. In Steps 5 and 11 of the algorithm in Algorithm 2, it selects the vendor in position $\left\lfloor \lambda \left| O_j^p \right| \psi \right\rfloor$ of the list, where $\lambda \in [0,1]$ is a random number and $\psi \in [0,1]$ is a *freedom percentage* parameter for the vendor selection/allocation algorithms.

4.4.4. Route Neighborhood Vendor Selection with Noise (RNVSN)

This algorithm also incorporates the degree of freedom concept described above. Using the same definitions with RNVS and NNVSN, let v^r be the vendor in position $r = \left\lfloor \lambda \left| O_j^p \right| \psi \right\rfloor$ of the list O_j^p as the rth closest vendor to node $k \in \mathcal{R}_l$ which supplies product $p \in P_j$. Then, the closest vendor v is selected as $v = argmin_{v^r; k \in \mathcal{R}_l}\{c_{v^r k}\}$ by the RNVSN algorithm.

Algorithm 3. The general structure of the *RVS* algorithm

```
Input: $j \in \left(C^P \cap \mathcal{L}\right), P_j$, and $P_v$ for each $v \in V$
1  Initialize $V_j \longleftarrow \varnothing$
2  Initialize $\mathcal{H}$, set of premium products not yet supplied,
   $\mathcal{H} \longleftarrow P_j$
3  While $\mathcal{H} \neq$ do
4  Select a random vendor $\boldsymbol{v} \in V$ for which $(\mathcal{H} \cap P_v) \neq \varnothing$
5     $V_j \longleftarrow V_j \cup \{v\}$
6     $\mathcal{H} \longleftarrow \mathcal{H}(\mathcal{H} \cap P_v)$
7  End while
8  Return $V_j$
```

4.4.5. *Random Vendor Selection (RVS)*

This algorithm randomly allocates a new external vendor set to each removed premium customer j to diversify the vendor selection procedure. The pseudo-code is given in Algorithm 3.

Algorithm 4. The overall structure of the ALNS algorithm with simulated annealing

```
    Input: $\mathcal{A}_R, \mathcal{A}_V, \mathcal{A}_I, \mathcal{N}, \mathcal{N}_S, \mathcal{N}_{IWI}, \mu, \varepsilon$
1   Generate an initial solution $S_0$ using Greedy or Regret−2Heuristic
2   Initialize $P_r^s, P_v^s, P_i^s$ for each $r \in \mathcal{A}_R$, $v \in \mathcal{A}_V$, and $i \in \mathcal{A}_I$
3   Initialize $T$ and $T_{Start}$ using $z(S_0)$ and $\mu$
4   Let $i$ be the outmost iteration counter initialized as $i \longleftarrow 1$
5   Let $S_i \longleftarrow S^* \longleftarrow S_0$
6   Let $j$ be the counter of iterations without improvement in $S^*$, $j \longleftarrow 0$
7   While $i \le \mathcal{N}$ do
8       If $j \ge \mathcal{N}_{IWI}$ then
9           Generate a new $S_0$ by adaptively selecting Greedy or Regret−2Heuristic
10          Let $S_i \longleftarrow S_0$
11      End if
12      Select a removal algorithm $r \in \mathcal{A}_R$ with probability $P_r^s$
13      Generate $S_i^-$ by applying algorithm $r$ to $S_i$
14      Select a vendor selection/allocation algorithm $v \in \mathcal{A}_V$ with probability $P_v^s$
15      Select an insertion algorithm $i \in \mathcal{A}_I$ with probability $P_i^s$
16      Generate $S_i^+$ by applying $i$ together with $v$ to $S_i^-$
17      If $z(S_i^+) < z(S_i)$ then
18          $S_i \longleftarrow S_i^+$
19      Else
20          Let $v = exp\,(-(z(S_i^+) - z(S_i))/T)$
21          Generate a random number $\ni \in [0, 1]$
22          If $\ni < v$ then
23              $S_i \longleftarrow S_i^+$
24          End if
25          End if
26        If $z(S_i) < z(S^*)$ then
27                $S^* \longleftarrow S_i$
28                $j \longleftarrow 0$
29          Else
30              $j \longleftarrow j+1$
31            End if
32            If $i \equiv 0 (mod \mathcal{N}_S)$ then
33                  Update all $P_\cdot^s$ probabilities using the adaptive weight procedure
34          End if
35          $T \leftarrow T\ \varepsilon$
36          $i \leftarrow i+1$
37  End while
38  Return $S^*$
```

4.4.6. *Historical Vendor Selection (HVS)*

This algorithm uses historical information when generating the set V_j. Let $V_j^l \subseteq V$ be a set of external vendors in S_i^+ able to satisfy premium product orders of customer $j \in C^P$ at iteration $l = 1 \ldots \mathcal{N}$. Let $\mho_l = V_j^l \cup \{j\}$ and $f_j^l = \sum_{z \in \mho_l} (c_{z_p z} + c_{z z_s})$ be the *service cost* of customer j at iteration l, where $z_p \in N$ and $z_s \in N$ are preceding and succeeding nodes to $z \in \mho_l$, respectively. HVS keeps a history of f_j^l and if used in iteration i, it returns vendor set V_j with the best known service cost up until iteration i.

4.5. *Insertion Algorithms*

In the final repair phase of the ALNS framework, the insertion algorithms described next convert a partially destroyed feasible solution S_i^- into a complete feasible solution S_i^+ by re-inserting each customer in $\mathcal{L}$. Throughout this repair process, time window, capacity, and vendor precedence constraints are taken into account to ensure feasibility. If feasibility cannot be achieved with the routes already in S_i^-, a new route is created. The first two algorithms (*Greedy Insertion*, *Zone Insertion*) are adopted from Ropke and Pisinger [39], Pisinger and Ropke [31], and Demir et al. [14], hence we provide their descriptions also in Appendix A. Here we only describe *Regret-k Insertion* and *Greedy Insertion with New Route Openings* algorithms that we propose:

4.5.1. *Regret-k insertion (R-kI)*

Since the Greedy Insertion algorithm generally delays the insertion of customers with large *insertion costs, R -kI* tries to overcome this behavior by considering a "look-ahead" information to decide which customer to insert next [39]. Let $\mathcal{R}_{jk} \in \{1, \ldots, |\mathbb{R}|\}$ correspond to the route for which customer $j \in \mathcal{L}$ has the kth lowest insertion cost. We then define a *regret-k value* Ψ_j^k for each customer $j \in \mathcal{L}$ as the difference in the cost of inserting customer j in its best route and its kth-best route. The *R-kI* algorithm selects a customer $j^* = argmax_{j \in \mathcal{L}}\left\{\Upsilon_j^k\right\}$ to insert along with its corresponding vendors $v \in V_{j^*}$, if any, at their minimum cost positions. In case of a tie, the customer with the lowest insertion cost is selected. Both *clean* and *noise-imposed* versions of *Regret-2*, *Regret-3*, *Regret-4* and *Regret-m* $(m = |\mathbb{R}|)$ insertion algorithms are implemented.

4.5.2. *Greedy Insertion with New Route Openings (GIN)*

This algorithm is a slightly modified version of *GI* in terms of customer insertion policy. Let $\mathbb{R}(S_i^-)$ and $\mathbb{R}(S^*)$ be the set of all routes in S_i^- and S^*, respectively. Also, let ξ be an integer parameter that represents a *maximum route allowance*. At each iteration, *GIN* selects a customer j^* with the lowest insertion cost. However, if $\left|\mathbb{R}(S_i^-)\right| \le \left|\mathbb{R}(S^*)\right| + \xi$ then it inserts customer j^* along with its vendors $v \in V_{j^*}$, if any, in a newly created route with a *new route opening probability* γ.

The master-level overview of the proposed ALNS heuristic is described in Algorithm 4.

5. Computational Study

We now present the results of our computational experiments to test the performance of the proposed ALNS approach. We start by parameter tuning for the ALNS. We then validate the performance of our ALNS against the published results for the well-known 100-node Solomon benchmark instances for VRPTW. Finally, we test our ALNS on 25-, 50- and 100-node benchmark instances we adapted to the EDRP using Solomon instances[1]. With the exception of 25-node instances, which can be solved via CPLEX, we present our results as benchmarks for future studies. All experiments are performed on a computer equipped with Intel Core2 Quad 2.40 GHz CPU (Q6600) and 4 GB RAM using the Java language.

5.1. *Parameter Tuning*

We utilize the following 100-customer VRPTW benchmark instances of Solomon to determine the parameter values: R104, R112, R201, R204, R207, RC104, RC106, RC206, and RC207. Although these instances do not involve vendors and premium customers, we assume the resulting parameter setting will likely perform well for our newly generated EDRP instances described below. This is because all parameters except ψ, the *freedom percentage* parameter for vendor selection/allocation algorithms, are common for both regular and premium customers. The parameter ψ is determined in a similar fashion after all other parameters have been tuned.

Our tuning methodology is similar to that of Ropke and Pisinger [39]. We set the initial values of the parameters as described in Ropke and Pisinger [39,40], Pisinger and Ropke [31], and Demir et al. [14] and perform 10 runs on nine tuning instances considering up to 10 values for each parameter. We calculate the average percent deviations from the average of the corresponding best known solutions, determine the value that yields the least average percent deviation, and fix the parameter. We repeat this procedure until all parameter values have been tuned. The parameters and their final selected values are summarized in Table 1. The tuning procedure is detailed in Appendix B.

Table 1
Parameters used in the proposed ALNS algorithm.

Parameter description	Parameter value
Maximum number of iterations ($\mathcal{N}$)	25000
Number of iterations for roulette wheel ($\mathcal{N}_S$)	100
Maximum number of iterations without improvement to refresh initial feasible solution ($\mathcal{N}_{IWI}$)	4000
Roulette wheel reaction factor (ρ)	0.1
New global solution score (σ_1)	20
Better solution score (σ_2)	16
Worse solution score (σ_3)	13
Start temperature control parameter (μ)	0.05
Cooling rate (ε)	0.9998
Noise parameter (3)	0.025
Lower limit of the number of customers to remove ()	$min\{0.1|N|, 30\}$
Upper limit of the number of customers to remove ($\overline{n}_i$)	$min\{0.4|N|, 60\}$
First Shaw parameter (ϕ_1)	9
Second Shaw parameter (ϕ_2)	3
Third Shaw parameter (ϕ_3)	5
Fourth Shaw parameter (ϕ_4)	2
Determinism factor for Shaw removal operators (η)	6
Determinism factor for worst removal operators (κ)	3
Number of zones (n^z)	11
New route opening probability (γ)	0.2
New route opening allowance (ξ)	2
Freedom percentage for vendor selection processes (ψ)	0.35

Similar to Ropke and Pisinger [39] and Demir et al. [14] our sensitivity analysis indicates that 25,000 iterations are enough to get good quality solutions in reasonable time. It is worth noting that the ALNS performs best when the number of customers to remove, n_i, is chosen randomly between a lower limit and an upper limit $\overline{n}_i = min\{0.4|N|, 60\}$. Moreover, experiments show that the performance of the insertion algorithms decreases as the upper limit $\overline{n}_i$ increases, whereas decreasing $\overline{n}_i$ mostly results in minor improvements. Since we introduce additional diversification mechanisms, our setting of the parameters σ_1, σ_2, and σ_3 is consistent with the expected setting $\sigma_1 \ge \sigma_2 \ge \sigma_3$ to reward an operator for good performance, unlike Ropke and Pisinger [39] and Demir et al. [14] where achieving a worse solution is rewarded more than finding a better solution.

5.2. *Experiments on VRPTW Instances*

5.2.1. *Computational Results*

We now validate our proposed ALNS on 100-node Solomon instances for the VRPTW. We use the truncated data to be able to compare our results to those given in Pisinger and Ropke [30] and Demir et al. [14] as well as to the optimal solutions reported in Roberti [37]. For each instance we perform 10 runs. Since there are no publicly available instances for the EDRP, our conjecture is that if the proposed ALNS works well for the VRPTW, we would expect it to work also well for the EDRP instances we generate and test.

The results are summarized in Table 2. *PR* and *DBL* represent Pisinger and Ropke [30] and Demir et al. [14], respectively. “Distance” refers to distance and “Optimal” refers to the optimal solution. Each column in the table reports the average values for each class of problem. % Deviation shows the average gap between the distances obtained by ALNS and those reported in the benchmark results. A negative value indicates that ALNS performs better. The detailed results are presented in Appendix C.

For the clustered problem sets C1 and C2, the proposed ALNS finds optimal solutions for all instances. For the remaining problem sets, we see that the optimality gaps are larger in problem

[1] The supermarket chain that the EDRP is motivated from operates multiple stores in big cities, scattered in different districts and each store is assigned distinct regions to service the online orders. The 100-node benchmark instances are of comparable size considering the delivery operations of the company. A brief description of the implementation of ALNS using the ArcGIS environment is provided at: http://people.sabanciuniv.edu/catay/EDRP

Table 2
Summary of results for VRPTW instances.

Class	Optimal	PR			DBL		The Proposed ALNS			Deviation from (%)			
	Distance	Best dist	Avg dist	Avg Time (s)	Best dist	Avg time (s)	Best dist	Avg dist	Avg time (s)	Optimal	PR best	DBL best	PR avg
R1	1173.6	1174.8	1177.4	32.8	1174.5	45.1	1174.1	1177.1	51.1	0.05	−0.06	−0.04	−0.03
C1	826.7	826.7	826.7	32.1	826.7	40.6	826.7	826.7	43.4	0.00	0.00	0.00	0.00
RC1	1334.5	1336.8	1342.2	30.3	1336.7	42.3	1334.7	1340.5	51.4	0.01	−0.16	−0.15	−0.14
R2	872.5	875.6	878.9	62.6	874.6	75.6	875.1	878.8	43.5	0.29	−0.04	0.04	0.00
C2	587.4	587.4	587.4	75.4	587.4	86.4	587.4	587.4	40.6	0.00	0.00	0.00	0.00
RC2	1000.7	1002.1	1009.8	53.6	1001.9	70.3	1002.3	1009.0	43.6	0.15	0.00	0.03	−0.11
All	**973.2**	**974.6**	**977.7**	**47.3**	**974.3**	**59.4**	**974.1**	**977.3**	**45.8**	**0.09**	**−0.04**	**−0.02**	**−0.04**

Table 3
Statistics for the removal and insertion algorithms.

Algorithm	Average % usage	Average time (ms)	Average % freq new best dist found	Average % dev in avg dist	Average % dev in min dist
RR	9.63	0.02	16.57	0.22	0.15
WDR	7.09	0.28	13.54	0.10	0.00
WTR	10.23	0.29	19.93	0.15	0.02
SR	9.01	0.27	15.31	0.20	0.07
PR	7.79	0.22	12.26	0.11	−0.01
TR	9.91	0.23	18.26	0.10	0.01
DR	10.21	0.24	17.35	0.18	0.10
HR	2.65	0.54	11.39	0.12	0.12
NR	6.03	0.39	16.72	0.16	0.07
NNR	4.47	0.01	11.05	0.10	−0.04
RoR	4.38	0.01	7.72	0.24	0.04
ZR	8.49	0.01	16.14	0.24	0.14
RNR	10.11	1.25	23.15	0.24	0.14
GI	12.93	1.07	13.36	0.07	0.07
R-2I	24.71	1.68	24.82	0.18	0.04
R-3I	23.45	1.68	23.26	0.20	0.11
R-4I	21.75	1.67	21.00	0.18	0.05
R-mI	14.76	1.61	17.25	0.03	0.00
ZI	0.68	0.32	2.28	0.16	0.09
GIN	1.72	1.83	5.39	0.10	−0.01

sets R2 and RC2 where the time windows are wider. Nevertheless, ALNS is able to provide competitive results, as the average gap between the optimal distances and our best distances is only 0.09%. These results also indicate that the proposed ALNS produces better results than both *PR* and *DBL* with similar computational effort. ALNS has also improved the upper bound of problem R208, the only unsolved Solomon instance, from 701.2 to 701 (see Appendix C), which has been recently proven to be optimal [38].

We tested our algorithm on the real-numbered data as well. Our ALNS shows a similar performance with an average deviation of 0.09% from the best-known distances given in Yildirim and Çatay [48]. We also note that ALNS has improved the best-known distances of the following five instances: R106, R107, R108, R210 and RC107. We report them in Appendix C.

5.2.2. *Sensitivity Analysis of the Removal and Insertion Algorithms*

In addition to the performance benchmark analysis above, we have also investigated the performance of the removal and insertion algorithms on the truncated VRPTW instances of Solomon. Table 3 presents the statistics on the different algorithms after 10 runs. The first column denotes the algorithm, the second column shows the number of iterations the corresponding algorithm is used as a percentage of 25,000, the third column is the average time that the algorithm takes in milliseconds, and the forth column gives the percentage of times the algorithm yielded a new best distance. The results show that Demand-Based Removal (DR), Worst-Time Removal (WTR), and Route Neighborhood Removal (RNR) mechanisms are the most frequently used removal algorithms while Random Removal (RR), Shaw Removal (SR) and Time-Based Removal (TR) are close followers. In terms of their frequency of discovering a new best distance, the most promising algorithm is Route Neighborhood Removal (RNR) followed by Worst-Time Removal (WTR).

Among the repair algorithms we observe that Zone Insertion (ZI) and Greedy Insertion with New Route Openings (GIN) are utilized very rarely and their contribution in finding new best solutions seems limited. On the other hand, an algorithm with low utilization or with less contribution in finding a new best solution does not mean that ALNS would benefit from its exclusion because it may allow it to escape from a local optimum and find an improved solution in the future iterations. In order to see whether we can leave out some of these algorithms, we performed additional experiments on the subset of instances which we used in the parameter tuning by excluding each algorithm at a time while keeping the others. The fifth and sixth columns report the average percentage degradation in the average and best achieved distances, respectively, without using the corresponding algorithm. A positive value means degradation in the solution quality. These results suggest that RR, ZR and RNR are the most useful removal algorithms whereas among the insertion algorithms Regret-3 Insertion (R-3I) and ZI are most useful. Although the performances of Proximity-Based Removal (PR), Node Neighborhood Removal (NNR) and GIN mechanisms in terms of the best distances achieved may be interpreted as they are not necessary, their existence helps ALNS find better solutions in general (on average) and excluding any of them would degrade the solution quality by 0.1%.

In light of the results above, we conclude that all of the destroy and repair algorithms contribute positively in obtaining high quality solutions for VRPTW and the proposed ALNS is competitive enough that we can expect our solution approach to work well for the EDRP problem instances that we solve next.

5.3. *Experiments on EDRP Instances*

In this section, we generate benchmark instances by adapting 25-node, 50-node, and 100-node Solomon problems to the EDRP setting and solve them using the proposed ALNS. For the 25-node instances, we solve the model formulated in Section 3.3 using CPLEX and compare them to the results found by ALNS. For larger instances, we present our results as benchmarks for future studies. The generated data is available at: http://people.sabanciuniv.edu/catay/EDRP_Instances.zip

5.3.1. *Generation of EDRP Instances*

We introduce a new set of benchmark instances for EDRP based on modifications of the original Solomon VRPTW instances. The VRPTW instances are classified in three sets C, R, and RC with respect to the clustered, random, and random-clustered geographical distribution of the customers, respectively. We name the modified new EDRP data sets as M-C, M-R, and M-RC, respectively.

Table 4
Structure of new EDRP instances.

Categories	p^v(%)	p^{pc}(%)	$\|P\|$
Category 1	12.5	20.0	2
Category 2	12.5	20.0	3
Category 3	12.5	20.0	4
Category 4	20.0	$33.\bar{3}$	2
Category 5	20.0	$33.\bar{3}$	3
Category 6	20.0	$33.\bar{3}$	4
Category 7	12.5	$33.\bar{3}$	3
Category 8	20.0	20.0	3
Category 9	$16.\bar{6}$	25.0	3
Category 10	25.0	50.0	3
Category 11[a]	12.5	20.0	1

[a] Category 11 is only generated for instances with 25 customers

Table 5
Summary of results for 25-node EDRP instances.

	M-C1	M-C2	M-R1	M-R2	M-RC1	M-RC2	All
# of instances	99	88	132	121	88	88	616
# CPLEX cannot solve	15	0	64	0	62	0	141
CPLEX avg optimality gap (%)	63	28	46	30	102	123	65
Avg % Gap between CPLEX UB & ALNS best distances	−1.86	−1.30	−1.34	−2.34	−0.22	−6.00	−2.18
Avg % Gap between CPLEX UB & ALNS avg distances	−1.85	−1.30	−1.34	−2.30	−0.02	−5.91	−2.15

Each set consists of two types of problems involving customers having narrow (type 1) and wide (type 2) time window lengths.

Let n be the number of nodes, s_i be the service duration and d_i be the demand of customer i of an original instance. To modify the instance, we first designate a percentage p^v of n nodes to be randomly selected external vendors. We then nominate a percentage p^{pc} of the remaining nodes as premium customers, again to be randomly selected. As a result, the remaining $\lfloor n(1-p^v)(1-p^{pc})\rfloor$ nodes are identified as regular customers. We assume the same time window for vendors and depots, and identical service duration s_i for each vendor. The time windows for premium customers are adjusted if they lead to any infeasibility. In addition, the size of the premium product set, the unit volume of each premium product, and vendor-product compatibility matrix are randomly set between lower and upper bounds. We also ensure that each vendor supplies at least one premium product. The matrices $B_{|C^p|\times|P|}$ and $Q_{|C^p|\times|P|}$ are also randomly populated between pre-specified lower and upper bounds such that the sum of the demand for a premium product, i.e. $\sum_{p\in P} q_{cp} w_p$, and modified demand for the standard product of each premium customer equals to the corresponding demand d_i in the original instance.

We have generated our data in 11 different categories for each truncated VRPTW instance of Solomon with 25, 50 and 100 customers by varying p^v, p^{pc} and $|P|$ values. The structure of each category type is described in Table 4. Category 11 data involving only one premium product is relatively simpler and is generated for only 25-customer case for comparison purposes against CPLEX results. A total of $(3\text{sizes})\times$ $(10\text{categories}) \times (56\text{instances}) + (1\text{size}) \times (1\text{category}) \times (56\text{instances}) = 1736$ instances are generated. All instances are available online for future studies.

Table 6
Statistics for the vendor selection/allocation algorithms.

Algorithm	**Average % usage**	**Average time (ms)**	**Average % freq new best dist found**	**Average % dev in avg dist**	**Average % dev in min dist**
NNVS	8.75	0.01	7.91	−0.23	−0.14
RNVS	39.45	0.15	14.78	0.03	−0.13
NNVSN	1.25	0.01	2.14	−0.07	0.07
RNVSN	38.00	0.15	14.77	−0.15	−0.17
RVS	2.48	0.01	3.56	−0.21	−0.01
HVS	9.08	0.01	9.59	−0.41	−0.27

5.3.2. *Computational Results*

In this section, we first apply the proposed ALNS to 25-node EDRP instances and compare the best results achieved in 10 runs to the solutions obtained by CPLEX 12.2 in three hours. Since CPLEX cannot even find feasible solutions for most instances we set the MIP emphasis parameter to feasibility in order to emphasize feasibility over optimality. By doing so, we were able to find the optimal solutions for 475 instances out of 616. We have also observed that the performance of CPLEX in terms of finding a good feasible solution deteriorates as the number of premium products, vendors and premium customers increase.

In Table 5 we summarize the performance of ALNS in comparison with CPLEX. We observe that M-C2 and M-RC2 problem sets are relatively easier as an upper bound is found for each instance whereas CPLEX struggles to find a feasible solution for problems in sets M-R1 and M-RC2. Overall, the average solution quality of ALNS is 2.18% and 2.15% better than that of CPLEX considering best distances and average distances, respectively, in 10 runs. These values also indicate the good convergence of ALNS. Considering the large optimality gaps of CPLEX in connection with the gaps between CPLEX and ALNS results, we can conclude that CPLEX cannot efficiently improve its lower bounds. Finally, we report that the average CPLEX run time is 10,172 sec while the average computation time of ALNS is less than 7 sec. The detailed results for each instance in each data category are presented in Tables S1–S11 in Supplemental Material. Here, we only note that the distances achieved by ALNS are better than or same as the upper bounds found by CPLEX for all instances.

Since CPLEX is unable to solve 50- and 100-node instances, we only report our detailed results in Tables S12-14 and S15-17, respectively, in Supplemental Material. Note that average ALNS execution time is 23 sec for 50-node instances and 99 sec for 100-node instances.

5.3.3. *Sensitivity Analysis of the Vendor Selection/Allocation Algorithms*

We present the statistics for the vendor selection/allocation (VSA) algorithms based on the 100-node experiments performed in the previous section in a similar was as in Section 5.2.2. The results are reported in Table 6. The columns are as defined in Table 3. Columns 2–4 present the average results of all problems in 10 categories whereas the average percentage deviations given in the fifth and sixth columns were obtained using a subset of representative problems.

The results clearly show that Route Neighborhood Vendor Selection (RNVS) and Route Neighborhood Vendor Selection with Noise (RNVSN) are clearly the most preferred mechanisms and have the ability to discover new best distances more frequently throughout the search procedure. The deviation percentages, on the other hand, reveal that ALNS may benefit from the removal of some algorithms. To further investigate the impact of excluding an algorithm, we selected Node Neighborhood Vendor Selection (NNVS), RNVSN, and Historical Vendor Selection (HVS) and performed 10 more runs on all Category 6 instances which involve the largest number of premium products. We also analyzed the cases when NNVS & HVS and RNVSN & HVS are

Table 7
Percentage distance deviations of the solutions found with and without using the corresponding algorithm(s).

Category	NNVS		RNVSN		HVS		NNVS & HVS		RNVSN & HVS	
	Min	Avg	Min	Avg	Min	Avg	Min	Avg	Min	Avg
1	0.09	0.10	0.00	−0.01	−0.02	−0.01	0.03	0.02	0.02	−0.03
2	−0.03	0.07	0.00	0.01	0.01	0.01	−0.04	0.03	0.02	0.02
3	0.02	0.05	0.00	−0.02	0.01	−0.01	0.04	0.06	0.01	−0.01
4	0.04	−0.04	0.05	−0.02	0.08	−0.03	0.09	−0.01	0.07	−0.03
5	−0.04	0.03	−0.01	0.06	−0.05	−0.02	−0.07	0.00	0.03	−0.01
6	−0.07	−0.05	−0.06	−0.08	−0.08	−0.12	−0.02	−0.09	−0.02	−0.10
7	0.01	0.07	0.02	0.00	−0.09	−0.05	0.01	−0.01	−0.06	−0.03
8	0.09	0.04	0.04	0.00	0.06	−0.02	0.05	0.00	0.02	−0.02
9	0.13	0.06	0.20	0.06	0.06	0.00	0.04	0.00	0.07	0.01
10	0.00	−0.03	0.12	0.07	0.09	−0.02	0.07	0.02	0.04	−0.02
Average	0.02	0.03	0.04	0.01	0.01	−0.03	0.02	0.00	0.02	−0.02

excluded together. Since these tests provided promising results, we extended our experimental study to cover all the instances in 10 categories and performed 10 runs for each. So, in total we made $(10\text{categories}) \times (56\text{instances}) \times (5\text{exclusions}) = 2800$ runs. We report the average percentage deviation of the average and best distances in Table 7. These results suggest that although the performance of the ALNS might be enhanced by removing some VSA algorithms, these enhancements are rather marginal and observed on some individual instances/categories, and cannot be generalized. Nevertheless, one may achieve better results for instances in a particular category by excluding one or a pair of algorithms as we observed in the case of Category 6 instances.

6. Conclusion and Future Research

In this article, we described the logistics of a collaborative delivery operation between online groceries and external vendors, and modeled the arising routing problem as a 0–1 mixed integer program. To solve this challenging problem named as EDRP, we proposed an effective solution procedure based on the ALNS framework. We validated the effectiveness of our methodology using VRPTW benchmark instances. Then, we generated a new set of instances for the EDRP and reported the results of our computational tests.

Our ALNS approach is inspired from the ideas presented in Ropke and Pisinger [39,40], Pisinger and Ropke [31], and Demir et al. [14]. Yet, we introduce new removal and insertion algorithms and propose modifications on some of the existing algorithms, which enable us to achieve high quality solutions. We also propose new problem-specific algorithms for vendor selection/allocation operations. We believe these algorithms can be efficiently used in similar ALNS approaches for solving similar VRP variants.

Our computational study on both the VRPTW and EDRP instances showed that the proposed ALNS heuristic is very effective in finding near optimal solutions in reasonable time. Indeed with the proposed ALNS, we were able to improve some of the best-known solutions for both the truncated and real numbered VRPTW instances. With the new benchmark instances for EDRP, we observed that the problem gets significantly more difficult as the number of premium products and the number of vendors as well as premium customers increase. More importantly, it is clear that the routing decisions have to be made very quickly in such a business environment. Our computational times reveal that the ALNS algorithm can be practically used as part of a routing and scheduling decision support system.

Further research on the solution methodology may focus on improving the removal and insertion algorithms targeted for external vendors. More specific worst removal cost functions that also take into account the cost of allocated vendors and Shaw removal relatedness measures can be introduced for premium customers. New adaptive scoring mechanisms directly based on removal, vendor selection/allocation, and insertion algorithm combinations may also enhance the solution quality. On the problem side, the multi-depot case is a natural extension as most e-groceries operate in multiple locations. Furthermore, additional business rules such as capacity limits for depots and vendors, different offerings for premium products, and split deliveries to regular and/or premium customers may be addressed as well. From a strategic planning point of view, the contracting and profit sharing issues amongst the e-grocery and the supplier vendors may be investigated and modeled in this new business setting. Furthermore, the decisions concerning whether to accept or reject an order and requested delivery time slot for the accepted deliveries may be studied from a profit maximization perspective as addressed by Campbell and Savelsbergh [11].

Acknowledgments

We thank Gilbert Laporte for coining the name for the E-grocery Delivery Routing Problem.

Appendix A. ALNS Algorithms Adopted from the Literature

A.1 Removal Algorithms

The removal algorithms described below are adapted from Ropke and Pisinger [39,40], Pisinger and Ropke [31] and Demir et al. [14], and used in our proposed ALNS framework.

Random Removal (RR)
This algorithm selects n_i customers at random, removes them one by and by and updates all affected routes accordingly.

Worst-Distance Removal (WDR)

Before any removal, all remaining customers in S_i are sorted in a non-increasing order of their *worst removal costs* defined for customer $j \in C$ as $f_j = |c_{lj} + c_{jk}|$, where l and k represent the preceding and succeeding node on the route for customer j. Given such a sorted list O, the algorithm removes the customer in position $\lfloor \lambda^{\kappa}|O| \rfloor$ of O, where $\lambda \in [0,1]$ is a random number and $\kappa \ge 1$ is the *worst removal determinism factor*, together which introduce enough randomness to avoid repeated removals of the same customers. A low value of κ yields more randomness [39]. After each removal, the list O is updated and the algorithm is repeated until n_i customers have been removed from S_i.

Worst-Time Removal (WTR)

This algorithm is similar to WDR; however, it considers the service start time in relation to the early service time of a customer, and calculate it as $f_j = |T_{jk} - e_j|$, where T_{jk} is the service start time of some vehicle k at customer j and e_j is its early service time.

Shaw Removal (SR)

As introduced originally by Shaw [42], this algorithm attempts to measure how two customers are related to one another, which is then used to determine the next customer to remove. Here we employ a *relatedness measure* Γ_{ij} for a pair of customers $i, j \in C$ as follows:

$$\Gamma_{ij} = \phi_1 c_{ij} + \phi_2 |beg_i - beg_j| + \phi_3 \Omega_{ij} + \phi_4 |\delta_i - \delta_j| \tag{A.1}$$

where $\phi_1, \dots, \phi_4$ are normalized weights known as the *Shaw parameters* and

$$\Omega_{ij} = \begin{cases} -1, \text{ifcustomers}i\text{and}j\text{areinthesameroute} \\ 1, \text{otherwise} \end{cases} \tag{A.2}$$

$$\delta_i = \begin{cases} d_i, \text{if} i \in C^R \\ d_i + \sum\limits_{p \in P} b_{ip} q_{ip} w_p, \text{if} i \in C^P \end{cases} \tag{A.3}$$

As the similarity of a pair of customers increases, Γ_{ij} decreases. We start with a random customer i^* to remove, calculate Γ_{i^*j} for all $j \in C$, and create a non-decreasing ordered list O of remaining customers by Γ_{i^*j}. We then choose the customer in position $\lfloor \lambda^{\eta}|O| \rfloor$ of list O as the next customer to remove, where $\lambda \in [0,1]$ is a random number and $\eta \ge 1$ is the *Shaw removal determinism factor*, similar to κ introduced in the WDR and WTR algorithms.

6.1.1. *Proximity-Based Removal (PR)*

As a special case of the SR algorithm, we set $\phi_1 = 1, \phi_2 = \phi_3 = \phi_4 = 0$ in (A.1).

6.1.2. *Time-Based Removal (TR)*

Again as a special case of SR algorithm, this time we set $\phi_2 = 1, \phi_1 = \phi_3 = \phi_4 = 0$ in (A.1).

Demand-Based Removal (DR)

Similar to PR and TR, we set $\phi_4 = 1, \phi_1 = \phi_2 = \phi_3 = 0$ in (A.1).

Historical Node Removal (HR)

Unlike the other removal algorithms presented above, this heuristic makes use of historical information when removing customers. Let $f_{ji} = c_{lj} + c_{jk}$ be the *position cost* of customer $j \in C$ in iteration $i = 1 \dots \mathcal{N}$, where $l, k \in N$ are the preceding and succeeding nodes of customer j, respectively. Let $f_j^* = min_{m=1,\dots,i-1}\left\{f_{jm}\right\}$ be the best position cost encountered for customer j up to but not including iteration i. The *HR* algorithm removes customer $j^* = argmax_{j \in C}\left\{f_{ji} - f_j^*\right\}$. Then the corresponding route is updated and the next customer j^* is identified. The procedure is repeated until n_i customers have been removed from S_i.

6.1.3. *Neighborhood Removal (NR)*

Let $\mathbb{R}$ be the set of all routes in S_i and let $\mathcal{R} = \left\{\gamma_1, \gamma_2, \dots, \gamma_{|\mathcal{R}|}\right\}$ be a route in $\mathbb{R}$ with an ordered sequence of its stops γ_l. The *NR* algorithm calculates an average distance as $\bar{c}_{\mathcal{R}} = \left(\sum\limits_{l=1}^{|\mathcal{R}|-1} c_{\gamma_l \gamma_{l+1}}\right) / |\mathcal{R}|$ for each $\mathcal{R} \in \mathbb{R}$ and removes the customer $j^* = argmax_{\mathcal{R}} \in \mathbb{R} j \in \mathcal{R}\left\{\bar{c}_{\mathcal{R}} - \bar{c}_{\mathcal{R}j}\right\}$. In other words, the algorithm removes the customer that increases its corresponding average route cost the most. The procedure is repeated until n_i customers have been removed from S_i.

Node Neighborhood Removal (NNR)

This algorithm is a plain adaptation of the node neighborhood removal operator used by Demir et al. [14]. It selects a random customer $j \in C$ and then removes it along with the closest $n_i - 1$ customers in its vicinity.

A.2 Insertion Algorithms

The algorithms described below are adopted from Ropke and Pisinger [39], Pisinger and Ropke [31] and Demir et al. [14], and used in our proposed ALNS framework.

Greedy Insertion (GI)

This algorithm is a simple construction heuristic that inserts each removed customer into the best feasible position of a route in S_i^- . At each step, *GI* selects a removed customer $j^* \in \mathcal{L}$ that gives the lowest incremental insertion cost among all $j \in \mathcal{L}$ evaluated for insertion at all possible positions in all routes $\mathcal{R}_l \in \mathbb{R}$. For regular customers, GI evaluates the insertion cost for the customer only, whereas for premium customers, GI additionally inserts the required vendors V_{j^*} identified by one of the vendor selection algorithms. In the latter case, the algorithm randomly selects a member of V_{j^*} and evaluates inserting it into the best position among all routes. This random selection and insertion procedure is repeated until all vendors have been inserted, which is followed by the insertion of the premium customer. After a customer and its vendors, if any are inserted, the customer is removed from $\mathcal{L}$ and the process is repeated until $\mathcal{L} = \phi$. Both *clean* and *noise-imposed* versions of *GI* are used.

Zone Insertion (ZI)

This algorithm is a modified version of the zone insertion operator used by Demir et al. [14]. To determine the best insertion position for each customer, it prioritizes time window over distance, aiming to leave enough room for future insertions. Let $\mathbb{W}_j^{\mathcal{R}_l} = min_{l_k \in \mathcal{R}_l} \{max\{0, (T_{l_k} + s_{l_k} + t_{l_k j}) - e_j\}\}$ be the *waiting cost* of inserting a customer $j \in \mathcal{L}$ into route $\mathcal{R}_l$ at a feasible position that yields the least waiting time. The algorithm adaptively selects one of the zone directions listed in the *ZR* algorithm and works with the corresponding set of zones $\mathbb{Z} = \{Z_1, \ldots, Z_{\mathbb{Z}}\}$. At each iteration, *ZI* randomly selects a customer $j \in \mathcal{L}$ located in zone Z_k. Then, it identifies the routes $\mathcal{R} \in \mathbb{R}$ for which $(Z_k \cap \mathcal{R}) \neq \varnothing$ and calculates *zone insertion cost* using the least waiting cost position. If no feasible position exists, the customer is inserted by means of the GI algorithm. Both *clean* and *noised-imposed* versions of *ZI* are used.

Appendix B. Parameter Tuning

The results of the parameter tuning are summarized in Table B.1. The order of the parameters indicates the tuning sequence. All parameters were initially set to the values given in the "Initial" column as described in Ropke and Pisinger [39,40], Pisinger and Ropke [31], and Demir et al. [14]. First, we performed ten runs on nine tuning instances (R104, R112, R201, R204, R207, RC104, RC106, RC206, and RC207) by considering ten different values of σ_2 shown in the first row. Then, we calculated the average percentage deviation (denoted as

Table B.1
Parameter tuning summary results.

Parameter		Initial	Values Tested								
σ_2	Value	9	0	2	4	6	12	14	**16**	18	20
	%Dev	0.87	0.93	0.96	0.99	0.91	0.99	0.87	**0.73**	0.80	0.95
N_{SI}	Value	**100**	50	150	200	250	300	350	400	450	500
	%Dev	**0.73**	0.74	0.91	0.94	0.95	0.88	0.94	0.86	0.89	0.91
N_{Iwl}	Value	25000	100	500	1000	2000	3000	**4000**	5000	6000	10000
	%Dev	0.74	1.63	0.98	0.89	0.87	0.96	**0.73**	0.75	0.78	0.75
ρ	Value	**0.1**	0.05	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
	%Dev	**0.73**	0.78	0.92	0.79	0.88	0.78	0.88	0.92	0.81	0.82
σ_1	Value	33	5	10	**20**	25	30	35	40	45	50
	%Dev	0.73	0.85	0.90	**0.71**	0.88	0.89	1.01	0.86	0.96	0.79
σ_3	Value	**13**	3	6	9	12	15	21	24	27	30
	%Dev	**0.71**	0.81	0.86	0.85	0.83	0.81	0.77	0.88	0.81	0.93
ϕ_1	Value	**9**	0.5	1	3	5	7	11	13	15	
	%Dev	**0.71**	0.77	0.93	0.93	0.86	0.87	0.86	0.96	0.87	
ϕ_2	Value	**3**	0.25	1	5	7	9	11	13	15	
	%Dev	**0.71**	0.73	0.84	0.87	0.90	0.81	0.73	0.87	0.90	
ϕ_3	Value	**5**	0.15	1	3	7	9	11	13	15	
	%Dev	**0.71**	0.86	0.79	0.91	0.86	0.84	0.87	0.85	0.77	
ϕ_4	Value	**2**	0.25	1	3	4	5	6	7	8	9
	%Dev	**0.71**	0.85	0.82	0.77	0.76	0.88	0.88	0.84	0.81	0.80
μ	Value	**0.05**	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
	%Dev	**0.71**	0.92	0.87	0.97	0.98	1.01	1.11	1.12	1.10	1.30
ε	Value	**0.9998**	0.999	0.9991	0.9992	0.9993	0.9994	0.9995	0.9996	0.9997	0.9999
	%Dev	**0.71**	1.37	1.12	1.26	1.09	1.08	1.04	0.89	0.84	0.95
3	Value	**0.025**	0.05	0.075	0.1	0.125	0.15	0.175	0.2		
	%Dev	**0.71**	0.90	1.04	0.98	0.90	0.90	0.89	0.96		
κ	Value	**3**	1	2	4	5	6				
	%Dev	**0.71**	0.85	0.84	0.85	0.82	0.89				
η	Value	**6**	2	4	8	10	12				
	%Dev	**0.71**	0.75	0.82	0.75	0.82	0.83				
γ	Value	**0.2**	0	0.05	0.1	0.15	0.25				
	%Dev	**0.71**	0.92	0.81	0.88	0.87	0.75				
ξ	Value	**2**	0	1	3	4	5				
	%Dev	**0.71**	0.78	0.76	0.74	0.88	0.84				
$n_{\mathbb{Z}}$	Value	**11**	5	7	9	13	15	19	21	25	30
	%Dev	**0.71**	0.73	0.78	0.87	0.82	0.75	0.77	0.73	0.80	0.86
ψ	Value	0.25	0.05	0.1	0.15	0.2	0.3	**0.35**	0.4	0.45	0.5
	%Dev	0.10	0.12	0.08	0.12	0.08	0.08	**0.04**	0.08	0.08	0.09

“%Dev”) from the average of the corresponding best known solutions, determined 0.73% as the least deviation, and fixed the value of the parameter σ_2 to 16, which had achieved this best average performance. We repeated this procedure for the remaining parameters in the order given until all parameter values had been tuned. Note that most parameters (15 out of 19) have preserved their initial values tested.

Appendix C. Detailed Results for the VRPTW Instances

Table C.1 provides the results found by the proposed ALNS and compares them with those presented in Pisinger and Ropke [30] and Demir et al. [14], and with the optimal solutions reported in Roberti [37]. Table C.2 presents the new best-known solutions for the real-numbered R106, R107, R108, R210, and RC107 instances. Finally, Table C.3 reports the optimal solution for the truncated R208 problem.

Table C.1
Detailed results for the VRPTW instances.

Instance	Optimal	PR			DBL		The Proposed ALNS			Deviation from (%)			
	distance	best dist.	avg. dist.	avg. time (s)	best dist.	avg. time (s)	best dist.	avg. dist.	avg. time (s)	optimal	PR best	DBL best	PR avg.
R101	1637.7	1637.7	1638.6	30.0	1637.7	48.0	**1637.7**	1639.0	62.5	0.00	0.00	0.00	0.02
R102	1466.6	1467.7	1467.7	33.0	1466.6	46.0	**1466.6**	1467.4	60.6	0.00	−0.07	0.00	−0.02
R103	1208.7	1208.7	1208.9	34.0	1208.7	46.0	**1208.7**	1209.3	52.2	0.00	0.00	0.00	0.03
R104	971.5	976.0	977.1	34.0	971.5	45.0	976.0	977.8	45.4	0.46	0.00	0.46	0.07
R105	1355.3	1355.3	1355.8	31.0	1355.3	47.0	**1355.3**	1356.6	53.2	0.00	0.00	0.00	0.06
R106	1234.6	1234.6	1234.6	33.0	1234.6	46.0	**1234.6**	1235.2	50.3	0.00	0.00	0.00	0.05
R107	1064.6	1064.6	1068.2	33.0	1064.6	42.0	**1064.6**	1068.0	47.9	0.00	0.00	0.00	−0.02
R108	932.1	933.7	943.5	36.0	936.1	44.0	**932.1**	939.4	45.6	0.00	−0.17	−0.43	−0.43
R109	1146.9	1146.9	1150.2	31.0	1146.9	46.0	**1146.9**	1148.4	50.0	0.00	0.00	0.00	−0.16
R110	1068.0	1075.6	1083.1	33.0	1073.9	49.0	**1068.0**	1079.8	50.4	0.00	−0.71	−0.55	−0.30
R111	1048.7	1048.7	1049.2	33.0	1049.9	42.0	**1048.7**	1050.1	48.5	0.00	0.00	−0.11	0.09
R112	948.6	948.6	952.2	33.0	948.6	40.0	950.4	954.2	46.7	0.19	0.19	0.19	0.21
C101	827.3	827.3	827.3	29.0	827.3	41.0	**827.3**	827.3	40.4	0.00	0.00	0.00	0.00
C102	827.3	827.3	827.3	32.0	827.3	40.0	**827.3**	827.3	43.1	0.00	0.00	0.00	0.00
C103	826.3	826.3	826.3	34.0	826.3	41.0	**826.3**	826.3	44.9	0.00	0.00	0.00	0.00
C104	822.9	822.9	822.9	36.0	822.9	40.0	**822.9**	822.9	45.9	0.00	0.00	0.00	0.00
C105	827.3	827.3	827.3	30.0	827.3	39.0	**827.3**	827.3	41.3	0.00	0.00	0.00	0.00
C106	827.3	827.3	827.3	31.0	827.3	41.0	**827.3**	827.3	42.5	0.00	0.00	0.00	0.00
C107	827.3	827.3	827.3	31.0	827.3	41.0	**827.3**	827.3	42.6	0.00	0.00	0.00	0.00
C108	827.3	827.3	827.3	32.0	827.3	42.0	**827.3**	827.3	43.5	0.00	0.00	0.00	0.00
C109	827.3	827.3	827.3	34.0	827.3	40.0	**827.3**	827.3	46.0	0.00	0.00	0.00	0.00
RC101	1619.8	1619.8	1629.8	28.0	1619.8	44.0	**1619.8**	1628.8	56.1	0.00	0.00	0.00	−0.06
RC102	1457.4	1463.5	1475.1	30.0	1463.5	42.0	**1457.4**	1470.4	54.4	0.00	−0.42	−0.42	−0.32
RC103	1258.0	1267.0	1272.2	31.0	1267.1	43.0	**1258.0**	1266.9	50.2	0.00	−0.71	−0.72	−0.42
RC104	1132.3	1132.6	1132.8	33.0	1133.1	42.0	1132.6	1134.9	48.3	0.03	0.00	−0.04	0.19
RC105	1513.7	1513.7	1514.2	30.0	1513.7	43.0	**1513.7**	1514.1	56.0	0.00	0.00	0.00	−0.01
RC106	1372.7	1373.9	1376.1	29.0	1372.7	41.0	1373.9	1378.4	50.1	0.09	0.00	0.09	0.17
RC107	1207.8	1209.3	1213.0	30.0	1209.3	40.0	**1207.8**	1212.1	49.1	0.00	−0.12	-0.12	−0.07
RC108	1114.2	1114.2	1124.6	31.0	1114.2	43.0	**1114.2**	1118.2	47.0	0.00	0.00	0.00	−0.57
R201	1143.2	1148.5	1153.9	45.0	1143.2	71.0	1148.1	1152.1	39.6	0.43	−0.03	0.43	−0.16
R202	1029.6	1036.9	1041.0	54.0	1032.2	72.0	1033.8	1038.8	41.9	0.41	−0.30	0.16	−0.21
R203	870.8	872.4	876.5	60.0	873.3	76.0	871.3	874.1	43.7	0.06	−0.13	−0.23	−0.27
R204	731.3	731.3	731.5	67.0	731.3	75.0	731.8	733.5	46.2	0.07	0.07	0.07	0.27
R205	949.8	949.8	952.4	58.0	950.4	71.0	**949.8**	952.8	40.4	0.00	0.00	−0.06	0.04
R206	875.9	880.6	880.6	61.0	881.0	76.0	880.6	882.3	43.0	0.54	0.00	−0.05	0.19
R207	794.0	794.0	796.4	72.0	794.0	85.0	**794.0**	796.0	45.1	0.00	0.00	0.00	−0.05
R208*	701.2	701.2	703.1	86.0	702.9	88.0	**701.0**	705.2	48.6	**−0.03**	−0.03	−0.27	0.30
R209	854.8	855.8	860.2	60.0	854.8	74.0	856.0	861.4	41.6	0.14	0.02	0.14	0.14
R210	900.5	908.4	914.0	59.0	906.3	70.0	908.4	914.7	43.5	0.88	0.00	0.23	0.08
R211	746.7	752.3	758.3	67.0	751.6	74.0	751.7	755.6	45.1	0.67	−0.08	0.01	−0.36
C201	589.1	589.1	589.1	69.0	589.1	82.0	**589.1**	589.1	37.9	0.00	0.00	0.00	0.00
C202	589.1	589.1	589.1	74.0	589.1	85.0	**589.1**	589.1	41.6	0.00	0.00	0.00	0.00
C203	588.7	588.7	588.7	80.0	588.7	92.0	**588.7**	588.7	41.5	0.00	0.00	0.00	0.00
C204	588.1	588.1	588.1	84.0	588.1	91.0	**588.1**	588.1	45.6	0.00	0.00	0.00	0.00
C205	586.4	586.4	586.4	76.0	586.4	86.0	**586.4**	586.4	38.5	0.00	0.00	0.00	0.00
C206	586.0	586.0	586.0	72.0	586.0	81.0	**586.0**	586.0	39.1	0.00	0.00	0.00	0.00
C207	585.8	585.8	585.8	74.0	585.8	86.0	**585.8**	585.8	40.8	0.00	0.00	0.00	0.00
C208	585.8	585.8	585.8	74.0	585.8	88.0	**585.8**	585.8	39.4	0.00	0.00	0.00	0.00
RC201	1261.8	1262.6	1272.3	42.0	1262.7	74.0	1264.4	1271.1	42.1	0.21	0.14	0.13	−0.09
RC202	1092.3	1095.8	1097.4	46.0	1095.8	71.0	1095.6	1098.2	44.5	0.30	−0.02	−0.02	0.07
RC203	923.7	923.7	937.6	56.0	923.7	73.0	926.0	934.5	43.5	0.25	0.25	0.25	−0.33
RC204	783.5	785.8	788.1	68.0	783.8	76.0	**783.5**	785.3	45.2	0.00	−0.29	−0.04	−0.36
RC205	1154.0	1154.0	1154.0	45.0	1154.0	64.0	**1154.0**	1156.1	43.2	0.00	0.00	0.00	0.18
RC206	1051.1	1051.1	1062.5	52.0	1051.1	68.0	1053.3	1065.0	41.7	0.21	0.21	0.21	0.24
RC207	962.9	966.6	976.2	55.0	966.6	64.0	963.3	974.4	43.6	0.04	−0.34	−0.34	−0.18
RC208	776.1	777.3	790.5	65.0	777.3	72.0	777.9	787.3	44.7	0.23	0.08	0.08	−0.40
Average	**973.2**	**974.6**	**977.7**	**47.3**	**974.3**	**59.4**	**974.1**	**977.3**	**45.8**	**0.09**	**−0.04**	**−0.02**	**−0.04**

Table C.2
New best-known solutions for the real-numbered VRPTW instances.

R106		**Total distance = 1239.37**
1	94 92 42 15 57 87 97 95 13	61.64
2	12 29 78 79 68 54 24 80	86.18
3	69 30 51 81 9 35 34 3 77	106.17
4	73 41 22 75 56 74 2 58	127.24
5	48 47 36 19 49 46 82 7 52	127.23
6	27 62 88 18 89	75.88
7	21 72 39 23 67 55 4 25 26	79.70
8	63 64 11 90 10 31	126.94
9	59 37 14 44 38 86 43 100 98 93	129.21
10	28 76 40 53	46.17
11	96 85 91 16 61 99 6	62.65
12	83 45 8 84 17 5 60	106.13
13	50 33 65 71 66 20 32 70 1	104.24
R107		**Total distance = 1072.12**
1	60 83 45 46 8 84 5 17 61 85 93	113.47
2	94 96 92 59 99 6 87 13	62.75
3	27 69 30 88 31 10 70 1	86.16
4	33 81 65 71 9 35 34 3 77	126.47
5	52 7 62 11 63 90 32 66 20 51 50	114.94
6	2 57 43 15 41 22 75 56 74 72 73 21	103.08
7	95 97 42 14 44 38 86 16 91 100 37 98	105.74
8	48 47 36 64 49 19 82 18 89	126.00
9	53 40 58	24.36
10	26 39 23 67 55 4 25 54	127.04
11	28 76 79 78 29 24 68 80 12	82.10
R108		**Total distance = 938.20**
1	2 57 15 43 42 87 97 95 94 13 58	124.44
2	73 22 41 23 67 39 56 75 74 72 21 40	107.11
3	6 96 59 99 93 5 84 17 45 83 60 89	107.75
4	52 88 62 19 11 64 63 90 32 10 31	90.26
5	26 12 80 68 29 24 55 4 25 54	78.99
6	27 69 50 76 3 79 78 34 81 33 77 28	115.17
7	1 70 30 51 9 35 71 65 66 20	84.67
8	92 98 91 44 14 38 86 16 61 85 100 37	114.80
9	18 7 82 8 46 36 49 47 48	106.08
10	53	8.94
RC107		**Total distance = 1211.11**
1	72 71 93 94 67 50 62 91 80	105.63
2	41 38 39 42 44 43 40 37 35 36	108.66
3	11 12 14 47 17 16 15 13 9 10	100.98
4	92 95 84 85 63 51 76 89 56	123.47
5	69 98 88 53 78 73 79 60 55 70 68	116.69
6	2 6 7 8 5 3 1 45 46 4 100	101.80
7	31 29 27 28 26 30 32 34 33	140.98
8	82 99 52 87 59 86 57 66	96.79
9	65 83 25 77 75 97 58 74	150.63
10	61 81 54 96	51.72
11	64 22 19 23 21 18 48 49 20 24	105.27
12	90	8.49
R210		**Total distance = 909.96**
1	95 92 59 5 83 45 36 47 48 82 18 7 88 62 19 11 63 64 49 46 8 84 17 85 98 37 100 91 93 60 89	274.50
2	21 73 72 23 67 39 56 75 22 41 74 4 55 25 54 26	145.37
3	27 69 1 30 51 33 71 65 66 20 32 90 10 70 31 52	152.31
4	28 12 76 3 79 29 78 81 9 35 34 24 80 68 77 50	144.95
5	2 57 15 42 14 44 38 86 16 61 99 96 6 94 87 43 97 13	168.47
6	53 40 58	24.36

Appendix D. Supplementary material

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.cor.2015.11.008.
Appendix A. Supplementary material

Table C.3
Optimal solution for R208.

R208		**Total distance = 701.0**
1	52 18 82 48 7 88 31 70 30 32 90 63 10 62 19 11 64 49 36 47 46 8 45 17 84 83 60 5 99 96 97 87 37 98 91 100 42 57 2 13	290.6
2	27 69 1 50 76 33 81 9 51 20 66 65 71 35 34 78 79 3 77 68 80 29 24 54 12 26 28	192.4
3	89 6 94 95 92 59 93 85 61 16 86 44 38 14 43 15 41 22 75 56 23 67 39 25 55 4 72 74 73 21 40 58	209.2
4	53	8.8

References

[1] Adulyasak Y, Cordeau J-F, Jans R. Optimization-based Adaptive Large Neighborhood Search for the production routing problem. Transp Sci 2014;48 (1):20–45.
[2] Aksen D, Kaya O, Salman S, Tüncel Ö. An Adaptive Large Neighborhood Search algorithm for a selective and periodic inventory routing problem. Eur J Oper Res 2014;239:413–26.
[3] Angelelli E, Speranza MG. The periodic vehicle routing problem with intermediate facilities. Eur J Oper Res 2002;137:233–47.
[4] Attanasio A, Cordeau J-F, Ghiani G, Laporte G. Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. Parallel Comput 2004;30:377–87.
[5] Azi N, Gendreau M, Potvin J-Y. An Adaptive Large Neighborhood Search for a vehicle routing problem with multiple routes. Comput Oper Res 2014;41:167–73.
[6] Bard JF, Huang L, Dror M, Jaillet P. A branch and cut algorithm for the VRP with satellite facilities. IIE Trans 1998;30:821–34.
[7] Bard JF, Huang L, Jaillet P, Dror M. A decomposition approach to the inventory routing problem with satellite facilities. Transp Sci 1998;32:189–203.
[8] Bent R, Hentenryck PV. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. Comput Oper Res 2006;33: 875–893.
[9] Bensinger G, Stevens, L (2014) Amazon, in threat to UPS, tries its own deliveries. The Wall Street Journal. (online.wsj.com/news/articles/SB10001424052702304788404579521522792859890, last accessed on June 10, 2014).
[10] Bozkaya B, De Kervenoael R, Aykac DSO. Premium e-grocery: exploring value in logistics integrated service solutions. Proceedings of the International Conference on Prospects for Research in Transportation and Logistics on a Regional–Global Perspective, 3. Istanbul, Turkey: (Doğuş University Publications; 2009. p. 173–8.
[11] Campbell AM, Savelsbergh MWP. Decision support for consumer direct grocery initiatives. Transp Sci 2005;39:313–27.
[12] Caricato P, Ghiani G, Grieco A, Guerriero E. Parallel tabu search for a pickup and delivery problem under track contention. Parallel Comput 2003;29:631–9.
[13] Cordeau J, Laporte G, Pasin F, Ropke S. Scheduling technicians and tasks in a telecommunications company. J Sched 2010;13:1–17.
[14] Demir E, Bektas T, Laporte G. An Adaptive Large Neighborhood Search heuristic for the pollution-routing problem. Eur J Oper Res 2012;223:346–59.
[15] Doerner K, Hartl RF, Reimann M. Ants solve time constrained pickup and delivery problems with full truckloads. Oper Res Proc 2001:395–400.
[16] Ganesh K, Narendran TT. CLOVES: A cluster-and-search heuristic to solve the vehicle routing problem with delivery and pick-up. Eur J Oper Res 2007;178:699–717.
[17] Hart SM. The modeling and solution of a class of dial-a-ride problems using simulated annealing. Control Cybern 1996;25:131–57.
[18] Hemmelmayr VC, Cordeau JF, Crainic TG. An Adaptive Large Neighborhood Search heuristic for two-echelon vehicle routing problems arising in city logistics. Comput Oper Res 2012;39:3185–99.
[19] Kristiansen S, Sørensen M, Herold MB, Stidsen TR. The consultation timetabling problem at Danish high schools. J Heuristics 2013;19:465–95.
[20] Li H, Lim A (2001) A metaheuristic for the pickup and delivery problem with time windows. In: Proceedings of the ICTAI-2001, the 13th IEEE conference on tools with artificial intelligence. (Dallas).
[21] Liu J, Liu D, Liu M, He Y (2010) An improved multiple ant colony system for the collection vehicle routing problems with intermediate facilities. In: Proceedings of the intelligent control and automation (WCICA), 2010 8th world congress. (Jinan, China).
[22] Masson R, Lehuede F, Peton O. An Adaptive Large Neighborhood Search for the pickup and delivery problem with transfers. Transp Sci 2013;47(3):344–55.
[23] Melachrinoudis E, Ilhan AB, Min H. A dial-a-ride problem for client transportation in a healthcare organization. Comput Oper Res 2007;34(3):742–59.
[24] Montane FAT, Galvao RD. A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. Comput Oper Res 2006;33(3):595–619.

[25] Muller LF (2009) An Adaptive Large Neighborhood Search algorithm for the resource-constrained project scheduling problem. In: Proceedings of the VIII metaheuristics international conference (MIC). 2009 (Hamburg, Germany).

[26] Muller LF, Spoorendonk S, Pisinger D. A hybrid Adaptive Large Neighborhood Search heuristic for lot-sizing with setup times. Eur J Oper Res 2012;218 (3):614–23.

[27] Nanry WP, Barnes JW. Solving the pickup and delivery problem with time windows using reactive tabu search. Transp Res Part B 2000;34:107–21.

[28] Parragh S, Doerner KF, Hartl RF. A survey on pickup and delivery problems, Part I: transportation between customers and depot. J Für Betriebswirtschaft 2008;58(1):21–51.

[29] Parragh S, Doerner KF, Hartl RF. A survey on pickup and delivery problems, Part II: transportation between pickup and delivery locations. J Für Betriebswirtschaft 2008;58(2):81–117.

[30] Pisinger D, Ropke S. A general heuristic for vehicle routing problems. Technical Report. Copenhagen: DIKU - Department of Computer Science, University of Copenhagen; 2005 17.06.2015.

[31] Pisinger D, Ropke S. A general heuristic for vehicle routing problems. Comput Oper Res 2007;34(8):2403–35.

[32] Pisinger D, Ropke S. Large neighborhood search. In: Gendreau M, Potvin JY, editors. Handbook of Metaheuristics; 2010. p. 399–419.

[33] Polacek M, Doerner K, Hartl R, Maniezzo V. A variable neighborhood search for the capacitated arc routing problem with intermediate facilities. J Heuristics 2008;14(5):405–23.

[34] Qu Y, Bard JF. A GRASP with Adaptive Large Neighborhood Search for pickup and delivery problems with transshipment. Comput Oper Res 2012;39 (10):2439–56.

[35] Rekiek B, Delchambre A, Saleh HA. Handicapped person transportation, an application of the grouping genetic algorithm. Eng Appl Artif Intell 2006;19:511–20.

[36] Ribeiro GM, Laporte G. An Adaptive Large Neighborhood Search heuristic for the cumulative capacitated vehicle routing problem. Comput Oper Res 2012;39(3):728–35.

[37] Roberti R. Exact algorithms for different classes of vehicle routing problems. Italy: University of Bologna; 2012 PhD Thesis.

[38] Ropke S (2014) Personal communication.

[39] Ropke S, Pisinger D. An Adaptive Large Neighborhood Search heuristic for the pickup and delivery problem with time windows. Transp Sci 2006;40(4):455–72.

[40] Ropke S, Pisinger D. A unified heuristic for a large class of vehicle routing problems with backhauls. Eur J Oper Res 2006;171(3):750–75.

[41] Sevilla FC, Blas CS (2003) Vehicle routing problem with time windows and intermediate facilities. In: S.E.I.O.'03 Edicions de la Universitat de Lleida. pp. 3088–3096.

[42] Shaw P. Using constraint programming and local search methods to solve vehicle routing problems. Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming. New York: Springer; 1998. p. 417–31.

[43] Solomon MM. Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper Res 1987;35:254–65.

[44] Tarantilis CD, Zachariadis EE, Kiranoudis CT. A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. INFORMS J Comput 2008;20(1):154–68.

[45] Toth P, Vigo D. Heuristic algorithms for the handicapped persons transportation problem. Transp Sci 1997;31:60–71.

[46] Toth P, Vigo D. An overview of vehicle routing problems. Veh Routing Probl 2002;9:1–26.

[47] Yanik S, Bozkaya B, deKervenoael R. A new VRPPD model and a hybrid heuristic solution approach for e-tailing. Eur J Oper Res 2013. http://dx.doi.org/10.1016/j.ejor.2013.05.023.

[48] Yildirim UM, Çatay B. A time-based pheromone approach for the ant system. Optim Lett 2012;6:1081–99.

Web Reference

[1] fresh.amazon.com, web reference, last accessed on June 10, 2014.

[2] www.peapod.com, web reference, last accessed on June 10, 2014.

[3] www.instacart.com, web reference, last accessed on June 10, 2014.

[4] http://people.sabanciuniv.edu/catay/EDRP_Instances.zip, EDRP benchmark instances, last accessed on June 10, 2014.