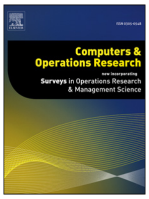
Computers and Operations Research 115 (2020) 104858

Contents lists available at ScienceDirect

Computers and Operations Research

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



The vehicle routing problem with simultaneous pickup and delivery and handling costs

Richard P. Hornstra[a], Allyson Silva[b], Kees Jan Roodbergen[a,*], Leandro C. Coelho[a,b,c]

[a] *Faculty of Economics and Business, Department of Operations, University of Groningen, the Netherlands*
[b] *CIRRELT and Faculté des Sciences de l'Administration, Université Laval, Canada*
[c] *Canada Research Chair in Integrated Logistics, Université Laval, Canada*

ARTICLE INFO

Article history:
Received 15 November 2018
Revised 20 November 2019
Accepted 20 November 2019
Available online 21 November 2019

Keywords:
Vehicle routing problem
Pickup and delivery
Handling policies
Hybrid heuristic

ABSTRACT

In this paper we introduce the vehicle routing problem with simultaneous pickup and delivery and handling costs (VRPSPD-H). In the VRPSPD-H, a fleet of vehicles operates from a single depot to service all customers, which have both a delivery and a pickup demand such that all delivery items originate from and all pickup items go to the depot. The items on the vehicles are organized as a single linear stack where only the last loaded item is accessible. Handling operations are required if the delivery items are not the last loaded ones. We implement a heuristic handling policy approximating the optimal decisions for the handling sub-problem, and we propose two bounds on the optimal policy, resulting in two new myopic policies. We show that one of the myopic policies outperforms the other one in all configurations, and that it is competitive with the heuristic handling policy if many routes are required. We propose an adaptive large neighborhood search (ALNS) metaheuristic to solve our problem, in which we embed the handling policies. Computational results indicate that our metaheuristic finds optimal solutions on instances of up to 15 customers. We also compare our ALNS metaheuristic against best solutions on benchmark instances of two special cases, the vehicle routing problem with simultaneous pickup and delivery (VRPSPD) and the traveling salesman problem with pickups, deliveries and handling costs (TSPPD-H), and on two related problems, the vehicle routing problem with divisible pickup and delivery (VRPDPD) and the vehicle routing problem with mixed pickup and delivery (VRPMPD). We find or improve 39 out of 54 best known solutions (BKS) for the VRPSPD, 36 out of 54 BKS for the VRPDPD, 15 out of 21 BKS for the VRPMPD, and 69 out of 80 BKS for the TSPPD-H. Finally, we introduce and analyze solutions for the variations of the VRPDPD and VRPMPD with handling costs – the VRPDPD-H and the VRPMPD-H, respectively.

1. Introduction

When consumers buy major appliances, it is common practice that the store home-delivers the newly bought products and takes back the old machinery. These appliances are not easily moved around in the delivery vehicle and, if pickup items are placed in front of delivery items, they may cause obstruction issues at subsequent stops. Handling these pickup items to access delivery items is then a time consuming task and should not be ignored when designing the set of routes to service customers, which should show a clear trade-off between routing and handling costs. Similar situations arise in the delivery and collection of large furniture, which often cover the entire vehicle width, the relocation of damaged and undamaged bicycles in public sharing systems (Battarra et al. (2010)) or when delivering calves to farms and collecting mature cows (Erdoğan et al. (2012)). Recently, other studies looked at the effect of obstruction issues in related problem settings (e.g., Veenstra et al. (2017)).

We refer to our vehicle routing problem (VRP) as the VRP with simultaneous pickup and delivery and handling costs (VRPSPD-H), in which a fleet of homogeneous vehicles operates from a single depot to service all customers, which may have both a delivery and a pickup demand. These demands are such that all delivery items originate from and all pickup items go to the depot. The items on the vehicles are organized as a single linear stack which obeys the last-in-first-out (LIFO) policy and is only accessible from the rear. This means that only the most recently loaded item is accessible, and if this is not the item of interest (for instance a pickup item

* Corresponding author.
E-mail addresses: r.p.hornstra@gmail.com (R.P. Hornstra), allyson.silva@cirrelt.ca (A. Silva), k.j.roodbergen@rug.nl (K.J. Roodbergen), leandro.coelho@cirrelt.ca (L.C. Coelho).

https://doi.org/10.1016/j.cor.2019.104858
0305-0548/Crown Copyright © 2019 Published by Elsevier Ltd. All rights reserved.

when a delivery is to be made), handling operations are required before the desired service can be made.

The VRPSPD-H generalizes some problems in the class of routing problems with pickups and deliveries, or linehauls and backhauls, such as the VRP with simultaneous pickup and delivery (VRPSPD) by extending it to consider handling operations, and the traveling salesman problem with pickups, deliveries and handling costs (TSPPD-H) by allowing for the construction of multiple routes. Other problems are also related, such as the VRP with mixed pickup and delivery (VRPMPD), which considers that customers either have pickup or delivery items, and the VRP with divisible pickup and delivery (VRPDPD), which considers that customers can be visited twice, once for pickup and once for delivery. These problems find many practical applications and for an extensive list of studies considering them, we refer the reader to literature reviews on routing problems with pickups and deliveries such as Berbeglia et al. (2007); Koç and Laporte (2018); Nagy and Salhi (2005) and Wassan and Nagy (2014); for an overview of real-life routing problems we refer to Coelho et al. (2016).

In this paper, we introduce, model and solve the VRPSPD-H. We propose a mathematical formulation to solve small problem instances optimally for the VRPSPD-H. We show that via a simple reformulation we can solve the generalization of the VRPMPD with handling costs (VRPMPD-H). Our model can find an upper bound solution to the VRPDPD with handling costs (VRPDPD-H). These two problems are also introduced here. An adaptive large neighborhood search (ALNS) metaheuristic in which we embed handling policies is also proposed as an alternative to solve larger instances. The handling decisions are made using a heuristic handling policy which approximates the optimal decisions to the handling sub-problem. Its performance is compared with two new myopic policies obtained by deriving bounds on the optimal handling policy. The quality of the proposed ALNS is shown by benchmarking on many problems and instances from the literature (VRPSPD, VRPDPD, VRPMPD, and TSPPD-H), and by comparing it with optimal results obtained from our mathematical formulation for the newly introduced problems (VRPSPD-H, VRPDPD-H and VRPMPD-H).

A closely related line of research is Veenstra et al. (2017), in which a single vehicle fulfills a set of requests. In contrast to our problem, a request is defined as the transportation of items from a specific pickup location to a specific delivery location. These may both be different from the depot, whereas in our problem the depot is always the origin and destination of the deliveries and pickups, respectively. The operating vehicle also contains a single linear stack subject to the LIFO policy and handling operations are considered as well.

In Battarra et al. (2010), a special case of our VRPSPD-H, employing only a single vehicle, is introduced and the authors propose branch-and-cut algorithms to solve the problem. Due to the complexity of the handling sub-problem, the authors introduce three handling policies and solve instances up to 25 customers optimally. The authors show that their Policy 3 (mixed policy) significantly outperforms the Policies 1 (rear policy) and 2 (front policy). The three policies are described in Section 2.1. A follow-up study by Erdoğan et al. (2012) focuses on the mixed policy of Battarra et al. (2010). The authors design an exact dynamic program (DP) with quadratic complexity and a linear-time heuristic, both to the mixed policy, to solve the handling sub-problem. These methods are integrated into three different metaheuristics (tabu search, iterated local search, and iterated tabu search) which are used to solve instances of up to 200 customers. We extend the works of Battarra et al. (2010) and Erdoğan et al. (2012) by adopting their approaches, i.e., the DP policy and the heuristic policy, to the handling sub-problem, in addition to our myopic policies, and integrate it with our metaheuristic for the VRPSPD-H to apply it on the multi-vehicle extension.

Many different heuristic methods have been proposed to solve the VRPSPD, the VRPMPD and the VRPDPD, including adaptive local search (Avci and Topaloglu (2015)), ant colony systems (Gajpal and Abad (2009); Kalayci and Kaya (2016)), variable neighborhood search (Polat (2017)) and tabu search (Nagy et al. (2013); Zachariadis and Kiranoudis (2011)). Despite the successes of these techniques, ALNS is growing in popularity over the last years. It extends the large neighborhood search (LNS) as first introduced by Shaw (1998) by an adaptive mechanism and has recently been implemented successfully in many different routing problems (Emeç et al. (2016); Grangier et al. (2016); Li et al. (2016)). We build upon these recent successes and design an ALNS metaheuristic for our problem.

Additional to heuristic solution methods, the VRPSPD-H, the VRPMPD-H, and the VRPDPD-H have also been solved using exact methods. Since the VRPSPD generalizes the standard capacitated VRP, a well-known NP-hard problem, it can be shown to be NP-hard as well. However, small instances have been solved using exact solution methods. Dell'Amico et al. (2006) use a branch-and-price method to solve instances of up to 40 customers optimally and Subramanian et al. (2013) propose a branch-cut-and-price method solving instances of up to 100 customers. Since our problem generalizes the VRPSPD, which we formally show in Section 3.1, our problem is NP-hard as well. We adapt the model of Dell'Amico et al. (2006) to fit handling costs and use it to solve small instances optimally. With a small reformulation, we show that the model can be used to solve the mixed version of the problem with handling costs (VRPMPD-H) and to find an upper bound to the divisible version of the problem (VRPDPD-H).

Other areas of research which are less related are the multi-vehicle pickup and delivery problem with LIFO constraints studied in Benavent et al. (2015) and the variant with time windows in Cherkesly et al. (2015). In contrast to our problem, these LIFO constraints prohibit delivery of an item not on top of the linear stack, leading to a setting without handling operations. Wang and Chen (2012) study the VRPSPD with time windows and an extension with multiple depots is studied in Nagy and Salhi (2005). Regarding handling operations, loading constraints in VRPs are discussed in Pollaris et al. (2015), and a VRP where products are loaded in multiple compartments is presented in Hübner and Ostermeier (2018). A problem with handling costs in the form of additional service time for a VRP with pickup and delivery and time windows is studied in Reimann and Ulrich (2006). Shuffling goods on-board may also be possible, without the need to unload, such as in the VRP with restricted mixing of deliveries and pickups as proposed by Casco et al. (1988).

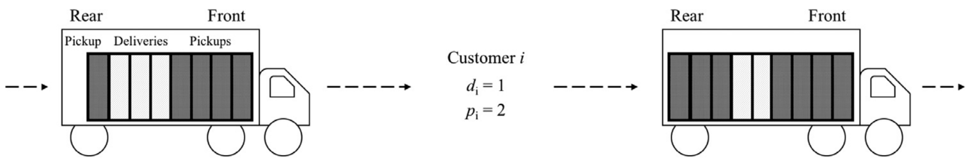
The remainder of this paper is structured as follows. In Section 2 we present a formal problem definition. Section 3 shows the aforementioned generalizations and related problems. Section 4 explains the ALNS metaheuristic proposed to solve the problem and the handling policies used to calculate handling costs. We report the results of an extensive numerical study in Section 6. Finally, Section 7 concludes the paper.

2. Problem definition

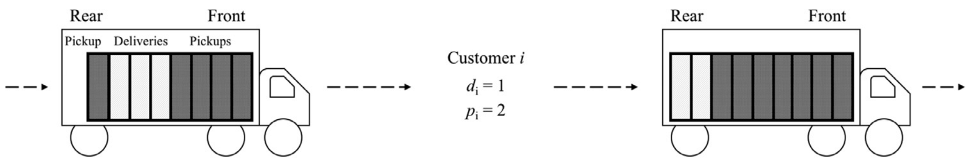
This section presents the model for the VRPSPD-H and is structured as follows. We formally define the model, explain handling policies and propose a MIP formulation in Section 2.1. Also, we present valid inequalities in Section 2.2.

2.1. Mathematical formulation

The VRPSPD-H is defined on a complete directed graph $G = (V, A)$, with $V = \{0, 1, \ldots, N\}$ being the set of vertices and A is the



(a) Placement of pickup items at the rear of the vehicle at cost h_p.



(b) Placement of pickup items at the front of the vehicle at cost $h_p + 2h_d$.

Fig. 1. Illustration of handling options at a customer. In the example, the customer requires one delivery item (light grey box) and supplies two pickup items (dark grey box).

set of arcs (i, j) between every pair of nodes $i, j \in V$, $i \neq j$. Let vertex 0 represent the depot, then $V_c = V \backslash \{0\}$ is the set of customer vertices. Define $A_r = \{(i, 0) : i \in V_c\}$ as the set of arcs which end at the depot. A positive travel cost c_{ij} satisfying the triangle inequality exists to each arc $(i, j) \in A$. Customer $i \in V_c$ requires $d_i > 0$ delivery items and supplies $p_i \geq 0$ pickup items. The delivery items originate from the depot and the pickup items destinate to the depot. A homogeneous fleet of vehicles with capacity Q is available at the depot.

We adopt the definition of an *additional operation* from Battarra et al. (2010), which is defined as the unloading and reloading of one item from a vehicle, with corresponding costs h_d and h_p for a delivery item and a pickup item, respectively. Battarra et al. (2010) defined three handling policies based on the following structure. The load in the vehicles is divided into three blocks: (i) the pickup items at the front of the vehicle which never require additional operations at remaining stops, (ii) the delivery items in the middle of the vehicle, and (iii) the pickup items at the rear of the vehicle which obstruct the delivery items.

Under the rear policy, all pickups are always placed at the rear of the vehicle, obstructing future deliveries. Under the front policy, all pickups are always placed at the front of the vehicle where they do not require additional operations at future stops. However, additional operations for the delivery items in the vehicle are required. The mixed policy is defined as follows. At each customer, the decision of placing the pickup items either at the rear or at the front of the vehicle is made. This decision is based on the volumes in each of the three blocks in the vehicle and the delivery and pickup demand of the customers positioned later in the route. Battarra et al. (2010) presented three mathematical models for the TSPPD-H, each considering one of the policies. Due to its generality and its better performance when compared to the rear and front policies, we consider in this work that a mixed policy is used.

Fig. 1 depicts the placement of pickups at the rear and front of the vehicle graphically. In the example, the customer demands a single delivery item and supplies two pickup items. To make the delivery, the obstructing pickup item needs to be unloaded in both cases. If the choice is to place the new pickup items at the rear of the vehicle (Fig. 1(a)) the two new pickup items are added to the already unloaded pickup item and are placed such that they obstruct the next delivery resulting in a total handling cost of h_p. Alternatively (Fig. 1(b)), the two remaining delivery items are unloaded prior to placing the pickup items at the front of the vehicle. This requires additional operations on the two delivery items at the current stop, but results in no obstruction for the next delivery, resulting in a total handling cost of $h_p + 2h_d$.

In a flow based formulation, let x_{ij} be a binary variable which is equal to one if arc $(i, j) \in A$ is part of the solution, and zero otherwise. Furthermore, y_{ij} represents the number of delivery items on board on arc $(i, j) \in A$, and w_{ij} and z_{ij} represent the number of pickup items on board at the front and rear of the vehicle on arc $(i, j) \in A$, respectively, such that $w_{ij} + z_{ij}$ represents the total number of pickup items on board of the vehicle on arc $(i, j) \in A$. Unless required, the number of items is not necessarily integer. Finally, we introduce the binary variable s_i, $i \in V_c$, which is equal to one if the pickup items are placed at the front of the vehicle, and zero if at the rear. Inspired by the models for the TSPPD-H by Battarra et al. (2010) and the VRPSPD by Dell'Amico et al. (2006), we propose the following formulation for the VRPSPD-H:

$$\text{minimize} \quad \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{(i,j)\in A\backslash A_r} h_p z_{ij} + \sum_{(i,j)\in A\backslash A_r} s_j h_d \left(y_{ij} - \frac{d_j}{|V_c|} \right) \tag{1}$$

subject to

$$\sum_{j\in V} x_{ij} = 1, i \in V_c, \tag{2}$$

$$\sum_{j\in V} x_{ij} = \sum_{j\in V} x_{ji}, i \in V, \tag{3}$$

$$\sum_{j\in V} y_{ji} - \sum_{j\in V} y_{ij} = d_i, i \in V_c, \tag{4}$$

$$\sum_{j\in V} \left(w_{ij} + z_{ij}\right) - \sum_{j\in V} \left(w_{ji} + z_{ji}\right) = p_i, i \in V_c, \tag{5}$$

$$w_{ij} + y_{ij} + z_{ij} \leq Q x_{ij}, (i, j) \in A, \tag{6}$$

$$\sum_{j\in V} z_{ij} = (1 - s_i)\left(\sum_{j\in V} z_{ji} + p_i\right) i \in V_c, \tag{7}$$

$$x_{ij} \in \{0, 1\}, (i, j) \in A, \tag{8}$$

$$s_i \in \{0, 1\}, i \in V_c, \tag{9}$$

$$w_{ij}, y_{ij}, z_{ij} \geq 0, (i, j) \in A. \tag{10}$$

Here, (1) states the objective function. The first term in the objective represents the routing cost, the second term corresponds to the handling costs for the pickup items at the rear of the vehicle and the third term corresponds to the handling costs for the delivery items when all pickup items are placed at the front of the vehicle. Due to the domain of the sum in the third term, we divide the quantity d_j by the number of arcs defined to go to customer j, $|V_c|$ since we are not considering arcs going from a customer to itself, to make sure we subtract it exactly once from the quantity y_{ij}. Constraints (2) force every customer to be visited exactly once, and constraints (3)–(5) induce flow conservation. Additionally, constraints (4)–(5) prevent subtours, and constraints (6) ensure that vehicle capacity is not violated. Constraints (7) update the location of the pickup items according to the decision of where to place them. Finally, constraints (8)–(10) define the nature of the variables. We also set the number of pickup items in the vehicles when leaving the depot and the number of delivery items in the vehicles going to the depot equal to zero, i.e., $\sum_{i\in V_c} w_{0i} = 0$, $\sum_{i\in V_c} z_{0i} = 0$, and $\sum_{i\in V_c} y_{i0} = 0$.

Due to our objective function, this formulation works only when there are items to deliver in all customers. In Section 3.2 we present a reformulation of the problem to accept $d_j = 0$ for any customer j, which is used to solve the VRPMPD-H and the VRPDPD-H.

2.2. Valid inequalities

In order to tighten the formulation of Section 2.1, we present some valid inequalities which strengthen some of the constraints. For the delivery items on the arcs, we know that the delivery load in the vehicle should be at least as large as the demand of the customer at the end of the arc, and a similar reasoning holds for the flow of pickup items:

$$\sum_{j\in V} y_{ji} \geq \sum_{j\in V} x_{ji} d_i, \qquad i \in V_c,$$

$$\sum_{j\in V} w_{ij} + z_{ij} \geq \sum_{j\in V} x_{ij} p_i, \qquad i \in V_c.$$

Next, capacity constraints (6) can be strengthened as follows (cf. Battarra et al. (2010)):

$$w_{ij} + y_{ij} + z_{ij} \leq x_{ij}\big(Q - \min\{0, p_i - d_i, d_j - p_j\}\big), \quad (i, j) \in A.$$

Additionally, when $h_p = h_d$ and the number of pickup items at a customer and the pickup items in the rear of the vehicle is higher than the remaining delivery items in the vehicle, the best policy to adopt is always the front policy. This is proven in the Proposition 1 in Section 5. Therefore, given a sufficiently large number M and a sufficiently small number ϵ, we can add the following inequalities to the model:

$$\sum_{j\in V} z_{ij} + p_i \leq \sum_{j\in V} y_{ij} + Ms_i - \epsilon, \qquad i \in V_c.$$

3. Special cases and related problems

In this section, we discuss how to solve problems related to the VRPSPD-H. We first prove that the VRPSPD and the TSPPD-H are special cases of the VRPSPD-H. Then, we show how to reformulate the model to solve the VRPMPD-H, the VRPMPD and the VRPDPD, and to find an upper bound to the VRPDPD-H.

3.1. Special cases

In this section, we show that the VRPSPD and the TSPPD-H are special cases of the VRPSPD-H.

Theorem 1. *The VRPSPD-H with $h_d = h_p = 0$ is equivalent to the VRPSPD.*

Proof. Let an instance be given with $h_d = h_p = 0$. As handling costs are zero, an optimal solution for the VRPSPD, which disregards handling operations, will also be optimal here. Hence, the objective function only comprises the routing component. We can also omit all constraints involving handling operations (constraints (7) and (9)), and remove variables w_{ij} and s_i entirely. Optionally, we can consider a maximum number of vehicles, so that the remaining model is equivalent to the VRPSPD as in Dell'Amico et al. (2006). □

Theorem 2. *The VRPSPD-H with a single vehicle and $Q \geq \max\left\{\sum_{i\in V_c} d_i, \sum_{i\in V_c} p_i\right\}$ is equivalent to the TSPPD-H.*

Proof. Let an instance be given with $Q \geq \max\left\{\sum_{i\in V_c} d_i, \sum_{i\in V_c} p_i\right\}$ and a single available vehicle, where the capacity restriction is obtained from the TSPPD-H formulation of Battarra et al. (2010). Then, the construction of a single route to service all customers is the only possibility of a feasible solution. The solution space of the VRPSPD-H shrinks to the solution space of the TSPPD-H. The remaining model is equivalent to the TSPPD-H as in Battarra et al. (2010) for the mixed handling policy when $d_i > 0$ for all customers i. □

3.2. Related problems

In this section we show how to reformulate the VRPSPD-H model to accept customers without delivery requests ($d_i = 0$), thus modeling the VRPMPD-H and the VRPMPD. We also show that by splitting pickup and delivery into new dummy customers, we are able to optimally solve the VRPDPD and to find an upper bound solution to the VRPDPD-H.

To solve the VRPMPD-H the objective function (1) is changed to

$$\text{minimize} \quad \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{(i,j)\in A\setminus A_r, d_i>0} \left(h_p z_{ij} + s_j h_d \left(y_{ij} - \frac{d_j}{|V_c|}\right)\right) + \sum_{(i,j)\in A\setminus A_r, d_i=0} s_j\big(h_p z_{ij} + h_d y_{ij}\big) \tag{11}$$

subject to (2)–(10).

The new objective function (11) explicitly considers handling costs at the stops where there are items to be delivered. The first and second terms follow directly from (1). According to the third term, when there are no delivery items and the rear policy is applied, the handling cost is zero since placing the pickup items does not require handling any item in the vehicle. Whereas if the front policy is applied, the handling cost is the same as when there are delivery items. The VRPMPD-H with $h_d = h_p = 0$ is equivalent to the VRPMPD, and the proof is similar to that of Theorem 1.

Now, consider a customer $i \in V_c$ with $d_i \geq 0$ and $p_i \geq 0$. Nagy et al. (2013) indicate that the VRPDPD may be modeled as a VRPMPD by creating two dummy customers i_1 ($d_{i_1} = d_i$ and $p_{i_1} = 0$) and i_2 ($d_{i_2} = 0$ and $p_{i_2} = p_i$) located at the location of i, for all customers in V_c. The same approach can be used to solve a VRPDPD-H from the VRPMPD-H model. However, the solution found is an upper bound to the original problem. The reason is described next.

Theorem 3. *Solving the VRPSPD-H with objective function (11) and with dummy customers to represent pickups and deliveries gives an upper bound to the VRPDPD-H.*

Proof. Consider a vehicle traveling along arc (i, j) loaded with $z_{ij} > 0$ and $y_{ij} > 0$ items. Consider also that customer j has $d_j > 0$

and $p_j > 0$. In the VRPSPD-H, the handling process in j when the front policy is applied is as follows:

(i) z_{ij} obstructing pickup items are removed from the vehicle;
(ii) d_j delivery items are removed and delivered to j;
(iii) the remaining $(y_{ij} - d_j)$ delivery items are removed from the vehicle;
(iv) z_{ij} items are placed back in the front of the vehicle;
(v) p_j pickup items are placed next;
(vi) $(y_{ij} - d_j)$ items are placed back lastly.

The total handling cost in this process is given by the handling of the z_{ij} and $(y_{ij} - d_j)$ obstructing items given by $z_{ij}h_p + (y_{ij} - d_j)h_d$. When customers are divided into dummy customers, to achieve the same final vehicle setting the process is slightly different. Steps (i) and (ii) are the same. Then, if the dummy customer representing deliveries is first in the route, the process is followed by steps (iii), (iv) and (vi) with a handling cost of $z_{ij}h_p + (y_{ij} - d_j)h_d$. Next, when visiting the other dummy customer representing pickups, it is necessary to remove the $(y_{ij} - d_j)$ obstructing items to place p_j at the front, increasing the handling cost by $(y_{ij} - d_j)h_d$ units. If the order of the dummy customers were inverted, the handling costs of the first stop would be equal to $z_{ij}h_p + y_{ij}h_d$, while the second stop would not require any handling. Consequently, regardless of the case, whenever a customer is divided into two dummy nodes and they are visited in sequence, and the rear policy is applied, the result for the handling cost of these two stops will be higher than the handling cost in a one-stop case. Thus, the solution of the VRPSPD-H with dummy customers is an upper bound to the solution of the VRPDPD-H. □

4. Adaptive large neighborhood search metaheuristic

This section provides details about the ALNS metaheuristic designed to solve the VRPSPD-H. As mentioned in Section 1, LNS was introduced by Shaw (1998) and this technique is growing in popularity over the last years due to recent successes in diverse settings. Among these settings are problems including simultaneous pickups and deliveries (Ropke and Pisinger (2006b)), handling costs (Veenstra et al. (2017)), multiple stacks (Côté et al. (2012)) and under other restrictions such as time windows (Ropke and Pisinger (2006a)). This diverse and successful application of ALNS provides grounds for us to employ this method as well. We use the framework of Ropke and Pisinger (2006a) as a basis for our design, enriching the framework with a local search procedure. The outline of our algorithm is given in Algorithm 1.

The procedure starts with the construction of an initial solution and by initializing the relevant parameters. Then, the algorithm enters its iterative phase which runs until the stopping criterion is met. In each iteration, the solution is changed by a destroy and repair mechanism. First, a destroy operator removes a number of customers from the solution. Next, a repair operator reinserts the removed customers to construct a new solution. If the resulting solution is better than the currently best solution, a local search procedure is applied to potentially improve the solution further and the best solution is updated. A simulated annealing criterion determines whether the changed solution is accepted as the new current solution. The destroy and repair operator weights are updated based on the performance of the selected operators in the current iteration. Finally, after a pre-specified number of iterations the destroy and repair operator weights are reset to their original values. Each time the weights are reset, the local search procedure is applied to the current solution to intensify the search. If the stopping criterion is not met, the algorithm goes to the next iteration and the process repeats. Experiments with applying the local search procedure to all accepted solutions or to all solutions within a certain threshold of the global best solution resulted in significantly higher calculation times without improving the solution quality.

Algorithm 1 Outline of ALNS metaheuristic.

```
 1: construct initial solution $s$
 2: solution $s_{best} \leftarrow s$
 3: **repeat**
 4:     $s' \leftarrow s$
 5:     destroy $s'$
 6:     repair $s'$
 7:     **if** $(f(s') < f(s_{best}))$ **then**
 8:         local search $s'$
 9:         $s_{best} \leftarrow s'$
10:     **end if**
11:     **if** accept$(s', s)$ **then**
12:         $s \leftarrow s'$
13:     **end if**
14:     update operator weights
15:     **if** operator weights are reset **then**
16:         local search $s$
17:     **end if**
18: **until** stopping criterion
19: **return** $s_{best}$
```

To evaluate the changes made by the metaheuristic, we evaluate the routing and handling costs separately. The routing cost difference is computed by assessing only the changed parts of a route since an unchanged route segment has the same cost in both cases. The handling sub-problem is solved using a handling policy among those that are presented in Section 5. The handling costs have to be recalculated for the entire route since handling decisions at the customers also depend on future customer visits. Feasibility of a change is assessed by checking if the vehicle capacity and number of used vehicles is within the allowed limits.

Details on the construction of an initial solution are reported in Sections 4.1–4.3 explain the destroy and repair operators, respectively. Section 4.4 reports details regarding the local search procedure, and Section 4.5 provides details about the acceptance criterion. Finally, Section 4.6 explains how the adaptive mechanism operates.

4.1. Initial solution

An initial solution is constructed by greedily inserting a random customer at its best position in the solution. The first customer to be inserted creates a new route, after which a random customer is inserted at its best feasible location considering the increase in total cost. If no feasible insertion can be found for the current customer, it is inserted in a new route. This process continues until all customers are inserted at a feasible position.

4.2. Destroy operators

In the destroy phase of the metaheuristic, a roulette wheel selection procedure randomly selects one destroy operator based on its weight. This operator removes a predefined number of q customers from the solution and places them in the customer pool. A total of eight different destroy operators are used and are described in this section. The *random removal* and *worst removal* operators are adapted from Ropke and Pisinger (2006a), whereas the *worst distance removal* and *worst handling removal* operators were introduced by Veenstra et al. (2017). The *related removal* operator was described by Shaw (1998) and the *route removal* and *minimum quantity removal* operators are also commonly seen in the destroy phase of a LNS heuristic. We newly introduce the *cross route removal* operator, inspired by the *cluster removal* operator described in Ropke and Pisinger (2006b). The destroy operators are explained below.

1. *Random removal*
   The *random removal* operator selects q customers randomly and removes them from the solution.
2. *Worst removal*
   The *worst removal operator* removes q customers based on their cost. Letting $f(x)$ be the objective value of solution x, the operator computes the cost of all customers in the solution as $c_i(s) = f(s) - f(s_{-i})$, which denotes the difference in objective value of the current solution s compared to the solution in which customer $i \in V_c$ is removed, s_{-i}. It then selects the y-th worst customer, with $y \sim \lceil U[0,1]^p \cdot N_s \rceil$, where N_s is the number of customers in the current solution and p is a measure of randomness. When a customer is removed, the costs of the remaining customers are recalculated and the process repeats until q customers are removed, as in Algorithm 2.

Algorithm 2 Outline of *worst removal* operator.

```
1: while number of customers in the pool < $q$ do
2:     $c_i(s) \leftarrow f(s) - f(s_{-i}),\ \forall i \in V_c$
3:     $y \sim \lceil U[0,1]^p \cdot (\text{number of customers in the solution}) \rceil$
4:     remove customer with $y$-th highest $c_i(s)$
5: end while
```

3. *Worst distance removal*
   The *worst distance removal* operator is similar to the *worst removal* operator. The difference is in the evaluation of the cost of customer $i \in V_c$. This cost, $\tilde{c}_i(s) = f^d(s) - f^d(s_{-i})$, is now only the difference in routing cost. Again here, the y-th worst customer is removed.
4. *Worst handling removal*
   The *worst handling removal* operator is similar to the *worst distance removal* operator. The cost of customer $i \in V_c$ is computed solely as the difference in handling cost.
5. *Related removal*
   The *related removal* operator removes customers which are related to each other. Such customers are likely to be exchanged more easily whereas more unique customers are often repaired in their original position and hence do not aid much in the diversification of the search process. For the *related removal* we define the relatedness measure $R(i, j)$ between customers i and j, $i, j \in V_c, i \neq j$, as the inverse of their mutual distance so that customers located close to each other have a high relatedness score. That is, $R(i, j) = \frac{1}{c_{ij}}$.
6. *Route removal*
   The *route removal* operator randomly selects a route from the solution. This selection is purely based on the number of routes and does not take into account the length of the routes, such that smaller routes are selected as often as larger routes. This is a desirable property due to the ease of diversification of the search when removing a small route. If the number of customer in the selected route, q_r, is smaller than q, the route is removed and the *route removal* operator restarts with $q' = q - q_r$. If there are more than q customers in the route, the operator randomly selects q customers from the given route and removes them.
7. *Minimum quantity removal*
   The *minimum quantity removal* operator removes customers with low demand quantity, computed as the sum of pickup and delivery demand per customer. The intuition behind this operator is that customers with low demand do not affect capacity restrictions much and are therefore more easily moved around than customers with high demand. Selection of the customer to be removed is similar as for the *worst removal* operator, i.e., removes the y-th lowest demand customer using the same measure of randomness $y \sim \lceil U[0,1]^p \cdot N_s \rceil$.
   When the customer pool is empty, the *related removal* operator selects a random customer in the solution and removes both this and a related customer from the solution. Then, as long as the customer pool does not contain q customers, a random customer from the pool is selected and a related customer in the solution is found which is then removed as well. The selection of a related customer is similar to the *worst removal* operator. Experiments with including the pickup and delivery demands as a term in the relatedness measure yielded no significant improvement.
8. *Cross route removal*
   In the *cross route removal* operator, a random customer in the solution is selected, as well as the customers immediately preceding and succeeding the selected customer, if present. Next, using the relatedness measure $R(i, j)$, a related customer of the initially selected customer in a different route and its neighboring customers are selected. All these customers, or at most q, are removed from the solution. If there are more than q customers selected, we remove the last ones starting from the neighbors of the related customer, then the related customer itself, and finally the neighbors of the initial customer, until there are only q customers to be removed. If there are fewer than q, this process repeats until there are q customers in the pool. This operator intensifies variation between routes as route chunks with related customers from different routes are destroyed in one iteration.

4.3. Repair operators

After a destroy operator has placed q customer in the customer pool, a repair operator is randomly selected which inserts all customers back into the solution. Similar to the selection of the destroy operator, a roulette wheel selection procedure randomly selects a repair operator based on its weight. The three repair operators employed in the reparation phase are explained is this section. The *random repair* operator is commonly seen in the literature, and the *sequential best insertion* operator is adopted from Veenstra et al. (2017). We have created a perturbed version of the *sequential best insertion* operator to prevent repeating the same insertions. The repair operators are explained below.

1. *Random repair*
   The *random repair* operator randomly selects a customer from the pool and inserts it at a random feasible location in the solution.
2. *Sequential best insertion*
   The *sequential best insertion* operator randomly selects a customer from the customer pool and inserts it at its best feasible location. It is a greedy, but therefore fast, operator.
3. *Perturbed sequential best insertion*
   The *perturbed sequential best insertion* operator diversifies the *sequential best insertion* operator to break out of potential local optima. It randomly selects a customer from the customer pool and inserts it at its y-th best location, where y is a random integer between 0 and min{3, number of feasible insertion locations}.

We have also experimented with two more calculation-intensive repair operators, both originating from Ropke and Pisinger (2006a). These are the *best insertion operator*, which inserts the overall best customer and recalculates the costs for the remaining customers after each insertion, and *regret insertion operator*, which inserts the customer with largest difference between its best and second best insertion location and recalculates regret values for the remaining customers after each insertion. However, inclusion of these operators did not improve the solution quality while calculation times increased significantly, so we excluded them from our final metaheuristic.

4.4. Local search

When a new best solution is found or when the destroy and repair operator weights are reset, a local search procedure is performed to try and improve the current solution. The following five operators, applied in this order, are used: *reinsertion, exchange, intra 2-opt, inter 2-opt* and *inter 3-opt*. The *reinsertion* operator finds the best reinsertion of a single customer, and the *exchange* operator performs the best exchange of two customers. The *intra 2-opt* operator performs the best possible 2-opt move within a route, whereas *inter 2-opt* performs the best 2-opt move between two routes. Finally, the *inter 3-opt* operator removes a block of two or more successive customers and inserts it at its best position in a different route. Whenever an operator finds an improvement and changes the solution, the process restarts by applying *reinsertion* again. The process continues until no improvement can be found. An outline of the local search procedure is presented in Algorithm 3.

Algorithm 3 Outline of local search procedure.

```
1:  for $k$ in 1 : 5 do
2:      improve ← true
3:      while improve do
4:          improve ← false
5:          Apply operator $k$ to solution
6:          if improvement found then
7:              $k \leftarrow 1$
8:              improve = true
9:          end if
10:     end while
11: end for
12: return (improved) solution
```

4.5. Acceptance decision

A new solution s' is accepted based on a simulated annealing decision rule. If the new solution is better than the previous one (s), it is always accepted. Otherwise we accept it with probability

$$P(\text{accept } s') = \exp\left\{-\frac{f(s') - f(s)}{T}\right\}, \tag{12}$$

where T is the temperature at the current iteration. The starting temperature is determined at the start of the metaheuristic, and it is decreased in every iteration by multiplying the temperature of the previous iteration with the cooling rate $\gamma \in (0, 1)$.

4.6. Updating operator weights

The updating procedure is based on the work of Ropke and Pisinger (2006a). In order to update the weights of the destroy and repair operators, their performance is determined by means of three measures: (i) a new best solution is found, (ii) the current solution is improved, yet the global best remains the same, and (iii) the solution is accepted as the new one without improving its objective. Each of these three events leads to increasing the weight of both the destroy and repair operator used in the current iteration by a factor σ_i, $i = 1, 2, 3$. Since we cannot differentiate the effect of the destroy and repair operators in one iteration, both are updated with the same amount. If none of the three scenarios occurs, the weights remain the same. After a predetermined number of iterations, the weights are reset to their initial values since different phases of the search may require different operators. Ropke and Pisinger (2006a) reset the operator weights at the start of a new segment to values which depend on the performance in the previous segment. However, we found that resetting the weights to the original values results in equally good solutions.

5. Handling policies

To solve the handling sub-problem, we adopt the mixed policy of Battarra et al. (2010), in which decisions on either placing pickup items at the rear or at the front of the vehicle are possible. Erdoğan et al. (2012) extensively studied the handling sub-problem. The main difficulty of the handling policy is to decide when to place the pickup items of a customer at the rear of the vehicle, and when to place all pickup items at the front of the vehicle so that they never obstruct future deliveries. This problem is modeled as a dynamic program (DP) in Erdoğan et al. (2012) and it gives the optimal choices for any given route in $O(n^2)$ time. This *DP policy* is one of the policies we use to calculate handling costs.

Due to the time complexity of the DP policy, Erdoğan et al. (2012) also propose a linear-time heuristic to solve the handling sub-problem. When designing a heuristic for the mixed policy, the objective is to determine a threshold which triggers the placement of all pickup items at the front of the vehicle. If this threshold is not triggered, the pickup items are placed at the rear. In their heuristic, Erdoğan et al. (2012) experimented with four different thresholds, showing the best one was the average of all pickup items of the remaining customers in the route. If the number of pickup items onboard at the rear of the vehicle exceeds this threshold, all pickup items are placed at the front of the vehicle. The authors conclude that using this heuristic reduces computation time substantially at the cost of only slightly worse solutions. This linear-time *heuristic policy* is also included here for comparisons.

Next, we report on some useful properties of the handling subproblem to propose two new thresholds based on bounds on the optimal solution. These two newly introduced methods are called myopic handling policies. The performance of our myopic handling policies is studied in Sections 6.4 and 6.5. We first introduce new notation before we propose the thresholds.

Let a route with $n \le |V_c|$ customers (depot not included) be denoted by a permutation $\phi(\ \cdot\ )$ of the location indices, such that $\phi(k)$ is the index of the k-th customer on the route for $k = \{1, \dots, n\}$. We consider the decision at customer k whether or not to place the pickup items at the front of the vehicle, given handling decisions at all customers $\phi(l)$ for $l < k$. Consistent with our notation in Section 2.1, we use $s_{\phi(k)}$ to represent the handling decisions, where $s_{\phi(k)} = 1$ if the pickup items from customer $\phi(k)$ and the other pickup items that were in the rear of the vehicle are placed at the front of the vehicle when visiting customer $\phi(k)$, and $s_{\phi(k)} = 0$ if they are placed at the rear.

5.1. Upper bound and myopic policy 1

We show that there exist situations in which it is always optimal to place the pickup items at the front of the vehicle in Proposition 1.

Proposition 1. *Given handling decisions $s_{\phi(1)}, \dots, s_{\phi(k-1)}$, at customer $\phi(k)$, it is always optimal to place $p_{\phi(k)}$ and*

Table 1
Computational results for the VRPSPD benchmark instances of Dethloff (2001).

Instance	N	BKS	ALNS: best	Gap (%)	ALNS: average	Gap (%)	Time (s)
SCA3-0	50	635.62	636.06	0.07	639.20	0.56	7.6
SCA3-1	50	697.84	**697.84**	0.00	**697.84**	0.00	7.7
SCA3-2	50	659.34	**659.34**	0.00	660.20	0.13	7.9
SCA3-3	50	680.04	**680.04**	0.00	680.27	0.03	8.0
SCA3-4	50	690.50	**690.50**	0.00	**690.50**	0.00	8.1
SCA3-5	50	659.91	**659.91**	0.00	661.59	0.25	8.2
SCA3-6	50	651.09	**651.09**	0.00	651.18	0.01	8.5
SCA3-7	50	659.17	666.15	1.05	667.69	1.28	8.7
SCA3-8	50	719.48	**719.48**	0.00	**719.48**	0.00	8.8
SCA3-9	50	681.00	**681.00**	0.00	**681.00**	0.00	8.9
SCA8-0	50	961.50	**961.50**	0.00	965.97	0.46	7.4
SCA8-1	50	1049.65	1050.38	0.07	1062.05	1.17	7.6
SCA8-2	50	1039.64	1047.95	0.79	1050.67	1.05	7.7
SCA8-3	50	983.34	**983.34**	0.00	1004.03	2.06	8.0
SCA8-4	50	1065.49	1067.55	0.19	1068.26	0.26	8.1
SCA8-5	50	1027.08	1040.08	1.25	1050.65	2.24	8.3
SCA8-6	50	971.82	972.49	0.07	974.60	0.29	8.4
SCA8-7	50	1051.28	1063.22	1.12	1066.94	1.47	8.4
SCA8-8	50	1071.18	**1071.18**	0.00	1072.86	0.16	8.4
SCA8-9	50	1060.50	**1060.50**	0.00	1066.47	0.56	8.5
CON3-0	50	616.52	**616.52**	0.00	618.21	0.27	8.1
CON3-1	50	554.47	**554.47**	0.00	555.55	0.19	8.2
CON3-2	50	518.00	521.38	0.65	521.38	0.65	8.8
CON3-3	50	591.19	**591.19**	0.00	**591.19**	0.00	8.9
CON3-4	50	588.79	**588.79**	0.00	590.85	0.35	9.3
CON3-5	50	563.70	**563.70**	0.00	564.17	0.08	9.3
CON3-6	50	499.05	**499.05**	0.00	501.55	0.50	9.4
CON3-7	50	576.48	**576.48**	0.00	577.29	0.14	9.5
CON3-8	50	523.05	**523.05**	0.00	523.48	0.08	9.7
CON3-9	50	578.25	578.31	0.01	584.11	1.00	9.9
CON8-0	50	857.17	**857.17**	0.00	860.50	0.39	7.7
CON8-1	50	740.85	750.38	1.27	752.75	1.58	8.2
CON8-2	50	712.89	**712.89**	0.00	713.64	0.10	8.3
CON8-3	50	811.07	821.26	1.24	821.66	1.29	8.3
CON8-4	50	772.25	**772.25**	0.00	777.31	0.65	8.4
CON8-5	50	754.88	**754.88**	0.00	758.28	0.45	8.5
CON8-6	50	678.92	681.38	0.36	691.02	1.75	8.5
CON8-7	50	811.96	814.79	0.35	814.79	0.35	8.5
CON8-8	50	767.53	**767.53**	0.00	775.83	1.07	8.7
CON8-9	50	809.00	**809.00**	0.00	811.12	0.26	8.7
Average		758.54	760.35	0.21	763.40	0.58	8.4

$\sum_{l=1}^{k-1}\left(\prod_{m=l}^{k-1}\left(1-s_{\phi(m)}\right)p_{\phi(l)}\right)$ *at the front of the vehicle if*

$$h_p\left[p_{\phi(k)}+\sum_{l=1}^{k-1}\left(\prod_{m=l}^{k-1}\left(1-s_{\phi(m)}\right)p_{\phi(l)}\right)\right]>h_d\sum_{l=k+1}^{n}d_{\phi(l)}. \tag{13}$$

Proof. Assume a route with $n \geq 2$ customers, and that customer $\phi(k)$ is not the last customer in the route. Let handling decisions $s_{\phi(1)},\ldots,s_{\phi(k-1)}$ be given. There are two options. Option 1 is to place $p_{\phi(k)}$ at the rear of the vehicle with cost $h_p\sum_{l=1}^{k-1}\left((\prod_{m=l}^{k-1}\left(1-s_{\phi(m)}\right)p_{\phi(l)}\right)$ at customer $\phi(k)$ and cost $h_p\left(p_{\phi(k)}+\sum_{l=1}^{k-1}\left(\prod_{m=l}^{k-1}\left(1-s_{\phi(m)}\right)p_{\phi(l)}\right)\right)$ at customer $\phi(k+1)$. Option 2 is to place $p_{\phi(k)}$ and $\sum_{l=1}^{k-1}\left(\prod_{m=l}^{k-1}\left(1-s_{\phi(m)}\right)p_{\phi(l)}\right)$ at the front of the vehicle with cost $h_p\sum_{l=1}^{k-1}\left(\prod_{m=l}^{k-1}\left(1-s_{\phi(m)}\right)p_{\phi(l)}\right)+h_d\sum_{l=k+1}^{n}d_{\phi(l)}$ at customer $\phi(k)$ and cost 0 at customer $\phi(k+1)$. Inequality (13) follows from this. It can then be seen that placing $p_{\phi(k)}$, and thus also $\sum_{l=1}^{k-1}\left(\prod_{m=l}^{k-1}\left(1-s_{\phi(m)}\right)p_{\phi(l)}\right)$, at the front of the vehicle is always optimal if (13) holds. That is, given handling decisions at all customers visited prior to arriving at customer $\phi(k)$, it is optimal to place the pickup items at the front of the vehicle if the costs of handling the number of pickup items at the rear of the vehicle plus the pickup items of customer $\phi(k)$, exceed the costs of handling the number of items that still need to be delivered. □

Based on Proposition 1, we introduce *myopic policy 1*. Under myopic policy 1, the pickup items at the rear of the vehicle and the pickup items of customer $\phi(k)$ are placed at the front of the vehicle if and only if inequality (13) holds. Otherwise, they are placed at the rear.

5.2. Lower bound and myopic policy 2

Similar as in Section 5.1, and using the same notation, we show that there exist situations in which it is always optimal to place the pickup items at the rear of the vehicle in Proposition 2.

Proposition 2. *Given handling decisions $s_{\phi(1)},\ldots,s_{\phi(k-1)}$, at customer $\phi(k)$, it is always optimal to place $p_{\phi(k)}$ at the rear of the vehicle if*

$$h_p(n-k)\left[p_{\phi(k)}+\sum_{l=1}^{k-1}\left(\prod_{m=l}^{k-1}\left(1-s_{\phi(m)}\right)p_{\phi(l)}\right)\right]<h_d\sum_{l=k+1}^{n}d_{\phi(l)}. \tag{14}$$

Proof. Assume a route with $n \geq 2$ customers, that customer $\phi(k)$ is not the last customer in the route, and that $p_{\phi(l)}$, $k < l \leq n$ are placed at the rear of the vehicle. Let handling decisions $s_{\phi(1)},\ldots,s_{\phi(k-1)}$ be given. There are two options.

Table 2
Computational results for the VRPSPD benchmark instances of Salhi and Nagy (1999).

CMT	N	BKS	ALNS: best	Gap (%)	ALNS: average	Gap (%)	Time (s)
1X	50	470	**468**	−0.43	**469.2**	−0.16	6.2
1Y	50	459	**459**	0.00	**459.0**	0.00	6.3
2X	75	685	**677**	−1.18	687.5	0.36	16.8
2Y	75	651	**651**	0.00	653.8	0.43	16.9
3X	100	714	**709**	−0.71	**712.7**	−0.18	35.8
3Y	100	705	**702**	−0.43	708.6	0.51	34.3
4X	150	862	**853**	−1.06	863.0	0.12	118.5
4Y	150	831	**821**	−1.22	**824.6**	−0.77	111.1
5X	199	1063	**1039**	−2.31	**1052.7**	−0.98	276.7
5Y	199	982	985	0.30	997.1	1.51	243.7
11X	120	874	**871**	−0.34	895.4	2.39	79.8
11Y	120	826	**818**	−0.98	**822.7**	−0.40	84.1
12X	100	672	**670**	−0.30	675.8	0.57	37.5
12Y	100	632	**629**	−0.48	639.9	1.23	31.1
Average		744.71	739.43	−0.65	747.30	0.33	78.5

Table 3
Computational results for the VRPDPD benchmark instances of Dethloff (2001).

CMT	N	BKS	ALNS: best	Gap (%)	ALNS: average	Gap (%)	Time (s)
SCA3-0	50	635.62	636.06	0.07	639.20	0.56	30.2
SCA3-1	50	697.84	**697.84**	0.00	**697.84**	0.00	32.1
SCA3-2	50	659.34	**659.34**	0.00	660.20	0.13	31.9
SCA3-3	50	680.04	**680.04**	0.00	680.27	0.03	32.6
SCA3-4	50	690.50	**690.50**	0.00	**690.50**	0.00	31.1
SCA3-5	50	659.91	**659.91**	0.00	661.35	0.22	32.7
SCA3-6	50	651.09	**651.09**	0.00	651.18	0.01	30.8
SCA3-7	50	659.17	666.15	1.05	667.69	1.28	30.9
SCA3-8	50	719.48	**719.48**	0.00	**719.48**	0.00	33.0
SCA3-9	50	681.00	**681.00**	0.00	**681.00**	0.00	31.9
SCA8-0	50	961.50	**961.50**	0.00	965.97	0.46	28.2
SCA8-1	50	1049.65	1050.38	0.07	1061.79	1.14	27.4
SCA8-2	50	1039.64	1047.95	0.79	1050.16	1.00	27.7
SCA8-3	50	979.13	983.34	0.43	1003.90	2.47	27.3
SCA8-4	50	1065.49	1065.81	0.03	1067.90	0.23	26.9
SCA8-5	50	1027.08	1036.57	0.92	1046.31	1.84	27.7
SCA8-6	50	969.50	970.97	0.15	973.89	0.45	27.4
SCA8-7	50	1051.28	1060.45	0.86	1066.34	1.41	27.5
SCA8-8	50	1071.18	**1071.18**	0.00	1072.86	0.16	27.9
SCA8-9	50	1057.26	1058.07	0.08	1065.66	0.79	27.0
CON3-0	50	616.52	**616.52**	0.00	617.90	0.22	32.6
CON3-1	50	554.47	**554.47**	0.00	555.55	0.19	32.6
CON3-2	50	518.00	521.38	0.65	521.38	0.65	31.6
CON3-3	50	591.19	**591.19**	0.00	**591.19**	0.00	31.5
CON3-4	50	588.79	**588.79**	0.00	590.85	0.35	31.1
CON3-5	50	563.70	**563.70**	0.00	563.93	0.04	31.0
CON3-6	50	499.05	**499.05**	0.00	501.49	0.49	32.8
CON3-7	50	576.48	**576.48**	0.00	577.29	0.14	31.3
CON3-8	50	523.05	**523.05**	0.00	523.48	0.08	32.1
CON3-9	50	578.25	578.31	0.01	584.11	1.00	29.9
CON8-0	50	857.12	**857.12**	0.00	860.49	0.39	27.2
CON8-1	50	739.44	750.38	1.46	752.57	1.75	27.9
CON8-2	50	706.51	**706.51**	0.00	710.18	0.52	29.5
CON8-3	50	811.07	821.26	1.24	821.63	1.28	28.4
CON8-4	50	771.30	**771.30**	0.00	776.54	0.67	28.3
CON8-5	50	754.88	**754.88**	0.00	758.00	0.41	28.1
CON8-6	50	678.92	681.38	0.36	689.77	1.57	28.6
CON8-7	50	811.96	814.79	0.35	814.79	0.35	27.9
CON8-8	50	766.99	767.53	0.07	775.83	1.14	28.6
CON8-9	50	797.69	804.79	0.88	809.80	1.50	28.5
Average		757.78	759.76	0.24	763.01	0.62	29.8

Option 1 is to place $p_{\phi(k)}$ at the rear of the vehicle at cost $h_p(n-k)\sum_{l=1}^{k-1}\left(\prod_{m=l}^{k-1}\left(1-s_{\phi(m)}\right)p_{\phi(l)}\right) + h_p\sum_{m=k}^{n-1}(n-m)p_{\phi(m)}$ for the remainder of the route. Option 2 is to place $p_{\phi(k)}$ and $\sum_{l=1}^{k-1}\left(\prod_{m=l}^{k-1}\left(1-s_{\phi(m)}\right)p_{\phi(l)}\right)$ at the front of the vehicle at cost $h_d\sum_{l=k+1}^{n} d_{\phi(l)} + h_p\sum_{m=k+1}^{n-1}(n-m)p_{\phi(m)}$ for the remainder of the route. It follows that if

$$h_p\left((n-k)\sum_{l=1}^{k-1}\left(\prod_{m=l}^{k-1}\left(1-s_{\phi(m)}\right)p_{\phi(l)}\right)+\sum_{m=k}^{n-1}(n-m)p_{\phi(m)}\right) < h_d\sum_{l=k+1}^{n} d_{\phi(l)} + h_p\sum_{m=k+1}^{n-1}(n-m)p_{\phi(m)} \iff$$

Table 4
Computational results for the VRPDPD benchmark instances of Salhi and Nagy (1999).

CMT	N	BKS	ALNS: best	Gap (%)	ALNS: average	Gap (%)	Time (s)
1X	50	470	**468**	−0.43	**469.2**	−0.16	17.8
1Y	50	459	**459**	0.00	**459.0**	0.00	17.7
2X	75	684	**676**	−1.18	**682.5**	−0.22	48.7
2Y	75	650	**650**	0.00	653.2	0.49	48.1
3X	100	713	**703**	−1.42	**706.3**	−0.94	102.4
3Y	100	705	**698**	−1.00	**704.1**	−0.12	103.7
4X	150	862	**847**	−1.77	**853.9**	−0.94	302.7
4Y	150	831	**817**	−1.71	**819.0**	−1.47	282.2
5X	199	1062	**1025**	−3.61	**1038.0**	−2.31	740.9
5Y	199	982	**972**	−1.03	991.2	0.93	627.8
11X	120	873	**868**	−0.58	892.8	2.22	231.9
11Y	120	826	**818**	−0.98	**819.7**	−0.77	229.1
12X	100	672	**667**	−0.75	674.0	0.30	103.9
12Y	100	632	**629**	−0.48	638.8	1.06	93.9
Average		737.21	735.50	−0.19	743.00	0.74	210.8

Table 5
Computational results for the VRPMPD benchmark instances of Salhi and Nagy (1999).

CMT	N	BKS	ALNS: best	Gap (%)	ALNS: average	Gap (%)	Time (s)
1H	50	465	**462**	−0.65	**462.1**	−0.62	6.4
1Q	50	490	**488**	−0.41	**488.0**	−0.41	6.4
1T	50	520	**516**	−0.78	**516.0**	−0.78	5.8
2H	75	663	**658**	−0.76	**659.8**	−0.48	17.0
2Q	75	733	**729**	−0.55	736.3	0.45	17.0
2T	75	783	**783**	0.00	786.6	0.46	16.6
3H	100	701	**690**	−1.59	704.1	0.45	32.4
3Q	100	747	**736**	−1.49	**736.8**	−1.38	37.2
3T	100	798	**790**	−1.01	**795.8**	−0.28	33.1
4H	150	829	**824**	−0.61	829.4	0.05	104.4
4Q	150	918	**911**	−0.77	919.3	0.15	115.3
4T	150	1000	**984**	−1.63	**993.7**	−0.63	133.2
5H	199	983	**982**	−0.10	1008.8	2.56	251.9
5Q	199	1119	1130	0.97	1143.1	2.11	283.2
5T	199	1227	1234	0.57	1250.7	1.89	275.1
11H	120	818	**814**	−0.49	831.1	1.58	89.1
11Q	120	939	**932**	−0.75	**938.2**	−0.08	74.3
11T	120	1000	1031	3.01	1075.3	7.01	71.9
12H	100	629	635	0.94	650.7	3.34	33.8
12Q	100	729	729	0.00	737.6	1.17	35.6
12T	100	788	789	0.13	804.9	2.10	38.9
Average		744.71	739.43	−0.65	747.30	0.33	78.5

$$h_p\left((n-k)\sum_{l=1}^{k-1}\left(\prod_{m=l}^{k-1}\left(1-s_{\phi(m)}\right)p_{\phi(l)}\right)+(n-k)p_{\phi(k)}\right)$$
$$< h_d\sum_{l=k+1}^{n} d_{\phi(l)} \iff$$
$$h_p(n-k)\left[p_{\phi(k)}+\sum_{l=1}^{k-1}\left(\prod_{m=l}^{k-1}\left(1-s_{\phi(m)}\right)p_{\phi(l)}\right)\right] < h_d\sum_{l=k+1}^{n} d_{\phi(l)}$$

holds, it is always optimal to place $p_{\phi(k)}$ at the rear of the vehicle. That is, given handling decisions at all customers visited prior to arriving at customer $\phi(k)$, it is optimal to place the pickup items of customer $\phi(k)$ at the rear of the vehicle if the costs of placing the pickup items at the front of the vehicle exceed the costs of handling the pickup items located at the rear of the vehicle at all subsequent stops. □

Based on Proposition 2, we introduce *myopic policy 2*. Under myopic policy 2, the pickup items of customer $\phi(k)$ are placed at the rear of the vehicle if and only if inequality (14) holds. Otherwise, they are placed at the front.

We note that the pickup items of the last customer in any route never require additional operations. Additionally, at the penultimate customer in any route, inequalities (13) and (14) give the same decision, indicating that this is the optimal decision.

6. Computational results

The ALNS metaheuristic was programmed in C++ and the mathematical model of Section 2.1 was implemented in C++ and solved with CPLEX 12.7.1. Since CPLEX can handle some non-linear functions, such as ours, we did not have to linearize the objective functions and constraints (7). That would be simple using the technique of Glover and Woolsey (1974). All our experiments were run on a 2.4 GHz Intel Xeon Gold 6148 processor with a limit of 16 GB of memory per run. We provide details on the parameter configuration in Section 6.1. We test our ALNS metaheuristic on well-known benchmark instances of the VRPSPD, VRPDPD, VRPMPD, and TSPPD-H, which are special cases and related problems of the VRPSPD-H, in Section 6.2. A comparison with optimal solutions for the VRPSPD-H, VRPDPD-H, and VRPMPD-H is made in Section 6.3. We compare the performance of the heuristic handling policy and the two myopic policies for these same problems in Section 6.4. Finally, we investigate the influence of the number of available vehicles on the trade-off between routing and handling costs in Section 6.5.

6.1. Tuning

This section provides details about the parameter settings of the proposed ALNS metaheuristic. For the purpose of tuning the

Table 6
Computational results for the TSPPD-H instances of Erdoğan et al. (2012).

N	*Id.*	Erdoğan et al. (2012)			ALNS with heuristic policy				ALNS with DP policy			
		DP	Heuristic	Gap (%)	Best	Average	Gap (%)	Time (*s*)	Best	Average	Gap (%)	Time (*s*)
20	1	633.0	634.0	0.16	**633.0**	**633.0**	0.00	3	**633.0**	**633.0**	0.00	7
	2	584.0	587.0	0.51	**584.0**	**584.0**	0.00	3	**584.0**	**584.0**	0.00	7
	3	573.0	590.0	2.88	**573.0**	575.8	0.00	3	**573.0**	**573.0**	0.00	7
	4	706.0	712.0	0.84	**706.0**	707.3	0.00	3	**706.0**	**706.0**	0.00	7
	5	501.0	507.0	1.18	**501.0**	**501.0**	0.00	3	**501.0**	**501.0**	0.00	7
	6	578.0	**578.0**	0.00	**578.0**	**578.0**	0.00	3	**578.0**	**578.0**	0.00	7
	7	612.0	619.0	1.13	**612.0**	**612.0**	0.00	3	**612.0**	**612.0**	0.00	8
	8	567.0	571.0	0.70	**567.0**	**567.0**	0.00	3	**567.0**	**567.0**	0.00	7
	9	604.0	625.0	3.36	**604.0**	605.6	0.00	3	**604.0**	604.8	0.00	18
	10	574.0	588.0	2.38	**565.0**	**565.0**	-1.59	3	**565.0**	**570.1**	−1.59	18
40	1	909.5	915.5	0.66	**909.5**	910.7	0.00	18	913.5	913.5	0.44	61
	2	885.0	902.0	1.88	**883.0**	**884.3**	-0.23	18	**883.0**	890.2	-0.23	63
	3	815.5	864.0	5.61	**815.5**	820.5	0.00	18	**815.5**	816.6	0.00	61
	4	898.0	919.5	2.34	**898.0**	898.6	0.00	18	**898.0**	898.3	0.00	65
	5	743.5	751.5	1.06	**743.5**	744.3	0.00	18	745.0	745.0	0.20	63
	6	901.0	903.0	0.22	**883.5**	**884.3**	-1.98	18	**883.5**	**885.7**	-1.98	64
	7	798.5	833.5	4.20	**798.5**	800.1	0.00	18	**798.5**	802.9	0.00	65
	8	795.0	811.0	1.97	**795.0**	795.2	0.00	18	**795.0**	800.1	0.00	64
	9	876.5	895.5	2.12	**876.5**	879.5	0.00	18	**876.5**	878.9	0.00	64
	10	866.0	901.5	3.94	**862.5**	**862.8**	−0.41	18	**862.5**	**862.5**	−0.41	63
60	1	1060.1	1063.8	0.34	**1051.0**	**1053.2**	−0.87	53	**1051.0**	**1054.0**	−0.87	263
	2	1051.1	1060.9	0.92	**1044.3**	**1049.4**	−0.65	55	**1047.3**	**1048.7**	−0.36	266
	3	990.4	1012.4	2.18	993.7	993.7	0.33	54	993.7	1010.5	0.33	261
	4	1061.4	1086.3	2.29	1066.0	1073.4	0.43	54	1066.0	1068.7	0.43	268
	5	986.9	1033.7	4.52	989.7	991.3	0.28	55	989.7	992.5	0.28	267
	6	1086.3	1131.3	3.98	**1073.3**	**1079.8**	−1.21	55	**1067.7**	**1072.6**	−1.75	268
	7	1005.4	1018.0	1.23	1007.3	1020.4	0.19	48	1007.3	1007.3	0.19	267
	8	1027.2	1059.5	3.05	1031.0	1034.6	0.37	54	1031.0	1033.5	0.37	264
	9	1001.4	1013.9	1.23	1004.3	1004.3	0.29	54	1004.3	1004.3	0.29	264
	10	1062.1	1081.7	1.81	**1048.7**	**1049.3**	−1.28	54	**1048.7**	1062.7	−1.28	274
120	1	1472.5	1526.5	3.54	**1436.7**	**1448.9**	−2.49	309	**1442.6**	**1442.6**	−2.07	3569
	2	1482.3	1540.2	3.76	**1455.5**	**1468.4**	−1.84	312	**1459.6**	**1459.6**	−1.56	3517
	3	1510.0	1550.1	2.58	**1465.5**	**1478.9**	−3.04	316	**1483.9**	**1483.9**	−1.76	3561
	4	1563.5	1628.8	4.01	**1543.7**	**1549.3**	−1.28	361	**1549.5**	**1549.5**	−0.90	3550
	5	1457.0	1543.0	5.57	**1430.7**	**1454.7**	−1.84	360	**1437.5**	**1437.5**	−1.36	3536
	6	1546.2	1622.3	4.69	**1528.0**	**1538.3**	−1.19	365	**1540.7**	**1540.7**	−0.36	3599
	7	1557.3	1610.9	3.33	**1517.7**	**1526.3**	−2.61	403	**1520.0**	**1520.0**	−2.45	3578
	8	1524.1	1599.6	4.72	**1504.8**	**1518.0**	−1.28	391	**1517.2**	**1517.2**	−0.45	3572
	9	1547.9	1620.3	4.46	**1530.7**	**1535.6**	−1.13	403	**1528.9**	**1528.9**	−1.25	3613
	10	1573.9	1622.4	2.99	**1540.8**	**1550.0**	−2.14	396	**1548.7**	**1548.7**	−1.63	3613
140	1	1575.2	1645.6	4.28	1589.0	1599.1	0.87	630	1586.6	1586.6	0.72	6655
	2	1605.8	1670.8	3.89	**1584.9**	**1601.5**	−1.32	631	**1596.8**	**1596.8**	−0.56	6808
	3	1583.7	1634.4	3.10	**1571.3**	1587.3	−0.79	628	**1572.8**	**1572.8**	−0.69	6760
	4	1712.7	1760.6	2.72	**1682.7**	**1700.4**	−1.78	634	**1694.1**	**1694.1**	−1.09	6721
	5	1547.4	1610.5	3.92	1549.1	1563.2	0.11	621	**1542.9**	**1542.9**	−0.29	6712
	6	1662.3	1741.2	4.53	**1631.3**	**1656.8**	−1.90	1557	**1659.3**	**1659.3**	−0.18	6804
	7	1664.4	1691.4	1.59	**1621.4**	**1638.4**	−2.65	1549	**1633.9**	**1633.9**	−1.87	6818
	8	1646.1	1698.1	3.06	**1611.0**	**1637.2**	−2.18	624	**1628.7**	**1628.7**	−1.06	6808
	9	1629.1	1700.0	4.17	1645.3	1653.3	0.98	637	1643.5	1643.5	0.88	6932
	10	1693.8	1717.3	1.37	**1671.9**	**1686.0**	−1.31	623	**1683.1**	**1683.1**	−0.64	6832
160	1	1741.7	1836.6	5.17	**1712.6**	**1730.0**	−1.70	971	**1713.2**	**1713.2**	−1.67	11,383
	2	1773.6	1820.5	2.57	**1725.7**	**1746.2**	−2.77	974	**1730.3**	**1730.3**	−2.50	11,570
	3	1690.9	1748.4	3.29	**1689.7**	1712.8	−0.07	933	**1684.3**	**1684.3**	−0.39	11,287
	4	1858.1	1962.6	5.32	**1837.5**	**1848.7**	−1.12	946	**1842.8**	**1842.8**	−0.83	11,540
	5	1667.7	1756.7	5.07	**1658.0**	1677.0	−0.59	945	**1654.1**	**1654.1**	−0.82	11,479
	6	1813.0	1844.5	1.71	**1726.7**	**1745.6**	−4.99	948	**1738.6**	**1738.6**	−4.28	11,698
	7	1774.2	1835.4	3.33	**1742.4**	**1758.3**	−1.83	957	**1753.7**	**1753.7**	−1.17	11,628
	8	1770.7	1820.5	2.73	**1749.2**	**1766.9**	−1.23	931	**1762.4**	**1762.4**	−0.47	11,528
	9	1800.9	1893.5	4.89	**1793.2**	1804.3	−0.43	952	**1786.7**	**1786.7**	−0.79	11,772
	10	1764.4	1809.4	2.49	**1757.6**	1779.3	−0.38	939	**1756.3**	**1756.3**	−0.46	11,639
180	1	1854.2	1936.0	4.23	**1818.9**	**1845.5**	−1.94	1347	**1833.8**	**1833.8**	−1.11	17,775
	2	1863.3	1927.5	3.33	**1838.9**	**1860.2**	−1.33	1351	**1851.4**	**1851.4**	−0.64	18,010
	3	1858.4	1889.9	1.66	**1830.9**	1860.3	−1.50	1331	**1822.1**	**1822.1**	−1.99	17,999
	4	1988.2	2059.5	3.46	**1942.8**	**1957.6**	−2.34	1363	**1943.9**	**1943.9**	−2.28	18,251
	5	1795.8	1853.7	3.12	**1785.4**	1804.1	−0.58	1353	**1785.2**	**1785.2**	−0.60	18,153
	6	1817.5	1873.2	2.97	1820.4	1851.4	0.16	1343	1826.4	1826.4	0.49	18,108
	7	1868.0	1929.6	3.19	**1848.3**	**1860.3**	−1.07	1358	**1846.8**	**1846.8**	−1.15	18,194
	8	1883.4	1930.5	2.44	**1862.2**	1884.1	−1.14	1326	**1865.3**	**1865.3**	−0.97	18,167
	9	1931.4	2004.6	3.65	1935.0	1945.3	0.18	1369	**1917.5**	**1917.5**	−0.73	18,385
	10	1852.5	1896.8	2.34	**1845.9**	1878.4	−0.36	1349	1857.6	1857.6	0.27	18,115

(*continued on next page*)

Table 6 (*continued*)

N	*Id.*	Erdoğan et al. (2012)			ALNS with heuristic policy				ALNS with DP policy			
		DP	Heuristic	Gap (%)	Best	Average	Gap (%)	Time (*s*)	Best	Average	Gap (%)	Time (*s*)
200	1	1976.3	2060.6	4.09	**1928.6**	**1961.9**	−2.47	1854	**1934.4**	**1934.4**	−2.16	27,134
	2	1982.0	2074.9	4.48	**1975.2**	1994.8	−0.34	1838	**1964.7**	**1964.7**	−0.88	27,031
	3	1976.7	2037.1	2.96	**1958.8**	1993.4	−0.91	1844	**1954.3**	**1954.3**	−1.15	26,724
	4	2119.5	2211.9	4.18	**2061.5**	**2079.9**	−2.81	1883	**2041.4**	**2041.4**	−3.83	27,354
	5	1905.6	1974.2	3.47	**1902.1**	1912.9	−0.18	1854	**1890.4**	**1890.4**	−0.80	27,247
	6	2011.1	2042.3	1.53	**1957.6**	**1978.1**	−2.73	1842	**1948.3**	**1948.3**	−3.23	27,467
	7	1983.0	2046.5	3.10	**1930.4**	**1954.3**	−2.72	1854	**1931.9**	**1931.9**	−2.65	27,394
	8	2000.3	2096.3	4.58	**1995.8**	2029.9	−0.23	1833	**1995.1**	**1995.1**	−0.26	26,944
	9	2052.7	2131.6	3.70	**2013.7**	**2048.4**	−1.94	1872	**2016.3**	**2016.3**	−1.80	27,735
	10	1977.9	2009.4	1.57	**1927.3**	**1950.2**	−2.63	1845	**1933.0**	**1933.0**	−2.32	27,420
Average		1408.3	1454.0	2.90	**1392.3**	**1404.6**	**−0.96**	675	**1394.4**	**1395.4**	**−0.82**	8492

Table 7
Comparison of ALNS with optimal results on small instances of Dethloff (2001) for the VRPSPD-H.

Instance	*N*	*h*	MIP		ALNS: best		ALNS: average		Time (s)
			Objective	Time (s)	Objective	Gap (%)	Objective	Gap (%)	
SCA3-0	5	2	317.74	0	**317.74**	0.00	**317.74**	0.00	0.32
	5	4	322.08	0	**322.08**	0.00	**322.08**	0.00	0.32
	10	1	467.77	29	**467.77**	0.00	**467.77**	0.00	0.69
	10	2	560.03	30	**560.03**	0.00	**560.03**	0.00	0.70
	15	0.67	664.25*	*15.3%*	**664.25**	0.00	**664.25**	0.00	1.57
	15	1.33	905.09*	*23.0%*	**905.09**	0.00	**905.09**	0.00	1.57
SCA8-1	5	2	384.12	0	**384.12**	0.00	**384.12**	0.00	0.32
	5	4	406.80	0	**406.80**	0.00	**406.80**	0.00	0.33
	10	1	598.17	84	**598.17**	0.00	**598.17**	0.00	0.70
	10	2	747.89	51	**747.89**	0.00	**747.89**	0.00	0.72
	15	0.67	726.14	17,901	**726.14**	0.00	**726.14**	0.00	1.53
	15	1.33	937.35	15,694	**937.35**	0.00	**937.35**	0.00	1.51
CON3-0	5	2	291.80	0	**291.80**	0.00	**291.80**	0.00	0.32
	5	4	319.82	0	**319.82**	0.00	**319.82**	0.00	0.33
	10	1	667.94	47	**667.94**	0.00	**667.94**	0.00	0.72
	10	2	893.87	65	**893.87**	0.00	**893.87**	0.00	0.74
	15	0.67	907.26*	*12.3%*	**907.26**	0.00	**907.26**	0.00	1.49
	15	1.33	1296.79*	*24.3%*	**1289.31**	−0.58	**1289.31**	−0.58	1.52
CON8-1	5	2	203.83	0	**203.83**	0.00	**203.83**	0.00	0.33
	5	4	218.31	0	**218.31**	0.00	**218.31**	0.00	0.33
	10	1	496.61	67	**496.61**	0.00	**496.61**	0.00	0.73
	10	2	694.64	61	**694.64**	0.00	**694.64**	0.00	0.74
	15	0.67	738.04*	*8.9%*	**738.04**	0.00	**738.04**	0.00	1.39
	15	1.33	1079.68*	*19.7%*	**1079.68**	0.00	**1079.68**	0.00	1.49
Average			618.58	6817.88	**618.27**	**−0.02**	**618.27**	**−0.02**	0.85

* indicates best integer solution for instances not solved to optimality.

parameters we generated new instances for the VRPSPD-H. First, we created 80 new instances based on the 40 instances for the VRPSPD of Dethloff (2001) by setting the handling cost parameters equal to $h = h_d = h_p = \{0.1, 0.5\}$. Additionally, we tested our algorithm on the same 40 instances of Dethloff (2001) for the VRPSPD and on instances for the TSPPD-H by Erdoğan et al. (2012).

As a starting point for the parameter tuning, we used the values reported by Ropke and Pisinger (2006b). The starting temperature of the simulated annealing procedure is set such that, in the first iteration, a solution with an objective up to 5% worse than the current solution is accepted with probability 0.5, and the cooling rate is set such that the temperature in the last iteration is 0.2 percent of the start temperature. The randomness parameter p, used to randomize some of the destroy operators, is initialized with value 3, and we remove $q \in [0.1|V_c|, 0.35|V_c|]$ customers in each iteration. The starting weights of the destroy and repair operators are set to 100, and the updating quantities are set to $\sigma_1 = 11$, $\sigma_2 = 3$ and $\sigma_3 = 4$. We find that setting the maximum number of iterations equal to 25,000 with weights being reset after every 100 iterations yielded good solutions compared to computation times. We sequentially changed the values of these parameters without finding significant improvements, which is in line with the conclusions of Ropke and Pisinger (2006b) and Veenstra et al. (2017), indicating that the algorithm is robust.

Since the handling cost component significantly increases the problem complexity, evaluation of solutions, which requires solving the handling sub-problem, is rather time consuming. Experiments showed that the destroy operators are all relatively fast compared to the repair operators. We forego a further extensive study on which operators to select since the adaptive mechanism increases the weights of well-performing operators.

6.2. Benchmarks on special cases and related problems

To test the quality of our metaheuristic when applied to special cases, we use it to solve various instances for the VRPSPD, the VRPMPD, and the TSPPD-H. Our obtained results are then compared to results from the literature to check the quality of our metaheuristic. We note that a direct comparison of calculation times reported for other methods especially designed for the special cases and our methods is not possible due to different computational environments. However, we see that the magnitude of the calculation times is similar even considering the generalizations we made in our ALNS to be able to solve the VRPSPD-H.

Table 8
Comparison of ALNS with optimal results on small instances of Nagy and Salhi (2005) for the VRPSPD-H.

Instance	*N*	*h*	MIP		ALNS: best		ALNS: average		Time (s)
			Objective	Time (s)	Objective	Gap (%)	Objective	Gap (%)	
1X	5	2	130	0	**130**	0.00	**130**	0.00	0.32
	5	4	140	0	**140**	0.00	**140**	0.00	0.32
	10	1	231	28	**231**	0.00	**231**	0.00	0.73
	10	2	263	16	**263**	0.00	**263**	0.00	0.71
	15	0.67	292.33	688	**292.33**	0.00	**292.33**	0.00	1.77
	15	1.33	328	764	**328**	0.00	**328**	0.00	1.63
1Y	5	2	128	0	**128**	0.00	**128**	0.00	0.32
	5	4	136	1	**136**	0.00	**136**	0.00	0.32
	10	1	242	45	**242**	0.00	**242**	0.00	0.72
	10	2	272	25	**272**	0.00	**272**	0.00	0.71
	15	0.67	304	4123	**304**	0.00	**304**	0.00	1.79
	15	1.33	336	1242	**336**	0.00	**336**	0.00	1.56
2X	5	2	153	0	**153**	0.00	**153**	0.00	0.32
	5	4	159	0	**159**	0.00	**159**	0.00	0.33
	10	1	226	16	**226**	0.00	**226**	0.00	0.70
	10	2	257	10	**257**	0.00	**257**	0.00	0.70
	15	0.67	295	1121	**295**	0.00	**295**	0.00	1.56
	15	1.33	336.67	1526	**336.67**	0.00	**336.67**	0.00	1.48
2Y	5	2	155	0	**155**	0.00	**155**	0.00	0.32
	5	4	159	0	**159**	0.00	**159**	0.00	0.33
	10	1	241	11	**241**	0.00	**241**	0.00	0.70
	10	2	269	8	**269**	0.00	**269**	0.00	0.69
	15	0.67	310	2066	**310**	0.00	**310**	0.00	1.60
	15	1.33	346.67	579	**346.67**	0.00	**346.67**	0.00	1.45
3X	5	2	153	0	**153**	0.00	**153**	0.00	0.33
	5	4	171	0	**171**	0.00	**171**	0.00	0.33
	10	1	235	45	**235**	0.00	**235**	0.00	0.72
	10	2	263	32	**263**	0.00	**263**	0.00	0.71
	15	0.67	321*	*3.4%*	**321**	0.00	321.67	0.21	1.82
	15	1.33	362.33	14,081	**362.33**	0.00	**362.33**	0.00	1.62
3Y	5	2	155	0	**155**	0.00	**155**	0.00	0.33
	5	4	175	0	**175**	0.00	**175**	0.00	0.33
	10	1	238	94	**238**	0.00	**238**	0.00	0.82
	10	2	264	33	**264**	0.00	**264**	0.00	0.71
	15	0.67	321*	*3.9%*	**321**	0.00	**321**	0.00	1.82
	15	1.33	362.33	10,505	**362.33**	0.00	**362.33**	0.00	1.55
Average			242.51	2229.42	**242.51**	**0.00**	**242.53**	**0.01**	0.89

* indicates best integer solution for instances not solved to optimality.

Table 9
Comparison of ALNS with optimal results on small instances of Dethloff (2001) for the VRPDPD-H.

Instance	*N*	*h*	MIP		ALNS: best		ALNS: average		Time (s)
			Objective	Time (s)	Objective	Gap (%)	Objective	Gap (%)	
SCA3-0	5	2	261.26	15	**261.26**	0.00	**261.26**	0.00	1.07
	5	4	265.54	19	**265.54**	0.00	**265.54**	0.00	1.07
	10	1	376.63	*38.4%*	**369.79**	−1.85	**370.14**	−1.75	5.56
	10	2	378.77	*31.7%*	**370.66**	−2.19	**371.57**	−1.94	4.57
SCA8-1	5	2	341.66	12	**341.66**	0.00	347.54	1.69	1.29
	5	4	356.14	17	**356.14**	0.00	363.69	2.08	1.25
	10	1	431.52	*36.7%*	**425.40**	−1.44	433.42	0.44	3.99
	10	2	434.96	7550	**434.96**	0.00	445.18	2.3	4.02
CON3-0	5	2	248.20	16	248.76	0.23	248.76	0.23	1.24
	5	4	248.20	14	**248.20**	0.00	250.79	1.03	1.25
	10	1	684.07	*70.9%*	**525.97**	−30.06	**528.93**	−29.33	5.66
	10	2	533.33	*72.5%*	**533.33**	0.00	551.10	3.22	7.40
CON8-1	5	2	189.47	8	**189.47**	0.00	191.66	1.15	1.21
	5	4	189.47	6	**189.47**	0.00	189.97	0.27	1.21
	10	1	271.49	*22.1%*	**271.49**	0.00	271.52	0.01	3.39
	10	2	271.49	*10.2%*	**271.49**	0.00	276.04	1.65	3.40
Average			197.75	10242.96	**195.37**	**−0.94**	**196.09**	**−0.50**	2.57

* indicates best integer solution for instances not solved to optimality.

6.2.1. *Performance on the VRPSPD*

Two well-known sets of benchmark instances for the VRPSPD without handling costs are solved. The 40 instances of Dethloff (2001) contain 50 customers each, divided into four different configurations of 10 instances. The capacity is such that the minimum number of vehicles required is either 3 or 8, and customers are either scattered uniformly over a square area or a fraction of the customers is clustered to resemble a more urban area. Furthermore, 14 instances adopted from Salhi and Nagy (1999) are solved. These instances range in size from 50 to 199 customers. An important remark on these instances is that there is some confusion in the literature on how to adapt them

Table 10
Comparison of ALNS with optimal results on small instances of Nagy and Salhi (2005) for the VRPDPD-H.

Instance	N	h	MIP		ALNS: best		ALNS: average		Time (s)
			Objective	Time (s)	Objective	Gap (%)	Objective	Gap (%)	
1X	5	2	129	10	**129**	0.00	**129**	0.00	1.18
	5	4	133	8	**133**	0.00	**133**	0.00	1.03
	10	1	248	*25.6%*	**233**	-6.44	**233**	-6.44	3.75
	10	2	252	*33.5%*	**251**	-0.40	252.6	0.24	3.72
1Y	5	2	128	7	**128**	0.00	**128**	0.00	1.04
	5	4	133	9	**133**	0.00	**133**	0.00	0.99
	10	1	255	*24.7%*	**247**	-3.24	**247**	-3.24	3.92
	10	2	271	*45.3%*	**256**	-5.86	**256.8**	-5.53	3.94
2X	5	2	138	5	**138**	0.00	141.8	2.68	1.12
	5	4	138	5	**138**	0.00	139.7	1.22	1.10
	10	1	223	*11.6%*	**222**	-0.45	**222**	-0.45	3.37
	10	2	242	*25.7%*	**242**	0.00	243.4	0.58	3.48
2Y	5	2	138	7	**138**	0.00	143.2	3.63	1.11
	5	4	138	4	**138**	0.00	139.4	1.00	1.10
	10	1	239	*2.4%*	**239**	0.00	239.6	0.25	3.37
	10	2	243	8055	**243**	0.00	243.2	0.08	5.87
3X	5	2	159	39	**159**	0.00	**159**	0.00	1.30
	5	4	177	31	179	1.12	179.3	1.28	1.28
	10	1	246	*23.8%*	**238**	-3.36	**238**	-3.36	3.62
	10	2	265	*30.7%*	**258**	-2.71	**258**	-2.71	3.62
3Y	5	2	161	31	**161**	0.00	**161**	0.00	1.31
	5	4	177	18	179	1.12	179	1.12	1.22
	10	1	243	*27.7%*	**243**	0.00	243.1	0.04	5.61
	10	2	270	*38.9%*	**264**	-2.27	**264**	-2.27	3.72
Average			342.64	9928.56	**331.47**	**−2.21**	**335.44**	**−1.19**	2.97

* indicates best integer solution for instances not solved to optimality.

to the VRPSPD, since they originally come from the classical VRP. Some common divergencies in the literature are if values of pickups and deliveries should be rounded and how to invert pickups and deliveries to generate the Y version of the instances (see Avci and Topaloglu (2015); Gajpal and Abad (2009); Kalayci and Kaya (2016); Subramanian et al. (2013)). Due to these divergences, solutions for these instances are being incorrectly compared in the literature. Since most studies do not detail how they adapted the instances, it is hard to know which studies used the same procedures we do. We have then contacted the authors of the original study that presented the instance set to clarify the original methodology. The formula to generate the pickup and deliveries is presented in the original study Salhi and Nagy (1999). Distances, pickups and deliveries should be rounded to the nearest integer. To generate the Y dataset, all customers with an even index (customers 2, 4, 6, etc.) should have their pickup and delivery values inverted. We note that all best known solutions (BKS) for the instances of Dethloff (2001) are proven to be optimal in Subramanian et al. (2013).

Table 1 reports the results on the 40 instances of Dethloff (2001). We solved each instance 10 times and report the ALNS best found solution as well as the average objective value over these 10 runs. Best known solutions are reported from Subramanian et al. (2013). Since we also provide the average solution value and not only the best, and considering that the variation between these two values is very small, we follow the common method from literature reporting the average time of all executions. See, e.g., the benchmarks of Gajpal and Abad (2009) and Polat (2017). Gaps are calculated as $(x - x^*)/x$, where x is the value of the objective function for the ALNS solution and x^* is the value of the objective function for the BKS. The ALNS finds 26 out of 40 best known solutions in at least one of the 10 runs, and on average over all 40 instances the gap to the BKS is only 0.21%. The results on the benchmark instances by Salhi and Nagy (1999) are reported in Table 2. We compare our results with those reported in Polat (2017) since they are the ones whose procedure appears more similar to ours. Of the 14 instances, we find 11 BKS and our gap is on average −0.65%, improving their solutions.

6.2.2. Performance on the VRPDPD

The same instance sets used for the VRPSPD were tested for the VRPDPD. The problem was solved by the ALNS in such a way that we start the solution from the VRPSPD solution, then we create the dummy customers and run the ALNS again using the same parameters to search for improvements. Tables 3 and 4 show the results for both instance sets. BKS are reported from Polat (2017). Times are reported as the sum of the time to solve the VRPSPD and to run ALNS again with the dummy customers. On the first set, our algorithm finds 22 out of 40 optimal solutions, and on average the gap with the BKS is 0.24%. The VRPDPD best solution found in this dataset is better than the VRPSPD best solution (Table 1) in nine instances. Our ALNS was able to improve the VRPSPD solution in six of these instances, finding the BKS for the VRPDPD in three of them. On the second set, the ALNS improves the best known solution in 12 out of 14 instances, and it has an average gap to the BKS of −0.19%. The VRPDPD best solution found is better than the VRPSPD best solution in 10 instances, which attests the capacity of our method of improving solutions for the divisible version of the problem.

6.2.3. Performance on the VRPMPD

We test the performance of the ALNS on instances for the VRPMPD derived from the CMT set of Salhi and Nagy (1999). This data set consists of 21 instances. For each VRPSPD instance of type X, every 2nd, 4th and 10th customer is set to be a pickup only customer while the others are set to be a delivery only. The new instances are labeled as H (half), Q (quarter) and T (tenth), respectively. This dataset is solved by several studies (Crispim and Brandão (2005); Gajpal and Abad (2009); Nagy and Salhi (2005); Ropke and Pisinger (2006b); Wassan et al. (2008)). In Table 5, we show the results with BKS reported from Ropke and Pisinger (2006b). The ALNS improves BKS in 15 out of 21 instances,

Table 11
Comparison of ALNS with optimal results on small instances of Nagy and Salhi (2005) for the VRPMPD-H.

Instance	N	h	MIP		ALNS: best		ALNS: average		Time (s)
			Objective	Time (s)	Objective	Gap (%)	Objective	Gap (%)	
1H	5	2	120	0	**120**	0.00	**120**	0.00	0.33
	5	4	120	0	**120**	0.00	**120**	0.00	0.33
	10	1	220	17	**220**	0.00	223	1.35	0.92
	10	2	220	17	**220**	0.00	**220**	0.00	0.87
	15	0.67	279.33	1980	**279.33**	0.00	288.4	3.14	2.97
	15	1.33	295.33	1294	**295.33**	0.00	297.47	0.72	2.90
	20	0.5	308*	*3.8%*	**308**	0.00	**308**	0.00	3.29
	20	1	325*	*14.9%*	**316**	-2.85	**316**	-2.85	3.19
1Q	5	2	106	0	**106**	0.00	**106**	0.00	0.33
	5	4	106	0	**106**	0.00	**106**	0.00	0.33
	10	1	190	6	192	1.04	192	1.04	0.91
	10	2	190	5	192	1.04	192	1.04	0.88
	15	0.67	249	97	**249**	0.00	**249**	0.00	1.75
	15	1.33	249	64	**249**	0.00	**249**	0.00	1.76
	20	0.5	309.5*	*4.0%*	**308**	-0.49	**308.2**	-0.42	3.10
	20	1	313*	*5.5%*	**313**	0.00	316.9	1.23	3.02
1T	5	2	106	0	**106**	0.00	**106**	0.00	0.33
	5	4	106	0	**106**	0.00	**106**	0.00	0.33
	10	1	175	0	**175**	0.00	**175**	0.00	0.90
	10	2	180	1	**180**	0.00	**180**	0.00	0.87
	15	0.67	247	8	**247**	0.00	**247**	0.00	1.63
	15	1.33	253	55	**253**	0.00	**253**	0.00	1.64
	20	0.5	302.5	380	**302.5**	0.00	303.55	0.35	2.45
	20	1	305	1050	**305**	0.00	**305**	0.00	2.40
2H	5	2	116	0	**116**	0.00	**116**	0.00	0.33
	5	4	116	0	**116**	0.00	**116**	0.00	0.33
	10	1	198	16	**198**	0.00	**198**	0.00	0.92
	10	2	202	7	**202**	0.00	**202**	0.00	0.90
	15	0.67	256	551	**256**	0.00	**256**	0.00	1.76
	15	1.33	256	622	**256**	0.00	**256**	0.00	1.74
	20	0.5	337*	*3.8%*	**337**	0.00	**337**	0.00	3.06
	20	1	344*	*9.5%*	**344**	0.00	**344**	0.00	2.98
2Q	5	2	103	0	**103**	0.00	**103**	0.00	0.33
	5	4	103	0	**103**	0.00	**103**	0.00	0.33
	10	1	178	1	**178**	0.00	**178**	0.00	0.70
	10	2	178	1	**178**	0.00	**178**	0.00	0.70
	15	0.67	229	53	**229**	0.00	**229**	0.00	1.43
	15	1.33	229	13	**229**	0.00	**229**	0.00	1.44
	20	0.5	312	4880	**312**	0.00	**312**	0.00	2.34
	20	1	312	20,604	**312**	0.00	**312**	0.00	2.24
2T	5	2	103	0	**103**	0.00	**103**	0.00	0.33
	5	4	103	0	**103**	0.00	**103**	0.00	0.33
	10	1	184	1	**184**	0.00	**184**	0.00	0.71
	10	2	184	2	**184**	0.00	**184**	0.00	0.71
	15	0.67	238	8	**238**	0.00	**238**	0.00	1.30
	15	1.33	238	5	**238**	0.00	**238**	0.00	1.31
	20	0.5	314	5900	**314**	0.00	**314**	0.00	2.16
	20	1	315	14,859	**315**	0.00	**315**	0.00	2.12
Average			217.14	3793.69	**217.00**	**−0.03**	**217.40**	**0.12**	1.42

* indicates best integer solution for instances not solved to optimality.

and it has an average gap to the BKS of –0.65%, improving the best results from the literature.

6.2.4. Performance on the TSPPD-H

Another benchmark is performed on the instances proposed by Erdoğan et al. (2012). The authors adapted 10 instances containing 200 customers proposed by Gendreau et al. (1999) to consider simultaneous pickup and delivery. Smaller instances were created by considering only the first 20, 40, 60, 80, 100, 120, 140, 160, 180 and 200 customers, respectively. The handling cost parameters are chosen such that $h = h_d = h_p$, where the product $hN = 20$ is kept constant. For further details concerning the exact configuration of these instances we refer to Erdoğan et al. (2012). In our results, we left out the instances with $|V_c| = \{80, 100\}$. For these instances, the results reported in Erdoğan et al. (2012) have objectives smaller or close to those of the $N = 60$ instances, while adding 20 and 40 customers to the same set of 60 customers, respectively. All other instances display a logical growth in objective values when the number of customers increases.

To be able to solve the instances for the TSPPD-H we adapt our metaheuristic so that it will only create a single route. For instance, operators which by design require multiple routes (e.g., *cross route removal* and *3-opt*) were excluded from the metaheuristic, and feasibility issues which normally lead to construction of new routes have been solved. We solve the handling sub-problem with both the exact DP and the approximation as proposed by Erdoğan et al. (2012) and compare these outcomes with their best overall and best heuristic solutions. Inspired by Erdoğan et al. (2012), to prevent compromising the time complexity of our metaheuristic, neighborhood searches are evaluated with the heuristic handling policy rather than the DP. When a change in the solution is made, the costs are computed with the DP. Assessing changes with the heuristic handling policy rather than the DP resulted in faster computation times without loss of solution quality. We present the results in Table 6. Results

Table 12
Comparison of handling policies on instances of Dethloff (2001) for the VRPSPD-H.

Instance	N	K	MIP	Policies	Heuristic policy			Myopic policy 1			Myopic policy 2		
			UB	BKS	Best (%)	Avg. (%)	Time (s)	Best (%)	Avg. (%)	Time (s)	Best (%)	Avg. (%)	Time (s)
SCA3-0	50	5	1579.85	1426.19	0.00	**0.05**	17.4	0.90	0.90	11.3	10.83	11.71	12.4
SCA3-1	50	5	1613.04	1476.19	0.00	**0.12**	17.6	1.99	2.06	10.9	11.67	12.17	12.3
SCA3-2	50	5	1659.41	1535.05	0.00	**0.17**	17.7	0.46	0.49	10.9	12.20	12.57	12.5
SCA3-3	50	5	1697.40	1547.17	0.00	**0.37**	17.5	0.50	1.82	10.7	11.72	12.17	12.1
SCA3-4	50	5	1864.25	1724.52	0.00	0.32	17.6	0.02	**0.17**	11.3	13.23	14.06	12.4
SCA3-5	50	5	1677.63	1467.84	0.00	**0.03**	18.4	2.64	2.79	11.3	11.19	11.53	12.6
SCA3-6	50	5	1554.32	1421.40	0.00	**0.44**	18.0	1.67	1.86	11.2	11.01	11.32	12.8
SCA3-7	50	5	1695.64	1536.16	0.00	**0.41**	18.0	1.31	1.64	11.1	12.46	12.76	12.3
SCA3-8	50	5	1608.04	1550.41	0.00	**0.30**	18.1	1.23	1.38	11.4	10.26	10.36	12.4
SCA3-9	50	5	1602.00	1467.83	0.00	**0.00**	16.7	1.42	1.71	10.7	10.58	11.10	11.9
SCA8-0	50	12	1476.76	1369.93	0.00	0.25	10.5	0.00	**0.10**	8.1	3.37	3.37	8.5
SCA8-1	50	12	1508.19	1458.51	0.00	**0.39**	11.3	0.72	0.78	8.3	2.73	2.78	8.8
SCA8-2	50	12	1617.55	1496.52	0.43	0.62	10.6	0.00	**0.09**	8.0	3.97	4.07	8.4
SCA8-3	50	12	1517.64	1455.43	0.00	**0.18**	10.2	0.47	0.60	8.0	2.65	2.71	8.6
SCA8-4	50	12	1647.93	1621.62	0.00	**0.10**	10.4	0.10	**0.10**	8.1	2.25	2.27	8.4
SCA8-5	50	12	1531.89	1456.27	0.11	0.23	10.9	0.00	**0.01**	8.7	2.29	2.42	9.1
SCA8-6	50	12	1444.14	1378.91	0.00	**0.07**	10.6	0.30	0.30	8.5	2.25	2.28	9.0
SCA8-7	50	13	1535.74	1470.16	0.00	**0.02**	10.6	0.28	0.28	8.5	3.02	3.03	8.5
SCA8-8	50	12	1562.68	1531.18	0.00	**0.09**	11.0	0.31	0.42	8.6	2.67	2.77	8.9
SCA8-9	50	12	1540.27	1467.44	0.20	0.32	10.2	0.00	**0.02**	8.1	2.47	2.59	8.5
CON3-0	50	5	1547.44	1398.60	0.00	**0.47**	19.3	1.00	1.42	12.3	11.52	11.75	13.5
CON3-1	50	5	1630.45	1478.77	0.00	**0.57**	20.0	0.84	1.29	12.9	11.57	12.17	13.5
CON3-2	50	5	1461.92	1293.78	0.00	0.43	19.7	0.36	**0.36**	12.5	14.47	15.49	13.6
CON3-3	50	5	1544.62	1426.86	0.31	0.82	19.2	0.00	**0.00**	11.8	14.44	14.68	13.1
CON3-4	50	5	1635.52	1463.11	0.00	**0.66**	19.9	1.23	1.65	12.4	12.67	13.68	13.5
CON3-5	50	5	1455.55	1303.25	0.00	**0.68**	19.4	1.71	1.77	12.0	11.17	11.68	13.4
CON3-6	50	5	1245.62	1159.15	0.27	0.96	20.0	0.00	**0.41**	13.0	8.75	9.16	14.0
CON3-7	50	5	1548.06	1381.00	0.00	**0.56**	18.4	0.53	0.70	12.1	12.60	12.88	13.0
CON3-8	50	5	1510.60	1369.02	0.00	**0.25**	19.4	1.18	1.74	12.3	13.76	14.33	13.2
CON3-9	50	5	1377.61	1261.29	0.00	**0.14**	18.7	0.80	0.81	11.8	10.63	11.29	13.3
CON8-0	50	12	1265.96	1242.72	0.00	**0.30**	11.1	0.41	0.43	8.5	1.97	2.03	9.3
CON8-1	50	12	1295.97	1222.29	0.12	0.27	12.6	0.00	**0.02**	8.7	1.91	2.01	9.5
CON8-2	50	12	1141.49	1107.11	0.04	0.15	12.5	0.00	**0.02**	9.3	3.92	4.05	10.1
CON8-3	50	13	1262.77	1220.64	0.00	0.22	11.9	0.00	**0.00**	8.8	1.85	1.90	9.1
CON8-4	50	12	1362.12	1232.67	0.04	0.09	11.6	0.00	**0.00**	8.8	2.94	3.00	9.5
CON8-5	50	12	1175.41	1127.85	0.00	**0.03**	10.9	0.23	0.23	8.6	3.13	3.22	9.0
CON8-6	50	12	1103.76	1013.72	0.00	**0.02**	11.9	0.17	0.24	9.2	0.58	0.91	9.7
CON8-7	50	12	1301.36	1223.76	0.07	0.31	11.2	0.00	**0.00**	8.1	3.46	3.46	9.2
CON8-8	50	12	1267.14	1202.92	0.02	0.03	12.2	0.00	**0.00**	8.9	2.80	2.90	9.6
CON8-9	50	12	1195.55	1157.10	0.01	0.16	11.2	0.00	**0.04**	8.8	1.52	1.65	9.9
Average			1481.58	1378.61	0.04	**0.29**	14.9	0.57	0.72	10.1	7.21	7.51	11.0

are reported for the ALNS with both heuristic and DP handling policies. Solutions reported in Erdoğan et al. (2012) columns are the BKS found by their algorithm when using the DP and the heuristic handling policies. Their gap is calculated between the heuristic and DP column, while ours are calculated between the best solution found and the DP column of the benchmark algorithm. We show that although the DP policy leads to better results in certain instances, the time it takes is too high in larger instances.

Our ALNS with heuristic policy finds or improves 69 out of 80 best known solutions, which were found with the DP in Erdoğan et al. (2012), and the average gap is −0.96%. It improves the heuristic policy of the authors on all 80 instances. The average time needed to compute these solutions is competitive with the exact evaluation in Erdoğan et al. (2012). Our ALNS with DP policy finds or improves best known solutions on 68 out of 80 instances with an average gap to the BKS of −0.82%. The average solution with the DP policy is better than with the heuristic policy, but calculation times are on average 13 times higher.

6.3. Benchmark optimal solutions

Since we are the first to model and solve the VRPSPD-H, VRPDPD-H, and VRPMPD-H, no solutions in the literature exist for these problems for us to compare with. We have implemented the model of Section 2.1 to the VRPSPD-H, and that of Section 3.2 to the other two problems. We have then solved these models with CPLEX and compare our metaheuristic with the optimal solutions on small instances. We point that the ALNS was implemented in such a way to evaluate handling costs using the same method as in the model, thus also giving an upper bound for the VRPDPD-H. We have generated new instances for these three problems. We select the first N customers of four VRPSPD instances of Dethloff (2001) and six VRPSPD and another six VRPMPD instances of Salhi and Nagy (1999). The VRPSPD instances are used to solve the VRPSPD-H and the VRPDPD-H, while the VRPMPD instances are used to the VRPMPD-H. We start with $N = 5$ customers. If the MIP is solved to optimality within six hours in all instances of each problem, we generate another set with 5 more customers, and so on. This way we observe the maximum instance size that each problem can be solved with CPLEX. For all instances, we set the handling cost parameters to $\frac{10}{N}$ and $\frac{20}{N}$ and we limit the number of available vehicles to 4. The results are reported in Tables 7–11. We solved each instance with CPLEX on a single thread. Instances not solved to optimality are indicated with an asterisk, and the gap to the best lower bound found is reported as a percentage in the column *Time*. We compare the results with 10 runs of our ALNS metaheuristic with the DP handling policy. CPLEX fails to prove optimality within the time limit in instances with 15 customers in both sets of the VRPSPD-H, 10 customers in both sets

Table 13
Comparison of handling policies for the VRPDPD-H.

Instance	N	K	MIP	Policies	Heuristic policy			Myopic policy 1			Myopic policy 2		
			UB	BKS	Best (%)	Avg. (%)	Time (s)	Best (%)	Avg. (%)	Time (s)	Best (%)	Avg. (%)	Time (s)
SCA3-0	100	5	2035.14	925.29	0.25	**0.36**	98.4	0.00	1.19	58.0	0.80	1.98	63.6
SCA3-1	100	5	2018.02	1063.22	0.00	**0.19**	104.1	0.00	0.78	60.4	0.36	0.86	63.6
SCA3-2	100	5	-	1023.15	0.00	1.79	103.2	0.09	**1.50**	59.1	1.41	2.71	63.4
SCA3-3	100	5	-	1050.77	0.00	0.22	108.2	0.00	**0.12**	61.8	0.49	0.72	66.6
SCA3-4	100	5	-	1059.66	0.13	1.54	95.8	0.00	**1.31**	55.5	1.97	2.40	61.2
SCA3-5	100	5	-	967.86	0.00	2.24	109.5	0.46	3.12	62.8	0.43	**1.93**	67.1
SCA3-6	100	5	-	1013.14	0.00	**0.11**	111.4	0.07	0.13	62.9	0.48	0.52	67.9
SCA3-7	100	5	-	1008.12	0.00	0.45	109.3	0.00	**0.16**	59.3	0.44	0.66	63.9
SCA3-8	100	5	2546.33	1083.36	0.00	**0.87**	100.5	0.70	1.28	58.7	1.25	1.85	61.2
SCA3-9	100	5	2312.22	1016.53	0.00	**0.00**	96.9	0.00	0.39	56.9	0.00	**0.00**	59.5
SCA8-0	100	12	-	1125.04	0.47	0.79	59.3	0.00	**0.75**	41.9	0.70	1.28	44.0
SCA8-1	100	12	-	1264.21	0.00	**0.89**	60.6	0.73	1.13	40.8	1.03	1.25	42.6
SCA8-2	100	12	-	1237.98	0.00	**0.43**	60.2	0.16	0.80	42.1	0.54	0.72	44.1
SCA8-3	100	12	-	1209.17	0.00	0.48	61.8	0.00	**0.32**	43.2	0.31	0.69	45.6
SCA8-4	100	12	-	1278.29	0.00	0.84	57.2	0.05	0.35	40.7	0.12	**0.27**	42.3
SCA8-5	100	12	-	1236.41	0.00	**0.75**	61.7	0.05	1.61	43.6	0.05	0.76	45.9
SCA8-6	100	12	-	1193.44	0.00	**0.91**	59.2	0.00	0.98	42.3	0.51	1.15	44.4
SCA8-7	100	13	-	1242.84	0.00	0.91	59.4	0.01	0.92	41.9	0.15	**0.80**	44.3
SCA8-8	100	12	-	1294.43	0.00	**0.27**	61.2	0.55	0.83	43.2	0.36	0.77	44.4
SCA8-9	100	12	-	1258.90	1.26	1.48	57.4	1.10	1.38	40.7	0.00	**1.33**	42.8
CON3-0	100	5	-	939.84	0.00	**0.15**	112.5	0.26	0.61	63.1	1.31	1.67	68.0
CON3-1	100	5	1918.20	834.27	0.00	**0.04**	122.6	0.00	0.10	68.6	0.21	0.22	72.3
CON3-2	100	5	2543.41	760.42	0.50	1.05	112.7	0.00	**0.96**	64.8	1.94	2.80	70.7
CON3-3	100	5	2400.12	886.05	0.00	**0.07**	110.8	0.19	0.23	61.5	1.06	1.06	67.1
CON3-4	100	5	1648.01	855.97	0.00	**0.64**	117.5	0.52	1.00	67.0	1.22	1.88	69.3
CON3-5	100	5	1224.09	818.45	0.48	**0.68**	110.0	0.00	0.78	60.4	0.54	1.33	64.6
CON3-6	100	5	-	730.74	0.00	0.52	113.1	0.04	**0.30**	63.1	1.03	1.40	67.9
CON3-7	100	5	1560.31	885.60	0.00	**0.35**	113.2	0.88	0.98	62.0	1.59	2.03	66.0
CON3-8	100	5	-	751.93	0.00	**0.14**	98.7	0.00	0.68	60.4	0.06	1.60	65.5
CON3-9	100	5	1393.27	853.88	0.00	**0.08**	113.5	0.30	0.37	61.4	1.46	1.54	66.6
CON8-0	100	12	-	1057.34	0.00	**0.80**	60.9	0.41	1.23	43.3	0.69	1.60	45.7
CON8-1	100	12	-	913.18	0.49	**1.34**	63.4	0.00	1.37	43.2	0.69	1.91	45.6
CON8-2	100	12	-	841.25	0.15	**0.46**	64.4	0.00	0.72	45.8	0.59	1.17	48.7
CON8-3	100	13	-	991.62	0.00	**0.56**	60.5	0.20	0.81	43.2	2.28	2.56	45.7
CON8-4	100	12	-	957.90	0.00	**1.05**	61.2	0.50	1.12	44.3	0.46	1.15	46.5
CON8-5	100	12	-	920.30	0.37	**1.58**	58.4	0.00	1.89	43.8	1.30	2.30	44.0
CON8-6	100	12	-	816.50	0.00	**0.94**	63.2	0.56	1.27	46.2	0.72	1.26	47.5
CON8-7	100	12	-	1007.50	0.00	1.16	60.5	0.29	1.28	43.0	0.50	**0.87**	45.3
CON8-8	100	12	-	920.99	0.04	0.27	62.6	0.00	**0.11**	44.7	0.09	0.22	47.1
CON8-9	100	12	-	959.41	0.32	1.23	60.5	0.00	**1.09**	42.7	0.52	1.45	48.3
Average			-	1006.37	0.11	**0.71**	84.4	0.20	0.90	52.2	0.74	1.32	55.5

of the VRPDPD-H, which is equivalent to 20 customers when considering the doubled customers, and 20 customers in the set of the VRPMDP-H. Among all 148 instances tested, CPLEX proved optimality for 116. Our ALNS finds at least the best bound given by CPLEX in all VRPSPD-H instances, in all except three VRPDPD-H instances, and in all except two VRPMPD-H instances. Considering all instances, the average time our metaheuristic needs is 1.57 s, compared to 5595.66 s for CPLEX. These instances and results can be shared upon request.

6.4. Comparison of handling policies

We now compare the performance of the heuristic handling policy compared to myopic policies 1 and 2, respectively, when embedded in our ALNS structure. We have created 40 new VRPSPD-H and VRPDPD-H instances based on the VRPSPD instances of Dethloff (2001) and 21 new instances based on Nagy and Salhi (2005) for the VRPMDP-H. Our additions to the original instances are the handling cost parameter, which is set to 0.2 for all instances. This resulted in a distinct trade-off between routing and handling costs. Furthermore, we have imposed a limit on the number of available vehicles K, where $K = \max\{0.7\sum_{i\in V_c}(d_i + p_i), 1.4\sum_{i\in V_c} d_i, 1.4\sum_{i\in V_c} p_i\}/Q$, so that the number of vehicles is high enough to guarantee feasibility but not too high so that solutions result in the unrealistic construction of many short routes to decrease handling as much as possible. The results of the experiments are given in Tables 12–14, where we report the best objective value found by any policy in column *BKS*, the best and average gaps to the BKS, the computation times over 10 runs for all three policies, as well as the best upper bound found by the MIP after six hours whenever any is found. Since none of the instances were solved by the MIP within the time limit we omitted running times from the tables.

We see that myopic policy 1 outperforms myopic policy 2 on all VRPSPD-H instances in similar computation times. On the VRPSPD-H, our ALNS with heuristic policy has the best average performance on 25 instances, myopic policy 1 on 16 instances, and myopic policy 2 and MIP on none of the instances. Although the heuristic policy performed 0.43% better than the myopic policy 1, the average computation time of the latter is around 32% lower. Most of the improvements on myopic policy 1 are obtained when K is higher. This is an intuitive result as the importance of handling decisions declines when routes get shorter. Hence, there exist scenarios in which myopic policy 1 is competitive with the heuristic policy for the VRPSPD-H. We characterize such scenarios in Section 6.5.

We show important highlights on results for the VRPDPD-H. On most instances the MIP could not find a feasible solution to the problem within six hours. The heuristic policy performed best on 25 instances, while myopic policy 1 was the best one on 10 in-

Table 14
Comparison of handling policies for the VRPMPD-H.

Instance	N	K	MIP	Policies	Heuristic policy			Myopic policy 1			Myopic policy 2		
			UB	BKS	Best (%)	Avg. (%)	Time (s)	Best (%)	Avg. (%)	Time (s)	Best (%)	Avg. (%)	Time (s)
1H	50	4	631.6	528.4	0.00	**0.11**	15.9	0.94	0.94	9.3	1.82	1.82	9.4
1Q	50	6	550.8	535.4	0.00	0.11	14.3	0.04	**0.07**	8.9	0.26	0.32	9.2
1T	50	7	562.0	537.8	0.00	**0.06**	13.2	0.85	0.85	8.3	0.92	0.92	8.4
2H	75	8	-	719.6	0.00	**0.00**	31.2	0.30	0.30	21.1	0.91	0.91	22.0
2Q	75	10	-	771.6	0.00	**0.87**	32.0	1.20	1.62	22.1	2.50	2.77	22.6
2T	75	13	-	798.2	0.22	**1.11**	27.8	0.99	1.48	20.8	0.00	1.20	20.6
3H	100	6	2228.0	789.8	0.00	**0.45**	82.4	0.03	**0.45**	48.0	0.33	0.69	48.8
3Q	100	8	-	810.2	0.00	**0.22**	79.7	0.42	0.95	47.3	1.46	1.96	49.2
3T	100	10	-	812.8	0.00	**0.96**	70.6	0.05	1.06	46.1	1.17	1.56	46.6
4H	150	9	-	959.6	0.00	**0.53**	228.4	0.06	0.83	132.8	0.19	1.13	138.9
4Q	150	12	-	994.0	0.00	1.30	220.9	0.14	**1.25**	122.0	0.88	1.59	137.0
4T	150	15	-	1027.8	0.58	1.14	237.9	0.00	**1.04**	149.9	0.52	1.32	151.7
5H	199	12	-	1129.8	0.00	**1.15**	459.5	0.77	1.45	271.3	1.05	2.08	285.9
5Q	199	17	-	1237.4	0.11	0.81	485.1	0.00	**0.70**	293.4	0.34	1.47	301.7
5T	199	21	-	1278.2	0.79	**1.45**	472.7	0.00	1.47	299.5	0.64	1.72	300.9
11H	120	6	-	949.2	0.00	0.96	187.6	0.15	**0.72**	105.5	0.04	0.88	113.2
11Q	120	8	-	1004.4	0.00	**0.52**	178.9	0.16	0.53	101.7	0.12	**0.52**	104.3
11T	120	9	-	1032.4	3.06	5.62	167.3	0.00	5.97	97.4	0.33	**3.93**	99.7
12H	100	7	-	748.0	0.00	0.16	69.0	0.00	**0.12**	42.0	0.27	0.45	42.2
12Q	100	10	960.0	815.0	0.24	0.75	70.6	0.00	**0.65**	45.8	0.61	0.89	41.9
12T	100	12	870.0	809.0	0.00	**1.34**	72.2	0.49	1.62	46.4	0.86	1.59	47.7
Average			-	870.9	0.24	**0.94**	153.2	0.31	1.15	92.4	0.72	1.42	95.3

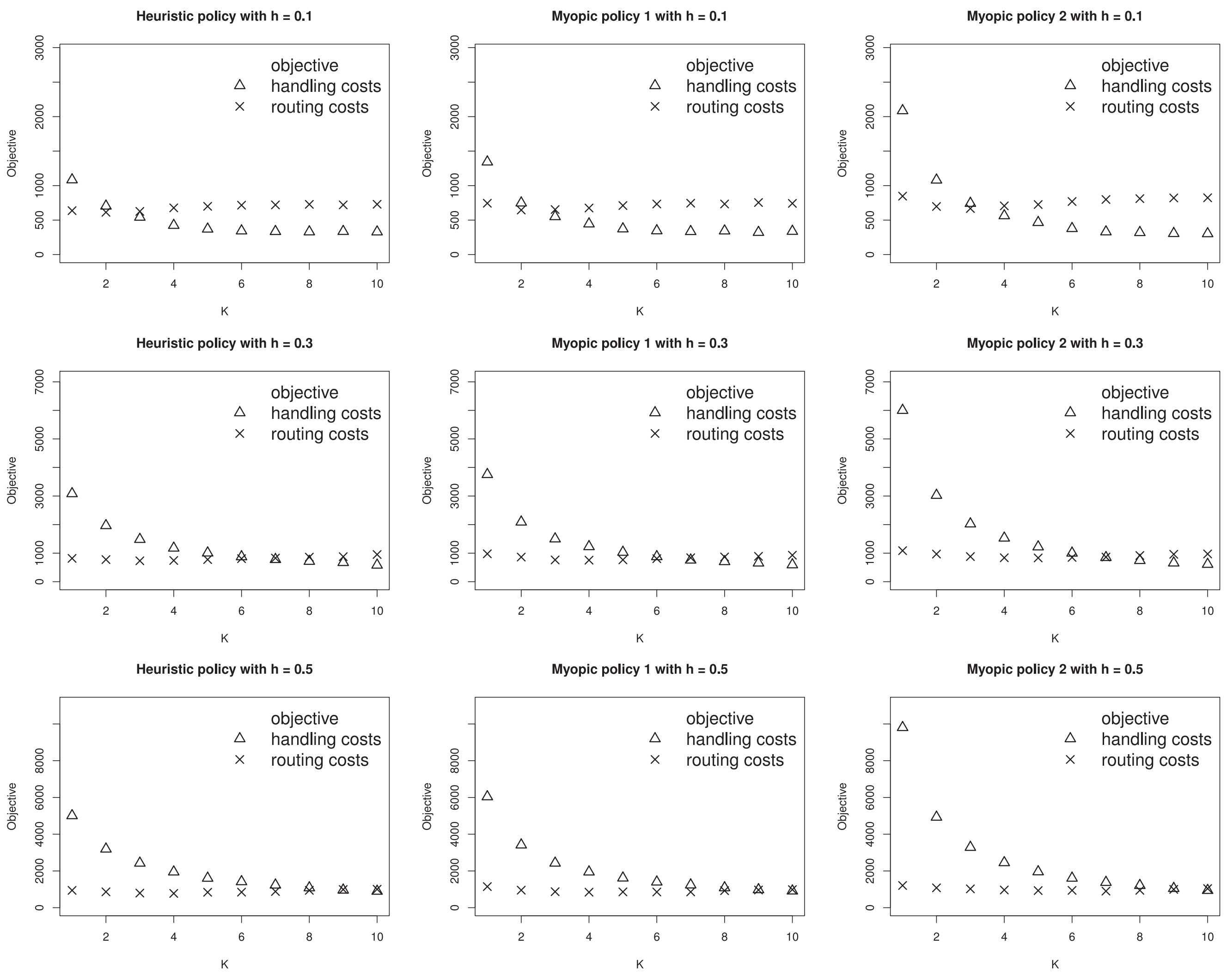


Fig. 2. Different cost components for myopic policies 1 and 2 and the heuristic policy for $K = \{1, \ldots, 10\}$ and $h = \{0.1, 0.3, 0.5\}$.

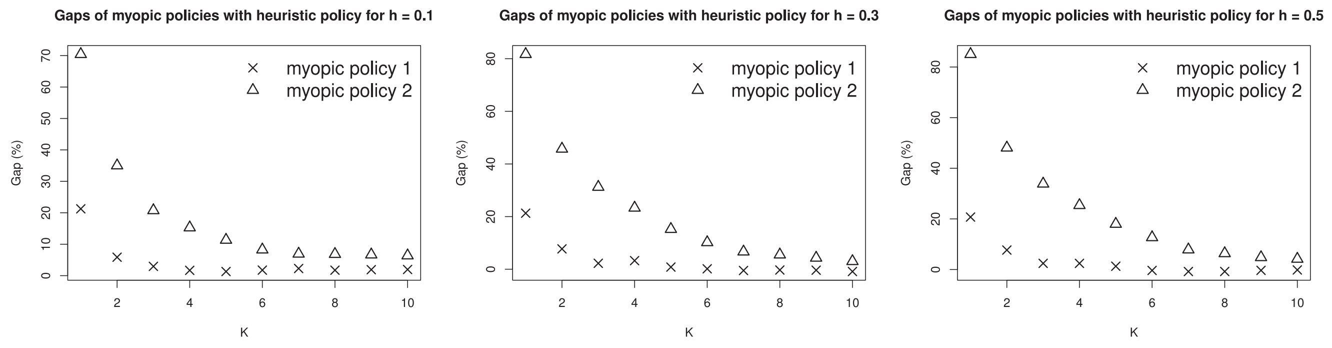


Fig. 3. Objective gaps of myopic policies 1 and 2 compared to the heuristic policy for $h = \{0.1, 0.3, 0.5\}$.

stances, and myopic policy 2 on 6 instances. Differences on average performance of all methods were less significant with the heuristic policy leading by only 0.19% of myopic policy 1, which performed only 0.42% better than myopic policy 2. Finally, average computation times for both myopic policies are more than 30% lower than for the heuristic policy.

On the VRPMDP-H, the observations are similar to the VRPDPD-H. The heuristic policy performed better on 12 instances, the myopic policy 1 on 8 instances, and the myopic policy 2 on only 1 instance. The average performance distances are quite similar with heuristic policy with a lead of 0.21% from myopic policy 1, while this is better than myopic policy 2 by 0.27%. Differences in computation times are also proportionally similar for the three policies.

6.5. Varying number of vehicles

We observed that the metaheuristic tends to create many short routes to decrease handling costs which may result in unrealistic scenarios. This section focuses on how the number of available vehicles impacts the trade-off between routing and handling costs. For these computational experiments, we iteratively increase the maximum number of allowed routes on various instances where we ignore capacity constraints. We solve instances with $N = 50$, $h = \{0.1, 0.3, 0.5\}$ and increase the maximum number of allowed routes K from 1 to 10 with both myopic policies and the heuristic policy for the VRPSPD-H. We explicitly distinguish between the routing and handling cost components. The results are presented in Fig. 2. The reported values are averages over 10 runs of our ALNS metaheuristic, for all configurations and handling policies. We see that routing costs stay fairly constant in all scenarios when increasing the maximum number of allowed routes, while the handling costs drastically decrease at first and stabilize eventually. This trend is true for all policies and all handling cost parameter values.

Fig. 3 shows the gaps of both myopic policies as percentages of heuristic policy. In line with the observation of Section 6.4, we see that myopic policy 1 outperforms myopic policy 2 in all configurations. When the number of routes is small, the handling cost component is the major driver of the objective value and it declines when an increasing number of routes is constructed. This leads to the property that the performance of myopic policy 1 as compared to the heuristic policy increases with the number of routes, and even becomes competitive with the heuristic policy when $K \geq 5$. From a practical perspective, it makes sense to use myopic policy 1 rather than the heuristic policy since it is an uncomplicated rule which the vehicle drivers can easily understand and execute and it computes solutions 32% faster than the heuristic policy.

7. Conclusion

We have introduced the vehicle routing problem with simultaneous pickup and delivery and handling costs (VRPSPD-H). We show that our problem generalizes both the vehicle routing problem with simultaneous pickup and delivery (VRPSPD), the variant without handling operations, and the single-vehicle variant called the traveling salesman problem with pickups, deliveries and handling costs (TSPPD-H). We have also shown that by reformulating the problem we can solve the vehicle routing problem with mixed pickup and delivery with handling costs (VRPMPD-H), which is also first introduced in this study, and its variant without handling costs (VRPMPD). Another related problem that can be solved from the reformulated model is the vehicle routing problem with divisible pickup and delivery (VRPDPD). Finally, we prove that the reformulated model can generate only upper bound solutions to the VRPDPD with handling costs (VRPDPD-H), which is another new problem in literature proposed here. We studied a heuristic handling policy which approximates the optimal decisions for the handling sub-problem, and we derived two new bounds on the optimal dynamic program (DP) policy which were used to define two myopic policies.

To solve our problem, we propose an adaptive large neighborhood search (ALNS) metaheuristic in which we embedded all four handling policies, and we implement the mathematical formulation in CPLEX to solve small instances optimally. With the ALNS metaheuristic, we also solve instances of the VRPSPD, the VRPDPD, the VRPMPD, and the TSPPD-H. Results on these special cases and related problems have shown that our ALNS metaheuristic improves 11 and finds 28 best known solutions (BKS) out of 54 instances from two sets of benchmark instances for the VRPSPD, improves 12 and finds 24 BKS out of 54 instances from two sets for the VRPDPD, and improves 14 and finds 1 BKS out of 21 instances from one set for the VRPMPD. For a set of 80 benchmark instances for the TSPPD-H, we find or improve 69 BKS with the heuristic handling policy. The DP handling policy, at the cost of significantly higher calculation times, performs slightly better on average, but finds or improves only 68 BKS. The average gaps for the two policies are −0.96% and −0.82% respectively. Furthermore, we show that our proposed metaheuristic finds optimal solutions on most instances for the VRPSPD-H of up to 15 customers, for the VRPDPD-H of up to 10 customers, and for the VRPMPD-H of up to 20 customers.

We also study the quality of the two myopic policies and the impact of the number of constructed routes on the objective values, where all handling policies are embedded in our ALNS structure. We see that for the VRPSPD-H when the number of constructed routes increases, routing costs stay fairly constant while the handling cost component, and thus the objective value, decreases significantly and stabilizes eventually. Myopic policy 1 outperforms myopic policy 2 in all configurations for this problem, and the difference between myopic policy 1 and the heuristic policy declines when the number of available vehicles is increased. Furthermore, the computation times of both myopic policies are similar and are 32% lower than the computation time when applying the heuristic policy. For the VRPDPD-H and the VRPMDP-H, differences on performance of the myopic policies and the heuristic policy is even lower. From a practical perspective, it may then be preferred to implement the simple myopic policy 1 rather than the more complicated heuristic policy when the two have similar performance. As suggestion for future research, exact methods are required to obtain dual bounds for the three new problems with handling costs, and new heuristic algorithms could provide competitive results to the those of our ALNS.

CRediT authorship contribution statement

Richard P. Hornstra: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - original draft, Visualization. **Allyson Silva:** Software, Validation, Formal analysis, Investigation, Writing - review & editing, Visualization. **Kees Jan Roodbergen:** Conceptualization, Methodology, Validation, Formal analysis, Writing - original draft, Funding acquisition. **Leandro C. Coelho:** Conceptualization, Methodology, Validation, Formal analysis, Writing - original draft, Funding acquisition.

Acknowlgedgments

The authors thank Gabor Nagy for kindly providing all information to correctly work with his dataset. This work was supported by the Netherlands Organisation for Scientific Research (NWO) under grant 439.16.612, and by the Canadian Natural Sciences and Engineering Research Council under grant 2019-00094. We thank Compute Canada for providing high-performance computing facilities. This support is gratefully acknowledged. We thank our associate editor and two anonymous referees for their valuable suggestions on an earlier version of this paper.

References

Avci, M., Topaloglu, S., 2015. An adaptive local search algorithm for vehicle routing problem with simultaneous and mixed pickups and deliveries. Comput. Ind. Eng. 83, 15–29.

Battarra, M., Erdoğan, G., Laporte, G., Vigo, D., 2010. The traveling salesman problem with pickups, deliveries, and handling costs. Transport. Sci. 44 (3), 383–399.

Benavent, E., Landete, M., Mota, E., Tirado, G., 2015. The multiple vehicle pickup and delivery problem with LIFO constraints. Eur. J. Oper. Res. 243 (3), 752–762.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., Laporte, G., 2007. Static pickup and delivery problems: a classification scheme and survey. Top 15 (1), 1–31.

Casco, D., Golden, B.L., Wasil, E.A., 1988. Vehicle routing with backhauls: models, algorithms and case studies. In: Golden, B.L., Assad, A. (Eds.), Vehicle Routing: Methods and Studies. North-Holland, Amsterdam, pp. 127–147.

Cherkesly, M., Desaulniers, G., Laporte, G., 2015. A population-based metaheuristic for the pickup and delivery problem with time windows and LIFO loading. Comput. Oper. Res. 62, 23–35.

Coelho, L.C., Renaud, J., Laporte, G., 2016. Road-based goods transportation: a survey of real-world logistics applications from 2000 to 2015. INFOR: Inform. Syst. Oper. Res. 54 (2), 79–96.

Côté, J.-F., Gendreau, M., Potvin, J.-Y., 2012. Large neighborhood search for the pickup and delivery traveling salesman problem with multiple stacks. Networks 60 (1), 19–30.

Crispim, J., Brandão, J., 2005. Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls. J. Oper. Res. Soc. 56 (11), 1296–1302.

Dell'Amico, M., Righini, G., Salani, M., 2006. A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. Transport. Sci. 40 (2), 235–247.

Dethloff, J., 2001. Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. OR Spektrum 23 (1), 79–96.

Emeç, U., Çatay, B., Bozkaya, B., 2016. An adaptive large neighborhood search for an e-grocery delivery routing problem. Comput. Oper. Res. 69, 109–125.

Erdoğan, G., Battarra, M., Laporte, G., Vigo, D., 2012. Metaheuristics for the traveling salesman problem with pickups, deliveries and handling costs. Comput. Oper. Res. 39 (5), 1074–1086.

Gajpal, Y., Abad, P., 2009. An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup. Comput. Oper. Res. 36 (12), 3215–3223.

Gendreau, M., Laporte, G., Vigo, D., 1999. Heuristics for the traveling salesman problem with pickup and delivery. Comput. Oper. Res. 26 (7), 699–714.

Glover, F., Woolsey, E., 1974. Converting the 0–1 polynomial programming problem to a 0–1 linear program. Operations Research 22 (1), 180–182.

Grangier, P., Gendreau, M., Lehuédé, F., Rousseau, L.-M., 2016. An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. Eur. J. Oper. Res. 254 (1), 80–91.

Hübner, A., Ostermeier, M., 2018. A multi-compartment vehicle routing problem with loading and unloading costs. Transport. Sci. 53 (1), 282–300.

Kalayci, C.B., Kaya, C., 2016. An ant colony system empowered variable neighborhood search algorithm for the vehicle routing problem with simultaneous pickup and delivery. Expert Systems with Applications 66, 163–175.

Koç, Ç., Laporte, G., 2018. Vehicle routing with backhauls: review and research perspectives. Comput. Oper. Res. 91, 79–91.

Li, Y., Chen, H., Prins, C., 2016. Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests. Eur. J. Oper. Res. 252 (1), 27–38.

Nagy, G., Salhi, S., 2005. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. Eur. J. Oper. Res. 162 (1), 126–141.

Nagy, G., Wassan, N.A., Speranza, M.G., Archetti, C., 2013. The vehicle routing problem with divisible deliveries and pickups. Transport. Sci. 49 (2), 271–294.

Polat, O., 2017. A parallel variable neighborhood search for the vehicle routing problem with divisible deliveries and pickups. Comput. Oper. Res. 85, 71–86.

Pollaris, H., Braekers, K., Caris, A., Janssens, G.K., Limbourg, S., 2015. Vehicle routing problems with loading constraints: state-of-the-art and future directions. OR Spectrum 37 (2), 297–330.

Reimann, M., Ulrich, H., 2006. Comparing backhauling strategies in vehicle routing using ant colony optimization. Central European Journal of Operations Research 14 (2), 105–123.

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

Ropke, S., Pisinger, D., 2006. A unified heuristic for a large class of vehicle routing problems with backhauls. Eur. J. Oper. Res. 171 (3), 750–775.

Salhi, S., Nagy, G., 1999. A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. J. Oper. Res. Soc. 50 (10), 1034–1042.

Shaw, P., 1998. Using constraint programming and local search methods to solve vehicle routing problems. Computer 1520, 417–431.

Subramanian, A., Uchoa, E., Pessoa, A., Ochi, L.S., 2013. Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery. Optim. Lett. 7 (7), 1569–1581.

Veenstra, M., Roodbergen, K.J., Vis, I.F.A., Coelho, L.C., 2017. The pickup and delivery traveling salesman problem with handling costs. Eur. J. Oper. Res. 257 (1), 118–132.

Wang, H.F., Chen, Y.Y., 2012. A genetic algorithm for the simultaneous delivery and pickup problems with time window. Comput. Ind. Eng. 62 (1), 84–95.

Wassan, N.A., Nagy, G., 2014. Vehicle routing problem with deliveries and pickups: modelling issues and meta-heuristics solution approaches. Int. J. Transport. 2 (1), 95–110.

Wassan, N.A., Nagy, G., Ahmadi, S., 2008. A heuristic method for the vehicle routing problem with mixed deliveries and pickups. J. Sched. 11 (2), 149–161.

Zachariadis, E.E., Kiranoudis, C.T., 2011. A local search metaheuristic algorithm for the vehicle routing problem with simultaneous pick-ups and deliveries. Exp. Syst. Applica. 38 (3), 2717–2726.