CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

The Vehicle Routing Problem with Simultaneous Pickup and Delivery and Handling Costs

Richard P. Hornstra
Kees Jan Roodbergen
Leandro C. Coelho

June 2018

CIRRELT-2018-27

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2018-015.

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

The Vehicle Routing Problem with Simultaneous Pickup and Delivery and Handling Costs

Richard P. Hornstra[1], Kees Jan Roodbergen[1,*], Leandro C. Coelho[2]

1. University of Groningen, Faculty of Economics and Business, Department of Operations, Netherlands
2. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, 2325 de la Terrasse, Université Laval, Québec, Canada G1V 0A6

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 destinate 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 policy, 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 myopic policy 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) heuristic to solve our problem, in which we embed the handling policies. Computational results indicate that our heuristic finds optimal solutions on instances of up to 15 customers. We also compare our ALNS heuristic 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). We find 32 out of 54 best known solutions for the VRPSPD, and we find or improve 71 out of 80 best known solutions for the TSPPD-H.

Keywords. Vehicle routing problem, pickup and delivery, handling policies, hybrid heuristic.

Acknowledgements. This work was supported by the Netherlands Organisation for Scientific Research (NWO) under grant 439.16.612, and by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant 2014-05764. This support is gratefully acknowledged.

---

* Corresponding author: k.j.roodbergen@rug.nl

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2018

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 collection and delivery of damaged and undamaged bicycles in public sharing systems [2] or when delivering calves to farms and collecting mature cows [8]. Recently, other studies looked at the effect of obstruction issues in related problem settings [e.g., 19].

We refer to our problem as the vehicle routing problem 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 have both a delivery and a pickup demand. These demands are such that all delivery items originate from and all pickup items destinate 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 when a delivery is to be made), handling operations are required before the desired service can be made.

Our problem generalizes the vehicle routing problem with simultaneous pickup and delivery (VRPSPD) introduced by Min [12] by extending it with handling operations. It also generalizes the single vehicle equivalent which is called traveling salesman problem with pickups, deliveries and handling costs (TSPPD-H) as introduced by Battarra et al. [2] by allowing for the construction of multiple routes.

In this paper, we introduce, model and solve the VRPSPD-H. We compare the performance of a heuristic handling policy which approximates the optimal decisions with two new myopic policies. These are obtained by deriving bounds on the optimal handling policy. We propose a mathematical formulation which we implement in CPLEX to solve small problem instances optimally and we propose an adaptive large neighborhood search (ALNS) metaheuristic in which we embed the handling policies to also solve larger problem instances. The quality of the proposed heuristic is shown by benchmarking on two special cases and by comparison with optimal results obtained from our mathematical formulation.

A closely related line of research is Veenstra et al. [19], in which a single vehicle fulfils 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, which may both be different from the depot. 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. [2], 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 unrestricted problem, the authors introduce three handling policies and solve instances up to 25 customers optimally. The authors show that their Policy 3, which we describe in Section 2.1, significantly outperforms the other two policies.

A follow-up study by Erdoğan et al. [8] focuses on Policy 3 of Battarra et al. [2]. The authors design an exact dynamic program (DP) with quadratic complexity and an approximate, linear-time heuristic which 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 adopt their approach to the handling sub-problem, in addition to our myopic policies, and integrate it with our metaheuristic for the VRPSPD-H.

Many different heuristic methods have been proposed to solve the VRPSPD, including adaptive local search [1], ant colony systems [9, 11] and tabu search [21]. Despite the successes of these techniques, ALNS is growing in popularity over the last years. It extends the LNS as first introduced by Shaw [17] by an adaptive mechanism and has recently been implemented successfully in many different routing problems. We build upon these recent successes and design an ALNS metaheuristic for our problem.

Additional to heuristic solution methods, the VRPSPD has 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. [6] use a branch-and-price method to solve instances of up to 40 customers optimally and Subramanian et al. [18] 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. [6] to fit our problem and use it to solve small instances optimally.

Other areas of research which are less related are the multi-vehicle pickup and delivery problem with LIFO constraints studied in Benavent et al. [3] and the single vehicle variant with time windows in Cherkesly et al. [4]. 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. Finally, Wang and

Chen [20] study the VRPSPD with time windows and an extension with multiple depots is studied in Nagy and Salhi [13].

The remainder of this paper is structured as follows. In Section 2 we present a formal problem definition and we elaborate on the heuristic handling policy, and Section 3 gives properties of the model and shows the aforementioned generalizations. Section 4 explains the ALNS metaheuristic we propose to solve the problem and we report the results of an extensive numerical study in Section 5. Finally, Section 6 concludes the paper.

2 Problem definition

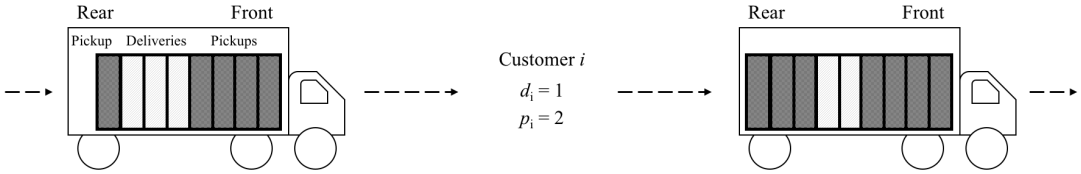
This section presents the model for the VRPSPD-H and is structured as follows. We formally define the model, explain the handling policy and propose a MIP formulation in Section 2.1. We implement this formulation in CPLEX and present the results in Section 5.3. To speed up calculation times, we present valid inequalities in Section 2.2. Finally, we elaborate on how we approach the proposed handling policy in Section 2.3.

2.1 Mathematical formulation

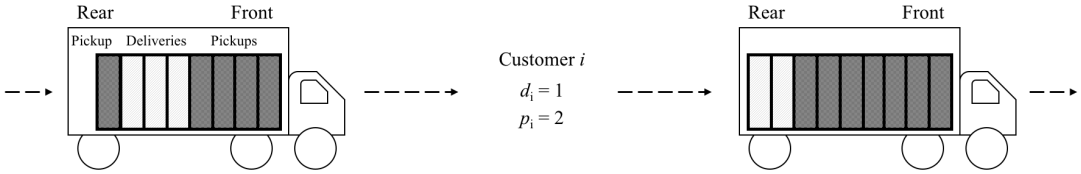
The VRPSPD-H is defined on a complete directed graph $G = (V, A)$, with $V = \{0, 1, \ldots, n\}$ being the set of vertices and A is the arc set. Let vertex 0 represent the depot, then $V_c = V \backslash \{0\}$ is the set of customer vertices. Define $A_r = \{(i, 0) : i \in V_c\}$ as the set of arcs which end at the depot. A positive travel cost c_{ij} satisfying the triangle inequality corresponds to each arc $(i, j) \in A$. Customer $i \in V_c$ requires d_i delivery items and supplies p_i 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. [2], 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. Our handing policy corresponds to Policy 3 of Battarra et al. [2]. Under this policy, the load in the vehicle 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. At each customer, the decision of placing the pickup items either at the rear or at

the front of the vehicle is made. If the pickup items are placed at the front of the vehicle, additional operations for the delivery items in the vehicle are required.



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



(b) Placement of pickup items at the front of the vehicle.

Figure 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).

Figure 1 depicts both options 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 (Figure 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. Alternatively (Figure 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.

In a flow based formulation, let x_{ij} be a binary variable indicating if arc $(i, j) \in A$ is part of the solution. 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 $(i, j) \in A$. Finally, we introduce the binary variable s_i, $i \in V_c$, indicating at each customer whether the pickup items are placed at the front or at the rear of the vehicle. Inspired by the models for the TSPPD-H by Battarra et al. [2] and the VRPSPD by Dell'Amico et al. [6], we

propose the following formulation:

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

$$\text{subject to} \quad \sum_{j\in V} x_{ij} = 1, \qquad i \in V_c, \tag{2}$$

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

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

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

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

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

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

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

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

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. 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.

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. 2]:

$$w_{ij} + y_{ij} + z_{ij} \leq x_{ij} \left(Q + \min\{0, p_i - d_i, d_j - p_j\}\right).$$

Furthermore, we restrict the possibility of constructing a route from a node to itself, $x_{ii} = 0$. Finally we 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, $\sum_{i \in V_c} w_{0i} = 0$, $\sum_{i \in V_c} z_{0i} = 0$, and $\sum_{i \in V_c} y_{i0} = 0$.

2.3 Heuristic handling policy

As previously mentioned, the handling policy adopted in our model corresponds to Policy 3 of Battarra et al. [2]. Erdoğan et al. [8] extensively studied the handling sub-problem, and we use their results as a basis for our analysis. 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 modelled as a dynamic program in Erdoğan et al. [8] and it gives the optimal choices for any given route in $O(n^2)$ time. Due to the time complexity of the DP, the authors also propose a linear time heuristic based on a special case where the demands of all customers are the same.

In this heuristic, the authors experimented with four different thresholds which trigger the placement of all pickup items at the front of the vehicle. Experiments showed that the number of pickup items on board was the best threshold measure, which is computed as the average of all pickup items of the remaining customers in the route. If the number of pickup items on board 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, which is why we include this heuristic handling policy in comparisons.

3 Special cases and properties

In this section, we report on some useful properties of our problem. We will first prove in Section 3.1 that the VRPSPD and the TSPPD-H are special cases of the VRPSPD-H. In Section 3.2 we give two bounds on the optimal handling policy, which we then use to define two myopic policies.

3.1 Special cases

In this section, we show that the 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, we can omit all constraints involving handling operations (constraints (7) and (9)) and remove variables w_{ij} entirely. The remaining model is equivalent to the VRPSPD as in (1)–(9) in Dell'Amico et al. [6]. □

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. [2]. 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 (31)–(48) in Battarra et al. [2]. □

3.2 Bounds on the optimal handling policy

In this section we propose two bounds on the optimal handling policy and formulate two alternative myopic handling policies based on these bounds. The performance of the myopic handling policies is studied in Sections 5.4 and 5.5. We first introduce new notation before we propose the bounds.

Let a route with $n \leq |V_c|$ customers be denoted by a permutation $\phi(\cdot)$ of the location indices, such that $\phi(i)$ is the index of the i-th customer on the route. We consider the decision at customer i whether or not to place the pickup items at the front of the vehicle, given handling decisions at all customers $\phi(j)$

for $j < i$. Consistent with our notation in Section 2.1, we use $s_{\phi(i)}$ to represent the handling decisions, where $s_{\phi(i)} = 1$ if the pickup items are placed at the front of the vehicle, and $s_{\phi(i)} = 0$ otherwise.

3.2.1 Upper bound

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 $s_{\phi(1)}, \ldots, s_{\phi(i-1)}$, at customer $\phi(i)$, it is always optimal to place $p_{\phi(i)}$ and $\sum_{j=1}^{i-1} \left(\prod_{k=j}^{i-1} \left(1 - s_{\phi(k)}\right) p_{\phi(j)} \right)$ at the front of the vehicle if*

$$h_p \left[p_{\phi(i)} + \sum_{j=1}^{i-1} \left(\prod_{k=j}^{i-1} \left(1 - s_{\phi(k)}\right) p_{\phi(j)} \right) \right] > h_d \sum_{j=i+1}^{n} d_{\phi(j)}. \tag{11}$$

Proof. Assume a route with $n \geq 2$ customers, and that customer $\phi(i)$ is not the last customer in the route. Let handling decisions $s_{\phi(1)}, \ldots, s_{\phi(i-1)}$ be given. There are two options. Option 1 is to place $p_{\phi(i)}$ at the rear of the vehicle with cost $h_p \sum_{j=1}^{i-1} \left(\prod_{k=j}^{i-1} \left(1 - s_{\phi(k)}\right) p_{\phi(j)} \right)$ at customer $\phi(i)$ and cost $h_p \left(p_{\phi(i)} + \sum_{j=1}^{i-1} \left(\prod_{k=j}^{i-1} \left(1 - s_{\phi(k)}\right) p_{\phi(j)} \right) \right)$ at customer $\phi(i+1)$. Option 2 is to place $p_{\phi(i)}$ and $\sum_{j=1}^{i-1} \left(\prod_{k=j}^{i-1} \left(1 - s_{\phi(k)}\right) p_{\phi(j)} \right)$ at the front of the vehicle with cost $h_p \sum_{j=1}^{i-1} \left(\prod_{k=j}^{i-1} \left(1 - s_{\phi(k)}\right) p_{\phi(j)} \right) + h_d \sum_{j=i+1}^{n} d_{\phi(j)}$ at customer $\phi(i)$ and cost 0 at customer $\phi(i+1)$. Inequality (11) follows from this. It can then be seen that placing $p_{\phi(i)}$, and thus also $\sum_{j=1}^{i-1} \left(\prod_{k=j}^{i-1} \left(1 - s_{\phi(k)}\right) p_{\phi(j)} \right)$, at the front of the vehicle is always optimal if (11) holds. That is, given handling decisions at all customers visited prior to arriving at customer $\phi(i)$, 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 $\phi(i)$, 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 $\phi(i)$ are placed at the front of the vehicle if and only if inequality (11) holds.

3.2.2 Lower bound

Similar as in Section 3.2.1, 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 $s_{\phi(1)}, \ldots, s_{\phi(i-1)}$, at customer $\phi(i)$, it is always optimal to place $p_{\phi(i)}$ at the rear of the vehicle if*

$$h_p(n-i)\left[p_{\phi(i)} + \sum_{j=1}^{i-1}\left(\prod_{k=j}^{i-1}\left(1-s_{\phi(k)}\right)p_{\phi(j)}\right)\right] < h_d \sum_{j=i+1}^{n} d_{\phi(j)}. \tag{12}$$

Proof. Assume a route with $n \geq 2$ customers, that customer $\phi(i)$ is not the last customer in the route, and that $p_{\phi(j)}$, $i < j \leq n$ are placed at the rear of the vehicle. Let handling decisions $s_{\phi(1)}, \ldots, s_{\phi(i-1)}$ be given. There are two options. Option 1 is to place $p_{\phi(i)}$ at the rear of the vehicle at cost $h_p(n-i)\sum_{j=1}^{i-1}\left(\prod_{k=j}^{i-1}\left(1-s_{\phi(k)}\right)p_{\phi(j)}\right) + h_p\sum_{k=i}^{n-1}(n-k)p_{\phi(k)}$ for the remainder of the route. Option 2 is to place $p_{\phi(i)}$ and $\sum_{j=1}^{i-1}\left(\prod_{k=j}^{i-1}\left(1-s_{\phi(k)}\right)p_{\phi(j)}\right)$ at the front of the vehicle at cost $h_d\sum_{j=i+1}^{n} d_{\phi(j)} + h_p\sum_{k=i+1}^{n-1}(n-k)p_{\phi(k)}$ for the remainder of the route. It follows that if

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

holds, it is always optimal to place $p_{\phi(i)}$ at the rear of the vehicle. That is, given handling decisions at all customers visited prior to arriving at customer $\phi(i)$, it is optimal to place the pickup items of customer $\phi(i)$ 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 $\phi(i)$ are placed at the rear of the vehicle if and only if inequality (12) holds.

We note that the pickup items of the last customer in any route never require additional operations since the vehicle visits the depot directly thereafter, and that there is no difference between placing the pickup items at the front or rear of the vehicle as there are no remaining delivery items in the vehicle. Additionally, at the second last customer in any route, inequalities (11) and (12) give the same decision, indicating that this is the optimal decision.

4 Adaptive large neighborhood search heuristic

This section provides details about the ALNS heuristic designed to solve the VRPSPD-H. As mentioned in Section 1, LNS was introduced by Shaw [17] 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 [15], handling costs [19], multiple stacks [5] and other restrictions such as time windows [14]. 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 [14] as a basis for our design, where we enrich 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.

Details on the construction of an initial solution are reported in Section 4.1, and Sections 4.2 and 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.

Algorithm 1 Outline of ALNS heuristic

```
construct initial solution s
solution s_best ← s
repeat
    s' ← s
    destroy s'
    repair s'
    if (f(s') < f(s_best)) then
        local search s'
        s_best ← s'
    end if
    if accept(s', s) then
        s ← s'
    end if
    update operator weights
    if operator weights are reset then
        local search s
    end if
until stopping criterion
return s_best
```

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. 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 heuristic, 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 [14], whereas the *worst distance removal* and *worst handling removal* operators were introduced by Veenstra et al. [19]. The *related removal* operator was described by Shaw [17] 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 [15]. 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. It 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

```
while number of customers in the pool < q do
    c_i(s) ← f(s) − f(s_{−i}), ∀i ∈ V_c
    y ∼ ⌈U[0,1]^p · (number of customers in the solution.)⌉
    remove customer with y-th highest c_i(s)
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. *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.

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. *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, $R(i, j) = \dfrac{1}{c_{ij}}$.

   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 customer directly before and the customer directly after the selected customer, if present. Next, using the relatedness measure $R(i, j)$, a related customer in a different route and its neighboring customers are selected. All these customers, or at most q, are removed from the solution, and 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. [19]. 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 [14]. 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 heuristic.

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 are used: *10exchange*, *11exchange*, *intra 2-opt*, *inter 2-opt* and *inter 3-opt*. The *10exchange* operator finds the best reinsertion of a single customer, and the *11exchange* 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 *10exchange* 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

```
for k in 1 : 5 do
    improve ← true
    while improve do
        improve ← false
        Apply operator k to solution
        if improvement found then
            k ← 1
            improve = true
        end if
    end while
end for
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

$$P(\text{accept } s') = \exp\left\{-\frac{f(s') - f(s)}{T}\right\}, \tag{13}$$

where T is the temperature at the current iteration. The starting temperature is determined at the start of the heuristic, 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 [14]. 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 [14] 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 Computational results

The ALNS heuristic was programmed in C++ and the mathematical model of Section 2.1 was implemented in C++ and solved with CPLEX 12.7.1. All our experiments were run on a 2.7 GHz Intel Core i5 processor. We provide details on the parameter configuration in Section 5.1. We test our ALNS heuristic on well-known benchmark instances of the VRPSPD and TSPPD-H, which are special cases of our problem, in Section 5.2. A comparison with optimal solutions for our problem is made in Section 5.3. We compare the performance of the heuristic handling policy and the two myopic policies in Section 5.4. Finally we investigate the influence of the number of available vehicles on the trade-off between routing and handling costs in Section 5.5.

5.1 Tuning

This section provides details about the parameter settings of the proposed ALNS heuristic. For the purpose of tuning the 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 [7] 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 of instances Dethloff [7] for the VRPSPD and on instances for the TSPPD-H by Erdoğan et al. [8].

As a starting point for the parameter tuning, we used the values reported by Ropke and Pisinger [15]. 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 is initialized with value 3, and we remove $q \in [0.15|V_c|, 0.3|V_c|]$ customers in each iteration. We find that setting the maximum number

of iterations equal to 25,000 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 [15] and Veenstra et al. [19], indicating that the algorithm is robust.

Since the handling cost component significantly increases the problem complexity, evaluation of solutions 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.

5.2 Benchmarks on special cases

To test the quality of our heuristic when applied to special cases, we use it to solve various instances for both the VRPSPD and the TSPPD-H. Our obtained results are then compared to results from the literature to check the quality of our heuristic. We note that calculation times may not be competitive compared to heuristic methods especially designed for one of the two special cases, since the generalizations made in this paper result in unnecessary overhead when it comes to the special cases.

5.2.1 Performance on the VRPSPD

Two well-known sets of benchmark instances for the VRPSPD are solved. The 40 instances of Dethloff [7] 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 [16] are solved. These instances range in size from 50 to 199 customers. We note that all best known solutions for instances of Dethloff [7] are proven to be optimal in Subramanian et al. [18].

Table 1 reports the results on the 40 instances of Dethloff [7]. We solved each instance 10 times and report our best found solution as well as the average objective value over these 10 runs. Our heuristic finds 30 out of 40 best known solutions, and on average over all 40 instances the gap with the best known solutions is 0.14%. The results on the benchmark instances by Salhi and Nagy [16] are reported in Table 2. Of the 14 instances, we find 2 best known solutions and our gap is on average 0.84%. We

note that performance on the CMT11X and CMT11Y instances is not in line with the other results, which are more competitive.

Table 1: Computational results for the VRPSPD benchmark instances of Dethloff [7].

Instance	n	BKS	Our best	Our average	Gap (%)	Time (s)
SCA3-0	50	635.62	636.06	640.81	0.07	19.14
SCA3-1	50	697.84	**697.84**	698.37	**0.00**	19.91
SCA3-2	50	659.34	**659.34**	659.52	**0.00**	20.03
SCA3-3	50	680.04	680.60	680.60	0.08	19.72
SCA3-4	50	690.50	**690.50**	692.28	**0.00**	19.09
SCA3-5	50	659.90	**659.90**	663.83	**0.00**	21.57
SCA3-6	50	651.09	**651.09**	651.46	**0.00**	19.60
SCA3-7	50	659.17	666.15	670.76	1.06	18.57
SCA3-8	50	719.47	**719.48**	**719.48**	**0.00**	19.75
SCA3-9	50	681.00	**681.00**	681.02	**0.00**	19.14
SCA8-0	50	961.50	**961.50**	971.97	**0.00**	16.34
SCA8-1	50	1049.65	**1049.65**	1059.10	**0.00**	16.55
SCA8-2	50	1039.64	1050.14	1050.35	1.01	16.83
SCA8-3	50	983.34	**983.34**	1002.43	**0.00**	15.76
SCA8-4	50	1065.49	**1065.49**	1068.07	**0.00**	15.89
SCA8-5	50	1027.08	**1027.08**	1037.69	**0.00**	15.93
SCA8-6	50	971.82	**971.82**	972.91	**0.00**	15.75
SCA8-7	50	1051.28	1063.22	1067.14	1.14	16.08
SCA8-8	50	1071.18	**1071.18**	1074.33	**0.00**	15.89
SCA8-9	50	1060.50	**1060.50**	1064.55	**0.00**	15.98
CON3-0	50	616.52	619.09	627.28	0.42	16.69
CON3-1	50	554.47	**554.47**	556.90	**0.00**	17.10
CON3-2	50	518.00	521.38	521.38	0.65	16.97
CON3-3	50	591.19	**591.19**	592.43	**0.00**	16.08
CON3-4	50	588.79	**588.79**	593.12	**0.00**	16.49
CON3-5	50	563.70	**563.70**	575.21	**0.00**	16.02
CON3-6	50	499.05	500.80	502.58	0.35	16.13
CON3-7	50	576.48	**576.48**	583.77	**0.00**	16.13
CON3-8	50	523.05	**523.05**	526.04	**0.00**	15.87
CON3-9	50	578.25	**578.25**	585.32	**0.00**	16.19
CON8-0	50	857.17	**857.17**	860.97	**0.00**	15.66
CON8-1	50	740.85	**740.85**	749.66	**0.00**	16.04
CON8-2	50	712.89	**712.89**	713.38	**0.00**	16.45
CON8-3	50	811.07	**811.07**	819.24	**0.00**	16.38
CON8-4	50	772.25	**772.25**	774.55	**0.00**	16.51
CON8-5	50	754.88	**754.88**	759.22	**0.00**	15.96
CON8-6	50	678.92	684.69	691.66	0.85	15.94
CON8-7	50	811.96	**811.96**	814.92	**0.00**	16.05
CON8-8	50	767.53	768.64	777.62	0.14	16.31
CON8-9	50	809.00	**809.00**	810.63	**0.00**	16.81
Average		758.54	759.66	764.06	0.14	17.08

5.2.2 Performance on the TSPPD-H

A third benchmark is performed on the instances proposed by Erdoğan et al. [8]. The authors adapted 10 instances containing 200 customers proposed by Gendreau et al. [10]. Smaller instances were created

Table 2: Computational results for the VRPSPD benchmark instances of Salhi and Nagy [16].

CMT	n	BKS	Our best	Our average	Gap (%)	Time (s)
1X	50	466.77	**466.77**	468.576	**0.00**	9.60
1Y	50	466.77	**466.77**	468.97	**0.00**	9.50
2X	75	684.21	684.75	692.231	0.08	19.78
2Y	75	684.21	684.89	694.103	0.10	19.74
3X	100	721.27	722.094	725.411	0.11	37.84
3Y	100	721.27	722.09	724.746	0.11	37.44
4X	150	852.46	854.17	862.785	0.20	85.07
4Y	150	852.46	855.70	863.951	0.38	84.70
5X	199	1029.25	1033.93	1050.22	0.45	147.77
5Y	199	1029.25	1032.76	1051.16	0.34	144.26
11X	120	833.92	873.89	888.03	4.79	59.66
11Y	120	833.92	874.15	878.56	4.82	59.86
12X	100	662.22	663.50	677.462	0.19	37.79
12Y	100	662.22	663.50	674.06	0.19	37.11
Average		750.01	757.07	765.73	0.84	56.44

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 $h|V_c| = 20$ is kept constant. For further details concerning the exact configuration of these instances we refer to Erdoğan et al. [8]. In our results, we left out the instances with $|V_c| = 80, 100$. For these instances, the results reported in Erdoğan et al. [8] have objectives smaller or close to those of the $|V_c| = 60$ 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 heuristic 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 heuristic, 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. [8] and compare these outcomes with their best overall and best heuristic solutions. Note that in the heuristic approximation, neighborhood searches are evaluated with the heuristic method but when a change in the solution is made, the costs are computed with the DP. We present the results in Table 3.

Our ALNS with heuristic policy finds or improves 64 out of 80 best known solutions, which were found with the DP in Erdoğan et al. [8], and the average gap is -0.73%. Our ALNS with heuristic policy 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. [8]. Evaluation of our ALNS with optimal DP further improves the results, finding or improving best known solutions on 69 out of

80 instances with an average gap of −0.99%, while average calculation times are on average 13 times higher.

Table 3: Computational results for the TSPPD-H instances of Erdoğan et al. [8].

$\|V_c\|$	*Id.*	Erdoğan et al. [8]			Our ALNS with heuristic policy				Our ALNS with optimal policy			
		Optimal	Heuristic	Gap (%)	Best	Average	Gap (%)	Time (s)	Best	Average	Gap (%)	Time (s)
20	1	633.00	634.00	0.16	**633.00**	**633.00**	**0.00**	3	**633.00**	**633.00**	**0.00**	9
	2	584.00	587.00	0.51	**584.00**	584.30	**0.00**	3	**584.00**	**584.00**	**0.00**	9
	3	573.00	590.00	2.97	**573.00**	574.40	**0.00**	3	**573.00**	**573.00**	**0.00**	9
	4	706.00	712.00	0.85	**706.00**	**706.00**	**0.00**	3	**706.00**	**706.00**	**0.00**	9
	5	501.00	507.00	1.20	**501.00**	**501.00**	**0.00**	3	**501.00**	**501.00**	**0.00**	9
	6	578.00	**578.00**	**0.00**	**578.00**	**578.00**	**0.00**	3	**578.00**	**578.00**	**0.00**	9
	7	612.00	619.00	1.14	**612.00**	**612.00**	**0.00**	3	**612.00**	**612.00**	**0.00**	9
	8	567.00	571.00	0.71	**567.00**	**567.00**	**0.00**	3	**567.00**	**567.00**	**0.00**	9
	9	604.00	625.00	3.48	**604.00**	**604.00**	**0.00**	3	**604.00**	**604.00**	**0.00**	10
	10	574.00	588.00	2.44	**565.00**	**565.00**	**−1.57**	3	**565.00**	**566.70**	**−1.57**	9
40	1	909.50	915.50	0.66	**909.50**	912.70	**0.00**	15	913.50	913.50	0.44	78
	2	885.00	902.00	1.92	**883.00**	885.25	**−0.23**	15	**883.00**	886.15	**−0.23**	79
	3	815.50	864.00	5.95	**815.50**	817.30	**0.00**	15	**815.50**	**815.50**	**0.00**	79
	4	898.00	919.50	2.39	**898.00**	898.40	**0.00**	15	**898.00**	**898.00**	**0.00**	81
	5	743.50	751.50	1.08	**743.50**	744.40	**0.00**	15	**743.50**	744.55	**0.00**	77
	6	901.00	903.00	0.22	**883.50**	**883.50**	**−1.94**	15	**883.50**	**883.50**	**−1.94**	80
	7	798.50	833.50	4.38	**798.50**	799.40	**0.00**	15	**798.50**	800.15	**0.00**	80
	8	795.00	811.00	2.01	**795.00**	795.40	**0.00**	15	**795.00**	**795.00**	**0.00**	79
	9	876.50	895.50	2.17	**876.50**	878.90	**0.00**	15	**876.50**	**876.50**	**0.00**	79
	10	866.00	901.50	4.10	**862.50**	**862.50**	**−0.40**	15	**862.50**	**862.50**	**−0.40**	80
60	1	1060.12	1063.79	0.35	**1051.00**	**1052.07**	**-0.86**	45	**1051.00**	**1051.00**	**−0.86**	299
	2	1051.13	1060.91	0.93	**1046.67**	**1047.27**	**−0.42**	45	**1044.33**	**1047.03**	**−0.65**	301
	3	990.36	1012.39	2.22	993.67	993.67	0.33	45	993.67	993.67	0.33	289
	4	1061.45	1086.32	2.34	1066.00	1070.13	0.43	45	1066.00	1066.00	0.43	294
	5	986.93	1033.69	4.74	989.67	991.07	0.28	45	989.67	990.07	0.28	295
	6	1086.31	1131.28	4.14	**1078.00**	**1079.87**	**−0.76**	46	**1067.67**	**1070.50**	**−1.72**	296
	7	1005.43	1017.97	1.25	1007.33	1008.47	0.19	45	1007.33	1007.33	0.19	298
	8	1027.19	1059.54	3.15	1031.00	1033.37	0.37	45	1031.00	1032.80	0.37	295
	9	1001.43	1013.90	1.25	1004.33	1004.33	0.29	45	1004.33	1004.33	0.29	290
	10	1062.11	1081.74	1.85	**1048.67**	**1050.07**	**−1.27**	45	**1048.67**	**1061.57**	**−1.27**	297
120	1	1472.49	1526.53	3.67	**1434.67**	**1444.50**	**−2.57**	317	**1436.17**	**1439.95**	**−2.47**	3197
	2	1482.29	1540.24	3.91	**1460.67**	**1467.47**	**−1.46**	321	**1451.00**	**1455.97**	**−2.11**	3200
	3	1510.05	1550.10	2.65	**1474.67**	**1481.60**	**−2.34**	325	**1464.17**	**1472.95**	**−3.04**	3306
	4	1563.50	1628.78	4.18	**1546.50**	**1553.22**	**−1.09**	323	**1539.67**	**1553.45**	**−1.52**	3250
	5	1457.05	1542.96	5.90	**1438.00**	**1452.65**	**−1.31**	320	**1428.33**	**1436.93**	**−1.97**	3208
	6	1546.25	1622.27	4.92	**1531.67**	**1539.52**	**−0.94**	323	**1527.83**	**1538.97**	**−1.19**	3256
	7	1557.27	1610.88	3.44	**1522.83**	**1529.47**	**−2.21**	322	**1512.00**	**1517.55**	**−2.91**	3254
	8	1524.07	1599.61	4.96	**1508.33**	**1519.90**	**−1.03**	315	**1504.83**	**1509.27**	**−1.26**	3257
	9	1547.95	1620.27	4.67	**1531.67**	**1537.65**	**−1.05**	323	**1538.17**	**1540.50**	**−0.63**	3243
	10	1573.87	1622.36	3.08	**1546.67**	**1554.38**	**−1.73**	320	**1556.83**	**1561.42**	**−1.08**	3054
140	1	1575.20	1645.60	4.47	1595.71	1603.97	1.30	503	1577.14	1587.50	0.12	5471
	2	1605.80	1670.76	4.05	**1594.57**	1608.89	**−0.70**	505	**1584.71**	**1600.02**	**−1.31**	5546
	3	1583.68	1634.40	3.20	**1581.86**	1595.21	**−0.11**	506	**1568.00**	**1577.02**	**−0.99**	5520
	4	1712.66	1760.56	2.80	**1677.29**	**1703.31**	**−2.07**	509	**1668.43**	**1703.89**	**−2.58**	5581
	5	1547.40	1610.50	4.08	1551.71	1559.17	0.28	506	**1541.14**	1557.62	**−0.40**	5552
	6	1662.28	1741.18	4.75	**1644.86**	1667.60	**−1.05**	507	**1650.71**	1669.40	**−0.70**	5539
	7	1664.42	1691.36	1.62	**1620.57**	**1637.01**	**−2.63**	504	**1623.29**	**1640.76**	**−2.47**	5561
	8	1646.08	1698.12	3.16	**1610.00**	**1633.63**	**−2.19**	499	**1619.71**	**1636.15**	**−1.60**	5550
	9	1629.10	1700.02	4.35	1645.14	1653.60	0.98	513	1640.00	1647.40	0.67	5665
	10	1693.82	1717.34	1.39	**1683.71**	1696.27	**−0.60**	499	**1663.29**	**1687.65**	**−1.80**	5637

Table 3 (continued)

$\|V_c\|$	*Id.*	Erdoğan et al. [8]			Our ALNS with heuristic policy				Our ALNS with optimal policy			
		Optimal	Heuristic	Gap (%)	Best	Average	Gap (%)	Time (s)	Best	Average	Gap (%)	Time (s)
160	1	1741.75	1836.63	5.45	**1724.88**	**1739.76**	**−0.97**	749	**1702.75**	**1713.65**	**−2.24**	9302
	2	1773.63	1820.50	2.64	**1726.12**	**1742.97**	**−2.68**	751	**1716.12**	**1739.70**	**−3.24**	9177
	3	1690.88	1748.38	3.40	**1688.88**	1712.74	**−0.12**	745	**1676.50**	**1687.03**	**−0.85**	9137
	4	1858.13	1962.63	5.62	**1841.88**	**1853.05**	**−0.87**	756	**1815.88**	**1847.88**	**−2.27**	9177
	5	1667.75	1756.75	5.34	1668.12	1680.65	0.02	748	**1664.75**	1675.57	**−0.18**	9515
	6	1813.00	1844.50	1.74	**1736.50**	**1752.01**	**−4.22**	748	**1735.62**	**1735.81**	**−4.27**	9480
	7	1774.25	1835.38	3.45	**1734.50**	**1758.09**	**−2.24**	760	**1756.62**	**1759.37**	**−0.99**	9326
	8	1770.75	1820.50	2.81	**1757.00**	**1769.17**	**−0.78**	745	**1758.00**	**1758.50**	**−0.72**	9554
	9	1800.88	1893.50	5.14	1801.50	1808.16	0.03	751	**1795.25**	**1796.32**	**−0.31**	9612
	10	1764.38	1809.38	2.55	**1762.25**	1778.99	**−0.12**	746	**1755.62**	1773.25	**−0.50**	9681
180	1	1854.21	1936.01	4.41	**1834.33**	**1851.64**	**−1.07**	1065	**1831.67**	**1834.95**	**−1.22**	14971
	2	1863.32	1927.48	3.44	**1849.33**	1870.47	**−0.75**	1070	**1823.11**	**1831.78**	**−2.16**	14969
	3	1858.41	1889.87	1.69	**1843.22**	1864.76	**−0.82**	1067	**1840.44**	**1848.17**	**−0.97**	14824
	4	1988.19	2059.51	3.59	**1928.67**	**1958.78**	**−2.99**	1095	**1939.22**	**1948.22**	**−2.46**	15116
	5	1795.81	1853.72	3.22	1805.56	1814.12	0.54	1082	**1790.00**	**1790.28**	**−0.32**	15739
	6	1817.47	1873.17	3.06	1832.67	1854.29	0.84	1065	1827.89	1841.00	0.57	15386
	7	1868.04	1929.56	3.29	**1831.78**	**1853.41**	**−1.94**	1077	**1842.56**	**1862.15**	**−1.36**	14612
	8	1883.40	1930.46	2.50	**1875.22**	1889.18	**−0.43**	1079	**1852.67**	**1863.52**	**−1.63**	15093
	9	1931.44	2004.60	3.79	**1926.56**	1940.56	**−0.25**	1122	**1900.67**	**1926.35**	**−1.59**	15119
	10	1852.51	1896.85	2.39	1864.44	1876.29	0.64	1082	**1846.22**	1862.08	**−0.34**	15170
200	1	1976.30	2060.60	4.27	**1949.40**	**1972.11**	**−1.36**	1475	**1956.20**	1958.3	**−1.02**	22068
	2	1982.00	2074.90	4.69	1982.70	2001.94	0.04	1473	**1966.00**	1978.37	**−0.81**	17012
	3	1976.70	2037.10	3.06	**1971.10**	1997.51	**−0.28**	1483	**1964.60**	1964.6	**−0.61**	23778
	4	2119.50	2211.90	4.36	**2054.00**	**2078.57**	**−3.09**	1516	**2038.80**	2042.25	**−3.81**	23102
	5	1905.60	1974.20	3.60	**1894.30**	1913.63	**−0.59**	1482	1925.30	1939.2	1.03	23301
	6	2011.10	2042.30	1.55	**1960.70**	**1982.89**	**−2.51**	1471	**1945.70**	**1967.67**	**−3.25**	24744
	7	1983.00	2046.50	3.20	**1943.00**	**1962.02**	**−2.02**	1468	**1933.90**	**1955.61**	**−2.48**	24455
	8	2000.30	2096.30	4.80	2020.50	2035.87	1.01	1459	**1992.60**	2004.72	**−0.38**	24246
	9	2052.70	2131.60	3.84	**2040.90**	2056.52	**−0.57**	1486	**2008.50**	**2041.85**	**−2.15**	24616
	10	1977.90	2009.40	1.59	**1928.90**	**1951.89**	**−2.48**	1453	**1918.20**	**1927.91**	**−3.02**	25680
Average		1408.27	1453.97	3.00	**1396.06**	**1406.17**	**−0.73**	525	**1391.41**	**1399.18**	**−0.99**	7121

5.3 Benchmark optimal solutions

Since we are the first to model and solve the VRPSPD-H, for this problem no solutions in the literature exist for us to compare with. We have implemented the model of Section 2.1 in CPLEX and compare our heuristic with the optimal solutions on small instances. We have generated new instances by selecting the first 5, 10 and 15 customers of four VRPSPD instances of Dethloff [7]. For these 12 instances, we set the handling cost parameters to $\frac{10}{|V_c|}$ and $\frac{20}{|V_c|}$ and we limit the number of available vehicles to 4. The results are reported in Table 4. We solved each instance with CPLEX on a single thread with a maximum run time of six hours. Instances not solved to optimality are indicated with an asterisk, and the gap with the best lower bound is reported in the column of the calculation time. We compare the results with 10 runs of our ALNS heuristic with the optimal handling policy. CPLEX finds 17 out of 24 optimal solutions in the given time limit, and the remaining 7 instances could not

be solved to optimality. Our ALNS heuristic finds all 17 optimal solutions in every run. Of the 7 best integer solutions, our ALNS heuristic finds the same solution as CPLEX on 4 instances, and beats the CPLEX solutions on the remaining 3 instances. The average time our heuristic needs is 2.28 seconds, compared to 6755.58 seconds for CPLEX.

Table 4: Comparison of ALNS with optimal results on small instances 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	317.735	0	**317.74**	**0.00**	**317.74**	**0.00**	0.71
	5	4	322.077	0	**322.08**	**0.00**	**322.08**	**0.00**	0.71
	10	1	467.774	22	**467.77**	**0.00**	**467.77**	**0.00**	1.91
	10	2	560.026	17	**560.03**	**0.00**	**560.03**	**0.00**	1.87
	15	0.67	664.25*	*12.4%*	**664.25**	**0.00**	**664.25**	**0.00**	4.58
	15	1.33	905.09*	*20.6%*	**905.09**	**0.00**	**905.09**	**0.00**	4.49
SCA8-1	5	2	384.117	0	**384.12**	**0.00**	**384.12**	**0.00**	0.71
	5	4	406.795	0	**406.80**	**0.00**	**406.80**	**0.00**	0.71
	10	1	598.174	100	**598.17**	**0.00**	**598.17**	**0.00**	1.97
	10	2	747.89	68	**747.89**	**0.00**	**747.89**	**0.00**	1.93
	15	0.67	726.14	10460	**726.14**	**0.00**	**726.14**	**0.00**	4.19
	15	1.33	937.35*	*2.1%*	**937.35**	**0.00**	**937.35**	**0.00**	4.14
CON3-0	5	2	291.803	0	**291.80**	**0.00**	**291.80**	**0.00**	0.73
	5	4	319.817	0	**319.82**	**0.00**	**319.82**	**0.00**	0.70
	10	1	667.942	35	**667.94**	**0.00**	**667.94**	**0.00**	1.90
	10	2	893.871	88	**893.87**	**0.00**	**893.87**	**0.00**	1.86
	15	0.67	907.26*	*10.5%*	**907.26**	**0.00**	**907.26**	**0.00**	4.27
	15	1.33	1295.60*	*25.2%*	**1289.31**	**−0.49**	**1289.31**	**−0.49**	4.21
CON8-1	5	2	203.827	0	**203.83**	**0.00**	**203.83**	**0.00**	0.71
	5	4	218.311	0	**218.31**	**0.00**	**218.31**	**0.00**	0.71
	10	1	496.609	66	**496.61**	**0.00**	**496.61**	**0.00**	1.90
	10	2	694.643	64	**694.64**	**0.00**	**694.64**	**0.00**	1.87
	15	0.67	738.037*	*9.0%*	**694.53**	**−5.90**	**694.53**	**−5.90**	3.91
	15	1.33	1079.68*	*23.5%*	**977.28**	**−9.48**	**977.28**	**−9.48**	3.98
Average			618.53	6755.58	**612.19**	**−0.66**	**612.19**	**−0.66**	2.28

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

5.4 Comparison of handling policies

We now compare the performance of the heuristic handling policy compared to myopic policies 1 and 2, respectively. All three handling policies are embedded in our ALNS structure. We have created 40 new VRPSPD-H instances based on the VRPSPD instances of Dethloff [7]. 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, as otherwise solutions resulted in the unrealistic construction of many short routes to decrease handling as much as possible. The results of the experiments are given in Table 5, where we report the average objective values and computation times over 10 runs.

We see that myopic policy 1 outperforms myopic policy 2 on all instances in similar computation times, and that myopic policy 2 never improves the solution found by the heuristic policy. On average, myopic policy 1 has a gap of 0.25% compared to the heuristic policy, and the average gap of myopic policy 2 is 8.47% as compared to the heuristic policy. However, myopic policy 1 finds better solutions than the heuristic policy on 18 out of 40 instances, and its average computation time is around 15% lower. These improvements are mostly obtained when $K = 12$. 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. We characterize such scenarios in Section 5.5.

5.5 Varying number of vehicles

We observed that the heuristic 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. We explicitly distinguish between the routing and handling cost components. The results are presented in Figure 2. The reported values are averages over 10 runs of our ALNS heuristic, 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.

Figure 3 shows the gaps of both myopic policies as percentages of heuristic policy. In line with the observation of Section 5.4, 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 15% faster than the heuristic policy.

Table 5: Comparison of handling policies.

Instance	n	h	K	Heuristic		Myopic policy 1			Myopic policy 2		
				Objective	Time(s)	Objective	Gap (%)	Time (s)	Objective	Gap (%)	Time (s)
SCA3-0	50	0.2	5	1432.74	11.21	1439.85	0.50	9.53	1631.7	13.89	9.26
SCA3-1	50	0.2	5	1484.42	11.20	1515.07	2.06	9.51	1704.65	14.84	9.13
SCA3-2	50	0.2	5	1546.68	11.02	1547.11	0.03	9.38	1782.57	15.25	9.05
SCA3-3	50	0.2	5	1560.34	11.01	1585.06	1.58	9.40	1776.23	13.84	9.16
SCA3-4	50	0.2	5	1738.41	10.96	**1736.3**	**−0.12**	9.28	2035.65	17.10	9.07
SCA3-5	50	0.2	5	1471.7	11.08	1513.31	2.83	9.30	1677.71	14.00	9.09
SCA3-6	50	0.2	5	1438.45	11.10	1458.78	1.41	9.94	1618.83	12.54	9.19
SCA3-7	50	0.2	5	1551.41	11.07	1574.86	1.51	9.71	1779.3	14.69	9.10
SCA3-8	50	0.2	5	1570.08	11.15	1579.22	0.58	10.19	1739.29	10.78	9.27
SCA3-9	50	0.2	5	1471.68	11.25	1499.12	1.86	9.43	1663.73	13.05	9.15
SCA8-0	50	0.2	12	1383.11	8.28	**1382.47**	**−0.05**	8.46	1422.72	2.86	7.48
SCA8-1	50	0.2	12	1481.79	8.11	1482	0.01	7.89	1510.21	1.92	7.32
SCA8-2	50	0.2	12	1518	7.89	**1504.1**	**−0.92**	7.73	1568.41	3.32	7.14
SCA8-3	50	0.2	12	1467.67	8.07	1474.61	0.47	7.74	1509.61	2.86	7.39
SCA8-4	50	0.2	12	1632.09	8.08	**1629.83**	**−0.14**	7.69	1665.73	2.06	7.44
SCA8-5	50	0.2	12	1476.65	8.16	**1463.19**	**−0.91**	8.07	1515.55	2.63	7.37
SCA8-6	50	0.2	12	1393.23	8.17	**1389.4**	**−0.27**	7.68	1424.84	2.27	7.22
SCA8-7	50	0.2	12	1493.32	8.12	**1488.21**	**−0.34**	7.47	1528.76	2.37	7.32
SCA8-8	50	0.2	12	1549.21	8.36	1551.53	0.15	7.40	1589.07	2.57	7.34
SCA8-9	50	0.2	12	1487.35	8.09	1494.1	0.45	7.25	1527.5	2.70	7.17
CON3-0	50	0.2	5	1416.98	11.61	1427.57	0.75	9.22	1612.38	13.79	8.91
CON3-1	50	0.2	5	1501.88	11.20	1512.28	0.69	9.01	1726.62	14.96	9.15
CON3-2	50	0.2	5	1309.42	11.35	**1304.96**	**−0.34**	9.45	1560.96	19.21	9.93
CON3-3	50	0.2	5	1453.35	11.33	**1430.26**	**−1.59**	9.20	1704	17.25	9.36
CON3-4	50	0.2	5	1492.16	11.40	1499.88	0.52	9.06	1718.11	15.14	9.77
CON3-5	50	0.2	5	1326.91	12.08	1329.7	0.21	9.25	1495.05	12.67	9.18
CON3-6	50	0.2	5	1185.9	11.47	**1175.92**	**−0.84**	9.08	1298.11	9.46	9.18
CON3-7	50	0.2	5	1399.18	11.98	1408.22	0.65	8.88	1605.44	14.74	9.11
CON3-8	50	0.2	5	1383.91	12.01	1405.49	1.56	8.82	1637.1	18.30	8.89
CON3-9	50	0.2	5	1284.01	11.84	**1279.69**	**−0.34**	8.90	1443.85	12.45	9.03
CON8-0	50	0.2	12	1254.23	8.97	1257.37	0.25	7.41	1280.02	2.06	7.49
CON8-1	50	0.2	12	1235.43	9.20	**1228.63**	**−0.55**	7.53	1256.36	1.69	7.65
CON8-2	50	0.2	12	1118.18	9.44	1120.33	0.19	7.53	1162.2	3.94	7.75
CON8-3	50	0.2	12	1235.54	8.77	**1232.04**	**−0.28**	7.52	1267.15	2.56	7.59
CON8-4	50	0.2	12	1244.13	8.67	**1239.87**	**−0.34**	7.52	1277.91	2.72	7.67
CON8-5	50	0.2	12	1132.77	8.66	**1131.36**	**−0.12**	7.52	1169.97	3.28	7.55
CON8-6	50	0.2	12	1019.48	8.64	1023.63	0.41	7.45	1029.04	0.94	7.69
CON8-7	50	0.2	12	1236.91	8.72	**1230.04**	**−0.56**	7.38	1277.07	3.25	7.64
CON8-8	50	0.2	12	1216.22	8.63	**1208.21**	**−0.66**	7.41	1250.85	2.85	7.52
CON8-9	50	0.2	12	1166.35	8.49	**1165**	**−0.12**	7.22	1188.47	1.90	7.47
Average				1394.03	9.92	1397.96	0.25	8.46	1515.82	8.47	8.33

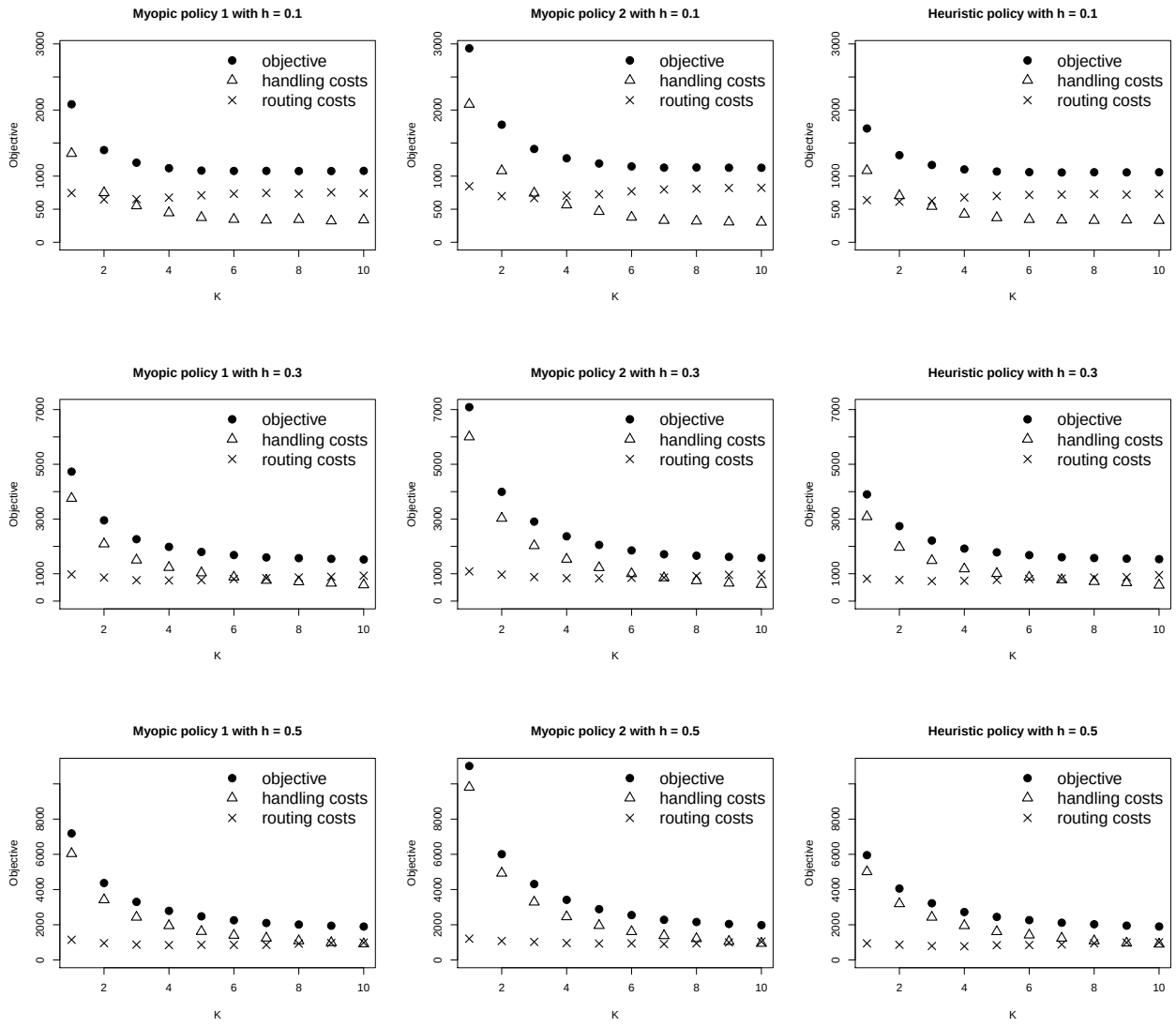


Figure 2: Different cost components for myopic policies 1 and 2 and the heuristic policy for $K = 1, \ldots, 10$ and $h = 0.1, 0.3, 0.5$.

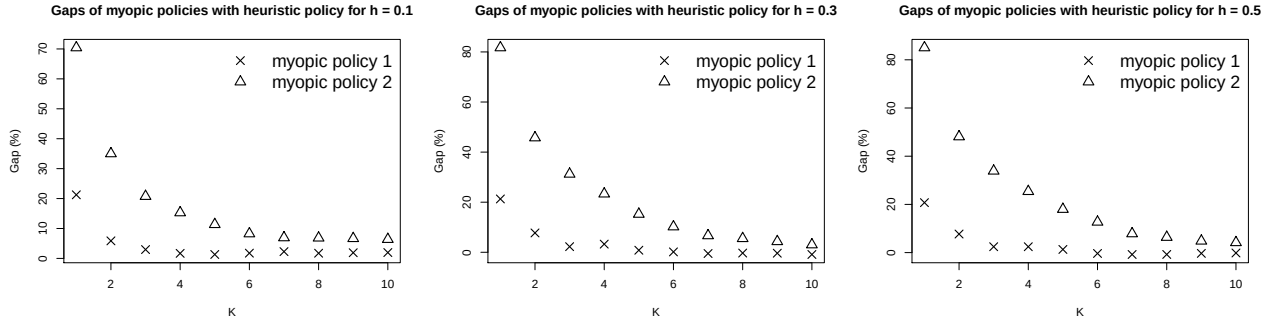


Figure 3: Objective gaps of myopic policies 1 and 2 compared to the heuristic policy for $h = 0.1, 0.3, 0.5$.

6 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 studied a heuristic handling policy which approximates the optimal handling decisions, and we derived two new bounds on the optimal policy which were used to define two myopic policies.

To solve our problem, we propose an adaptive large neighborhood search (ALNS) heuristic in which we embedded all three handling policies, and we implement the mathematical formulation in CPLEX to solve small instances optimally. With the ALNS heuristic, we also solve instances of the VRPSPD and the TSPPD-H. Results on these special cases showed that our ALNS heuristic finds 32 out of 54 best known solutions for the VRPSPD, and the average gaps are 0.14% and 0.84% on two sets of benchmark instances, respectively. For a set of 80 benchmark instances for the TSPPD-H, we find or improve 64 best known solutions with the heuristic handling policy. The optimal handling policy, at the cost of significantly higher calculation times, performs slightly better, improving 69 best known solutions. The average gaps for the two policies are -0.73% and -0.99% respectively. Furthermore, we show that our proposed heuristic finds optimal solutions on instances of up to 15 customers. It also beats 3 out of 7 best integer solutions of CPLEX for instances not solved to optimality within the given time limit, matching the remaining 4.

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 three handling policies are embedded in our ALNS structure. We see that 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, 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 15% lower than the computation time when applying the heuristic policy. 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.

References

[1] M. Avci and S. Topaloglu. An adaptive local search algorithm for vehicle routing problem with simultaneous and mixed pickups and deliveries. *Computers & Industrial Engineering*, 83:15–29, 2015.

[2] M. Battarra, G. Erdoğan, G. Laporte, and D. Vigo. The traveling salesman problem with pickups, deliveries, and handling costs. *Transportation Science*, 44(3):383–399, 2010.

[3] E. Benavent, M. Landete, E. Mota, and G. Tirado. The multiple vehicle pickup and delivery problem with LIFO constraints. *European Journal of Operational Research*, 243(3):752–762, 2015.

[4] M. Cherkesly, G. Desaulniers, and G. Laporte. A population-based metaheuristic for the pickup and delivery problem with time windows and LIFO loading. *Computers & Operations Research*, 62:23–35, 2015.

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

[6] M. Dell'Amico, G. Righini, and M. Salani. A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40(2):235–247, 2006.

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

[8] G. Erdoğan, M. Battarra, G. Laporte, and D. Vigo. Metaheuristics for the traveling salesman problem with pickups, deliveries and handling costs. *Computers & Operations Research*, 39(5): 1074–1086, 2012.

[9] Y. Gajpal and P. Abad. An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup. *Computers & Operations Research*, 36(12):3215–3223, 2009.

[10] M. Gendreau, G. Laporte, and D. Vigo. Heuristics for the traveling salesman problem with pickup and delivery. *Computers & Operations Research*, 26(7):699–714, 1999.

[11] C. B. Kalayci and C. Kaya. 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, 2016.

[12] H. Min. The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A: General*, 23(5):377–386, 1989.

[13] G. Nagy and S. Salhi. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, 162(1):126–141, 2005.

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

[15] S. Ropke and D. Pisinger. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(3):750–775, 2006.

[16] S. Salhi and G. Nagy. A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society*, 50(10):1034–1042, 1999.

[17] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. *Computer*, 1520(Springer):417–431, 1998.

[18] A. Subramanian, E. Uchoa, A.A. Pessoa, and L. S. Ochi. Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery. *Optimization Letters*, 7(7):1569–1581, 2013.

[19] M. Veenstra, K. J. Roodbergen, I. F. A. Vis, and L. C. Coelho. The pickup and delivery traveling salesman problem with handling costs. *European Journal of Operational Research*, 257(1):118–132, 2017.

[20] H. F. Wang and Y. Y. Chen. A genetic algorithm for the simultaneous delivery and pickup problems with time window. *Computers & Industrial Engineering*, 62(1):84–95, 2012.

[21] E. E. Zachariadis and C. T. Kiranoudis. A local search metaheuristic algorithm for the vehicle routing problem with simultaneous pick-ups and deliveries. *Expert Systems with Applications*, 38 (3):2717–2726, 2011.