J Heuristics (2013) 19:129–156
DOI 10.1007/s10732-011-9188-9

A heuristic solution method for node routing based solid waste collection problems

Vera Hemmelmayr · Karl F. Doerner · Richard F. Hartl · Stefan Rath

Received: 15 May 2010 / Accepted: 2 August 2011 / Published online: 7 September 2011

Abstract This paper considers a real world waste collection problem in which glass, metal, plastics, or paper is brought to certain waste collection points by the citizens of a certain region. The collection of this waste from the collection points is therefore a node routing problem. The waste is delivered to special sites, so called intermediate facilities (IF), that are typically not identical with the vehicle depot. Since most waste collection points need not be visited every day, a planning period of several days has to be considered. In this context three related planning problems are considered. First, the periodic vehicle routing problem with intermediate facilities (PVRP-IF) is considered and an exact problem formulation is proposed. A set of benchmark instances is developed and an efficient hybrid solution method based on variable neighborhood search and dynamic programming is presented. Second, in a real world application the PVRP-IF is modified by permitting the return of partly loaded vehicles to the depots and by considering capacity limits at the IF. An average improvement of 25% in the routing cost is obtained compared to the current solution. Finally, a different but related problem, the so called multi-depot vehicle routing problem with inter-depot

V. Hemmelmayr · K.F. Doerner (✉) · R.F. Hartl
Department of Business Administration, University of Vienna, Bruenner Strasse 72, 1210 Vienna, Austria
e-mail: Karl.Doerner@univie.ac.at

V. Hemmelmayr
e-mail: Vera.Hemmelmayr@univie.ac.at

R.F. Hartl
e-mail: Richard.Hartl@univie.ac.at

S. Rath
Department of Statistics and Operations Research, Universitaetsstrasse 5/9, 1010 Vienna, Austria
e-mail: Stefan.Rath@univie.ac.at

K.F. Doerner
Department of Production and Logistics, Johannes Kepler University Linz, Altenberger Strasse 69, 4040 Linz, Austria

routes (MDVRPI) is considered. In this problem class just a single day is considered and the depots can act as an intermediate facility only at the end of a tour. For this problem several instances and benchmark solutions are available. It is shown that the algorithm outperforms all previously published metaheuristics for this problem class and finds the best solutions for all available benchmark instances.

Keywords Reverse logistics · Metaheuristics · Solid waste collection · Periodic vehicle routing problems · Multi-depot vehicle routing problems

1 Introduction

Waste collection is an interesting class of rich vehicle routing problems that is of high practical relevance. Many different optimization problems have been proposed in this context. Basically, there are two broad classes of models. In the collection of normal residual waste, the citizens usually deposit their waste in special bags or bins in front of their house on the day of collection. Hence, collection is done along the streets. This type of waste collection problem is therefore modeled as some kind of arc routing problem. This case is not considered in this paper. We are concerned with a second class of waste collection models, in which more valuable material such as glass, metal, plastics, or paper is brought to waste collection points by the citizens of a certain region. In this case the collection of the waste from theses collection problems is a node routing problem. Typically, most waste collection points need not be visited every day, and therefore a planning period of several days has to be considered. Due to social regulations on working hours, each tour is subject to a maximum duration constraint for every day. The basic model here is therefore the periodic vehicle routing problem (PVRP) proposed by Cordeau et al. (1998). In waste collection, however, the waste is delivered to special sites, so called intermediate facilities (IF), that are typically not the same as the vehicle depot. A more appropriate model class is therefore the Periodic Vehicle Routing Problem with Intermediate Facilities (PVRP-IF). This problem was proposed by Angelelli and Speranza (2002b) without providing a formal model.

We present a MIP formulation for the PVRP-IF, that extends the formulation for the PVRP, that was provided in Cordeau et al. (1998). Furthermore, a set of benchmark instances is proposed, that essentially extends the MDVRPI instances by Crevier et al. (2007) with the visit day combinations of the PVRP instances by Cordeau et al. (1998). This is possible because the set of customers is the same for both data sets. We propose an efficient hybrid solution method based on variable neighborhood search and dynamic programming. More precisely, the basic mechanism for the PVRP is the variable neighborhood search (VNS) while the insertion of the intermediate facilities is done by dynamic programming. Various design decisions concerning the set of neighborhoods and the method of inserting the intermediate facilities in connection with local search are investigated.

The second contribution of this paper is a real world application, in which the PVRP-IF is modified by certain additional constraints requested by our industrial partners. The first modification concerns the collection of waste types such as glass,

that are not inflammable and that do not cause any putrid smell. In this case some municipalities do not require the vehicles to visit an intermediate facility at the end of the day. Rather, the vehicles can return to the depots partly loaded and hence start their trip partly loaded in the next morning. Using the set of PVRP-IF instances mentioned above, we show that this option permits some cost savings that are, however, on average not very significant (less than 1%). In our real world case we also have to consider capacity limits at the depots. Due to long range contracts between the collection company and the operators of the intermediate facilities, the quantities that must be delivered to each intermediate facility are roughly fixed, with certain tolerances. We perform an evaluation of this constraint using the benchmark instances for the PVRP-IF. It turns out that an even distribution of the quantities leads to average cost increases of about 6% compared to the unrestricted case. If the distribution is biased, i.e. if one facility is large and the others small, then the average cost increase is about 13% which is more than twice as large. Our algorithm is also compared to a few real world instances for which manual solutions are available. Compared to this manual solution, an average reduction of 25% in the routing cost is obtained. A further minor modification in the real world application refers to tour length restrictions that may vary from day to day and from vehicle to vehicle. This is because the vehicles do not have different compartments, and each tour is dedicated to the collection of a single type of waste. Hence, we can decompose the problem and solve it for each type of waste separately. Each type of waste is collected by a separate vehicle on a separate tour. However, the same vehicle can collect different types of waste on different days, or even on the same day, e.g., glass in the morning and paper in the afternoon. The availability of vehicles for a given type of waste can therefore be different for each vehicle, e.g., 8 hours or 4 hours. Currently this is exogenously given.

Finally, this paper also solves a different but related problem, the so called multi-depot vehicle routing problem with inter-depot routes (MDVRPI). In this problem class, just a single day is considered and the depots can act as an intermediate facility only at the end of a tour. The term multi-depot refers to the intermediate facilities. The MDVRPI was first introduced by Crevier et al. (2007): They also presented a tabu search (TS) algorithm to generate various different routes, which were then combined by a set partitioning approach. The same problem was also solved recently by Tarantilis et al. (2008). They proposed a three-step algorithmic framework, in which TS is used as a local search procedure within the VNS, and then guided local search (GLS) is used in a post-optimization phase to remove low-quality features from the solution. For this problem three different sets of benchmark instances are available. We show that our hybrid VNS algorithm (developed for the PVRP-IF) is also efficient for the MDVRPI after minor modifications. It is shown that the algorithm outperforms all previously published metaheuristics for this problem class and finds the best solutions for all available benchmark instances.

The remainder of this paper is organized as follows. In Sect. 2, we introduce the three models, i.e. the PVRP-IF, the real world variant, and the MDVRPI. Section 3 presents a survey on related work on waste collection. In Sect. 4, first the basic concept of the solution methods is introduced for the PVRP-IF, and then the modifications for the other two problem classes are presented. The numerical results for all problems can be found in Sect. 5 and Sect. 6 concludes the paper.

Since we consider three different types of problems, we decided to take the following sequence for Sects. 2, 4 and 5. We will first describe the PVRP-IF, then the MDVRPI and finally the real world extensions of the PVRP-IF.

2 Problem description

2.1 The PVRP with intermediate facilities (PVRP-IF)

The basic model considered in this paper is the periodic vehicle routing problem with intermediate facilities (PVRP-IF). The periodic vehicle routing problem (PVRP) extends the vehicle routing problem to a t day planning horizon. A set of customers (in our case the waste collection points) requires regular visits during the planning horizon. The exact days of the visits are not given, but every customer has a certain visit frequency e_i that must be respected. Depending on this frequency, for every customer a set C_i of allowable visit combinations is precalculated. For example if 2 visits are required, $e_i = 2$, we have a given set of periodic combinations $C_i = \{\{1, 4\}, \{2, 5\}, \{3, 6\}\}$, i.e. these visits can take place either on days 1 and 4, or on days 2 and 5, or on days 3 and 6. The demand delivered to the customers is the daily demand multiplied by the number of days in between two consecutive visits. Since we only consider periodic combinations, which means that there is always the same number of days in between two consecutive visits, the demand delivered to a given customer does not depend on the visit day combination chosen. A fixed, homogeneous fleet of vehicles is given for every day.

The vehicles are placed at a single depot from where they start their trip and where they go back to at the end of the day. There is a set of intermediate facilities (IF) where vehicles unload the waste. After visiting an IF, the empty vehicles can continue their trip to collect more waste. Note that only at the IF the vehicle can be emptied and that the vehicle must return empty to the depot. The objective is to minimize total time traveled over the planning period. There is a capacity and a tour length restriction that have to be respected. Figure 1 shows an example solution to a VRP- IF (for one day), where the triangles represent the IFs and the large dot represents the depot. The vehicles start from the depot, collect waste, then drop it at one of the IF and continue their trip afterwards. In the example in the picture they have to unload before they are going back to the depot.

The PVRP-IF was introduced by Angelelli and Speranza (2002b) without providing a formal model.

We now propose a MIP formulation for the PVRP-IF, that extends the formulation for the PVRP, that was provided in Cordeau et al. (1998). A description of the variables and parameters used is given in Table 1. A complete graph $G = (V, A)$ is given, where $V = 0, 1, \ldots, n + s$ is the set of vertices and A is the set of arcs. Vertex 0 represents the depot, vertices $1, \ldots, n$ represent the customers from where the waste is collected and vertices $n + 1, \ldots, n + s$ represent the intermediate facilities where the waste is dumped. Furthermore, a nonnegative cost c_{ij}, representing the travel time or

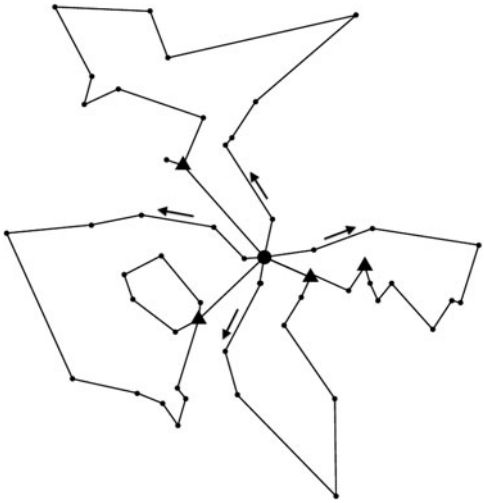

Fig. 1 PVRP-IF solution to a 1-day problem

Table 1 Notation

Variables	
x_{ijkl}	a 0–1 variable indicating whether vehicle k visits node j immediately after node i on day l
y_{ir}	a 0–1 variable indicating whether visit combination $r \in C_i$ is assigned to customer i
z_{pkl}	a 0–1 variable indicating whether vehicle k unloads at intermediate facility p on day l
f_{ijkl}	a real variable indicating the load of vehicle k on arc ij on day l (only used for real world extensions)
Parameters	
c_{ij}	travel time from node i to node j
a_{rl}	binary constant equal to 1 if day l belongs to visit combination r, 0 otherwise
Q	capacity of a vehicle
D	maximum permitted travel time per vehicle
q_i	amount of waste collected from customer i
d_i	service duration at customer i
t	planning period
n	number of customers
m	number of vehicles
s	number of intermediate facilities
C_i	set of allowable visit day combinations for customer i
e_i	given service frequency for customer i
I	set of intermediate facilities
P_p	capacity limit for IF p (for real world extensions)

cost, is given for each arc (i, j). Each customer i $(i = 1, \ldots, n)$ has a nonnegative demand d_i, that will be delivered from the depot by one of m identical vehicles.

$$\min \sum_{i=0}^{n+s} \sum_{j=0}^{n+s} \sum_{k=1}^{m} \sum_{l=1}^{t} c_{ij} x_{ijkl}$$

subject to

$$\sum_{r \in C_i} y_{ir} = 1 \quad \forall i = 1, \ldots, n \tag{1}$$

$$\sum_{j=0}^{n+s} \sum_{k=1}^{m} x_{ijkl} - \sum_{r \in C_i} a_{rl} y_{ir} = 0 \quad \forall i = 1, \ldots, n, l = 1, \ldots, t \tag{2}$$

$$\sum_{i=0}^{n+s} x_{ihkl} - \sum_{j=0}^{n+s} x_{hjkl} = 0 \quad \forall h = 0, \ldots, n+s, k = 1, \ldots, m, l = 1, \ldots, t \tag{3}$$

$$\sum_{j=1}^{n} x_{0jkl} \leq 1 \quad \forall k = 1, \ldots, m, l = 1, \ldots, t \tag{4}$$

$$\sum_{i=0}^{n+s} \sum_{j=0}^{n+s} (c_{ij} + d_i) x_{ijkl} \leq D \quad \forall k = 1, \ldots, m, l = 1, \ldots, t \tag{5}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ijkl} \leq |S| - r(S) \quad \forall k = 1, \ldots, m, l = 1, \ldots, t, \forall S \subseteq V \setminus I, |S| \geq 2 \tag{6}$$

$$\sum_{i \in S} \sum_{j \notin S} x_{ijkl} \geq z_{pkl} \quad \forall k = 1, \ldots, m, l = 1, \ldots, t, p = n+1, \ldots, n+s,$$
$$\forall S \subseteq V, 0 \in S, p \notin S \tag{7}$$

$$x_{pjkl} \leq z_{pkl} \quad \forall p = n+1, \ldots, n+s, k = 1, \ldots, m, l = 1, \ldots, t, j = 0, \ldots, n+s \tag{8}$$

$$x_{i0kl} = 0 \quad \forall i = 1, \ldots, n, k = 1, \ldots, m, l = 1, \ldots, t \tag{9}$$

$$x_{ijkl} \in \{0, 1\} \quad \forall i = 0, \ldots, n+s, j = 0, \ldots, n+s, k = 1, \ldots, m, l = 1, \ldots, t \tag{10}$$

$$y_{ir} \in \{0, 1\} \quad \forall i = 1, \ldots, n, r \in C_i \tag{11}$$

$$z_{pkl} \in \{0, 1\} \quad \forall p = n+1, \ldots, n+s, k = 1, \ldots, m, l = 1, \ldots, t \tag{12}$$

The objective is to minimize the total travel cost, which is proportional to the total travel time over all days. Constraints 1 state that every customer must be assigned one feasible visit combination. Constraints 2 ensure that a customer is visited exactly on the days specified by the assigned combination. Constraints 3 guarantee that if a vehicle visits a customer on one day, it also leaves that customer on that day. Constraints 4 say that every vehicle can be used at most once every day and tour length restrictions are ensured by constraints 5. Constraints 6 are capacity and subtour elimination constraints, where $r(S)$ is the minimum number of trips needed to serve S in order to meet the capacity constraints. Constraints 7 ensure the connectivity with the IF, so that every tour contains the depot plus at least one IF. Constraints 8 state that only when we use IF p in vehicle k on day l, it can be used for unloading. Constraints 9 ensure that before we go back to the depot, we unload at an IF. Finally, constraints 10, 11 and 12 enforce binary values on decision variables. Clearly the above formulation

can be extended to a formulation with heterogenous vehicles similar to the one in Cordeau et al. (1998).

2.2 The MDVRPI

The MDVRPI was studied in Crevier et al. (2007) and Tarantilis et al. (2008). There are two differences to the PVRP-IF. The first is that it only considers a single day. The second difference is that in the MDVRPI the depot can also act as an IF, but only at the end of the day. This means that the vehicles do not have to visit an IF at the end of the day in order to unload, but they can go back directly to the depot and unload there.

2.3 Additional real world constraints

In addition to the above classical PVRP-IF we also consider a model variant with two additional real world constraints. One constraint concerns the assignment of waste to the IFs. There are two possibilities: The first option is that waste can be delivered to any IF, as assumed in the above model. The second option is that a specific part of waste has to go to each IF. For example if we have three IFs, 20% of the total waste has to go to the first IF, 10% to the second and 70% to the third.

We extend our model formulation above to deal with that case. We introduce new variables f_{ijkl} that represent the amount of goods transported from node i to node j on day l by vehicle k. As additional data we need the maximum amount of waste P_p that can be brought to an intermediate facility. Constraints 6 can be omitted.

$$f_{ijkl} \leq Q x_{ijkl} \quad \forall i = 0, \ldots, n+s, j = 0, \ldots, n+s, k = 1, \ldots, m, l = 1, \ldots, t \tag{13}$$

$$\sum_{i=0}^{n+s} f_{ijkl} + q_j \sum_{i=0}^{n+s} x_{ijkl} = \sum_{i=0}^{n+s} f_{jikl} \quad \forall j = 1, \ldots, n, k = 1, \ldots, m, l = 1, \ldots, t \tag{14}$$

$$f_{pjkl} = 0 \quad \forall p = n+1, \ldots, n+s, j = 0, \ldots, n+s, k = 1, \ldots, m, l = 1, \ldots, t \tag{15}$$

$$f_{0jkl} = 0 \quad \forall j = 0, \ldots, n+s, k = 1, \ldots, m, l = 1, \ldots, t \tag{16}$$

$$\sum_{i=0}^{n+s} \sum_{k=0}^{m} \sum_{l=1}^{t} f_{ipkl} \leq P_p \quad \forall p = n+1, \ldots, s \tag{17}$$

$$f_{ijkl} \geq 0 \quad \forall i = 0, \ldots, n+s, j = 0, \ldots, n+s, k = 1, \ldots, m, l = 1, \ldots, t \tag{18}$$

Constraints 13 ensure that the capacity constraint of the vehicle is respected and they also link f_{ijkl} with x_{ijkl}. Constraints 14 guarantee that the outbound flow of customer j equals the inbound flow plus the waste collected from customer j and therefore forbid subtours. Constraints 15 guarantee that the flow out of every IF is zero, while constraints 16 state that the initial flow out of the depot is zero. Finally, constraints 17 ensure that the capacity limit at the IFs is respected and constraints 18 are

the non-negativity constraints. Note that the model without constraints 17 is equivalent to the previous PVRP-IF model.

A second modification suggested by our industrial partner was to relax the constraint that the vehicles have to return empty to the depot.

The model is modified accordingly. We omit constraints 9 because now it is allowed to go to the depot directly after a customer. Furthermore, we replace constraints 16 by 19 which state that the initial flow out of the depot has to be only zero at the first day of the planning period. We also introduce constraints 20 that ensure that the vehicle leaves on day $l+1$ with the load that it carried when it returned to the depot on day l. Finally, constraints 21 ensure that there is no waste delivered to the depot on the last day.

$$f_{0jk0} = 0 \quad \forall j = 0, \ldots, n+s, k = 1, \ldots, m \tag{19}$$

$$\sum_{j=0}^{n+s} f_{0jkl} = \sum_{j=0}^{n+s} f_{j0kl-1} \quad \forall k = 1, \ldots, m, l = 2, \ldots, t \tag{20}$$

$$f_{i0kt} = 0 \quad \forall i = 0, \ldots, n+s, k = 1, \ldots, m \tag{21}$$

Moreover, the tour length varies from day to day, depending on the availability of the vehicle for the type of waste considered. This means that the fleet is not homogeneous anymore. Therefore, the maximum permitted travel time per vehicle, D, is changed to D_{kl} in constraints 5, since it depends on the vehicle and on the day.

3 Related work

Angelelli and Speranza (2002b) were the first to our knowledge that have considered the PVRP-IF. They have developed a TS method for solving the problem. This problem setting is closely related to our problem setting. However, unfortunately the instances are not available, therefore we cannot compare the performance of our algorithm to theirs. In Angelelli and Speranza (2002a) they extend their algorithm to measure the operating cost of three different waste-collection systems.

The PVRP has been studied by Cordeau et al. (1998) and Alegre et al. (2007).

Crevier et al. (2007) were the first to study the MDVRPI. A TS algorithm is used to generate different routes, which are then combined by a set partitioning approach. The TS generates MDVRP, VRP and inter-depot routes. In the end a post-optimization phase is performed. In Tarantilis et al. (2008) an algorithm is proposed that is able to improve the results by Crevier et al. (2007) on the benchmark instances. They propose a three-step algorithmic framework, which consists of a VNS, a TS and a GLS. The TS is used within the VNS as a local search procedure and the GLS is used for postoptimization to remove low quality features from the solution.

An early paper on waste collection was published by Beltrami and Bodin (1974) which at the same time was the first description of a PVRP. In a case study on waste collection, Golden et al. (2001) distinguish between commercial collection, residen-

tial collection and roll-on-roll-off problems. While the first ones involve the collection of containers at commercial locations and are therefore node routing problems, residential collection involves collecting household refuse along a street network and are therefore arc routing problems. Roll-on-roll-off problems occur when large containers or trailers have to be picked up at construction sites and then transported and unloaded. In our case however residential collection is also modeled as a node routing problem, because waste is not collected along the street network, but from certain waste collection points.

Tung and Pinnoi (2000) study a vehicle routing and scheduling problem to solve a waste collection problem in Hanoi, Vietnam. The problem is composed of several stages. Waste is first picked up by handcarts and delivered to gather sites. Then the waste has to be unloaded from the handcarts and loaded onto the tipper. The handcarts start their next round, but they are only allowed to come back after a certain period of time, the minimum time required between pick-ups. From there vehicles pick up the waste and deliver it to a landfill. There are time windows at the gather sites. At the moment the vehicles visit the gather sites in the same sequence all the time. The authors study the improvement of allowing flexible routing of the vehicles, allowing them to visit sites in any sequence, while leaving the handcart operations out of scope. As a solution method they use a modified construction and improvement heuristic.

In Bodin et al. (2000) the roll-on-roll-off VRP is studied, where tractors move large trailers between locations and a disposal facility. The tractors can only move one trailer at a time. They propose a mathematical programming formulation, two lower bounds and four heuristic algorithms.

Baptista et al. (2002) study the collection of recycling papers in the Almada municipality in Portugal. Their problem is modeled as a PVRP. However, unlike in the classical PVRP the visit frequency is not fixed, but a decision of the model. The problem was solved by a heuristic based on the method by Christofides and Beasley (1984). First initial frequencies and visit day combinations are assigned followed by an interchange procedure that tries to find better visit day combinations.

Teixeira et al. (2004) also solve a PVRP in the waste collection context. They deal with a long planning period and incorporate the collection of different types of waste. A three-phase heuristic was developed. First the problem is partitioned into geographic zones. Then for each zone and for each work shift the decision on which type of waste to collect is taken, where they try to distribute shifts of the same type regularly in time. Finally, the sites to collect are defined and the according routing problem is solved.

Eisenstein and Iyer (1997) study the scheduling of garbage trucks in the city of Chicago. In the current system a truck collects waste for a fixed amount of time and visits the disposal site afterwards, even though different city blocks usually produce different amounts of waste. In their paper they develop a more flexible system by using a dynamic solution method based on Markov decision process.

Archetti and Speranza (2004) consider the so-called 1-skip collection problem, which is a generalization of the roll-on-roll-off VRP. A fleet of vehicles must transport skips of waste, one at a time, from its location to one of different disposal sites, depending on the kind of waste contained in the skip. A number of real world constraints apply like time windows, shift-time, different kinds of skips, a limit on the

number of drivers and priorities. They propose a heuristic that first constructs a feasible solution based on the nearest neighbor heuristic and improves it afterwards.

While the works presented above dealt with node routing problems, the following waste collection problems are modeled as arc routing problems.

Mourão and Almeida (2000) study a waste collection problem in a quarter of Lisbon by solving a capacitated arc routing problem (CARP) with side constraints. More precisely, the vehicle collects waste, delivers it to a recycling facility, then collects waste again and so on. This problem is similar to the problem that we study, except that they solve an arc routing problem while we solve a node routing problem. They have developed two lower-bounding methods and a three-phase heuristic. The same problem is studied in Mourão and Amado (2005), where they present a new heuristic method.

In Lacomme et al. (2005) a waste collection problem is modelled as a periodic capacitated arc routing problem on a mixed graph. Moreover, the demand of an arc depends on the period or on the date of the previous visit. As a solution method a memetic algorithm is proposed.

Kulcar (1996) provide a case study about solid waste collection in Brussels. The problem they consider is modeled by grouping the arcs making up the streets to a set of points. Moreover, they investigate transportation by vehicle, rail and canal.

Prins and Bouchenoua (2005) define a more generic model, the node, edge and arc routing problem (NEARP), that generalizes the VRP and the CARP. They problem is motivated by applications in waste collection. The NEARP tackles mixed graphs with required nodes, edges and arcs. They develop a memetic algorithm and show its competitiveness on standard benchmark instances of the VRP and the CARP.

4 Solution techniques

We developed a hybrid solution method that is composed of a VNS using an exact method to insert the IFs. VNS is based on moving from one solution to the other by first selecting a random solution from the current neighborhood and then moving to the local optimum by using a local search phase. Then it is necessary to decide whether to move to the new solution or not. In the basic VNS, only improving solutions are accepted, but often it is necessary to have more sophisticated acceptance decisions in order to not get trapped in a local optimum. Figure 1 shows a pseudocode for VNS. The VNS algorithm starts from an initial solution. In every iteration there is the so-called shaking phase where a solution is selected randomly from the current neighborhood, followed by a local search phase and the acceptance decision step. The algorithm stops when the stopping condition is met which can be the number of iterations, a number of iterations without improvement or the CPU time. Several neighborhoods are used in a VNS algorithm and they are ideally nested. That means that a neighborhood with a higher index should be a superset of the neighborhood with a lower index. In highly complex real world problems this is typically not possible. Then a neighborhood with a higher index should at least contain a larger number of solutions, although other implementations can also be found in the literature. When a new solution is accepted in an iteration the first neighborhood is used again, other-

wise the neighborhood index is increased by one. See also Hansen and Mladenovic (2001) for a more detailed description of VNS.

Algorithm 1 Basic steps of the VNS

$s* \leftarrow InitialSolution$, choose N_κ $(\kappa = 1, \ldots, \kappa_{max})$
repeat
  $\kappa \leftarrow 1$
  repeat
    $s' \leftarrow RandomSolution(N_\kappa(s*))$
    $s*' \leftarrow LocalSearch(s')$
    if $AcceptanceDecision(s*', s*)$ **then**
      $s* \leftarrow s*'$
      $\kappa \leftarrow 1$
    else
      $\kappa \leftarrow \kappa + 1$
    end if
  until $\kappa > \kappa_{max}$
until stopping condition is met

In the next sections we explain the design decisions of our implemented VNS.

4.1 Initial solution

To build an initial solution we assign visit combinations randomly to the customers from the set of feasible visit combinations. Then VRPs are constructed for every day based on the savings heuristic of Clarke and Wright (1964). The tour length constraint is respected. Afterwards the intermediate facilities are inserted with the dynamic programming (DP) procedure explained below.

4.2 Shaking

The different neighborhood operators are described in the following.

The operator *move* inserts a segment of customers from one route into a different route. The *cross* operator exchanges two segments of customers of different routes. The orientation of the segment(s) and of the route(s) is preserved by the *move* and *cross* operators. The position and the length of each segment are all chosen randomly where the maximum sequence length depends on the current neighborhood. We have two versions of each operator, one that moves or swaps segments between two different routes and one that moves or swaps segments between two different trips of the same route. Note that by trip we refer to a leg between the depot and an IF or between two IFs. In an *intertrip move*, a segment of one trip is chosen randomly to be inserted in another trip of the same route. The segment is inserted in another trip, yielding the lowest cost and only moves that are feasible with respect to capacity can be accepted. In the *intertrip cross*, two segments of randomly chosen routes are

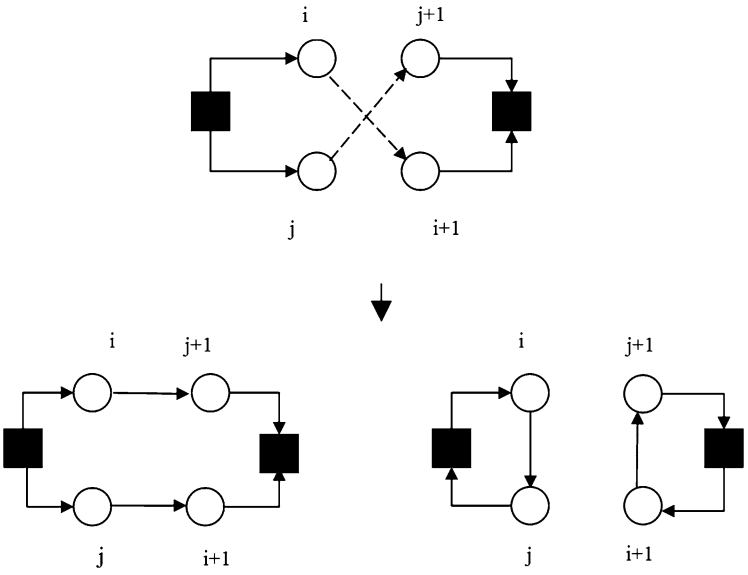

Fig. 2 The *2-opt** operator

exchanged. Here we just swap segments of random trips, because finding the best feasible insertion position would be too time consuming. Both *move* and *cross* are also performed inter-tour only, i.e. they do not move or exchange segments of one route, but between different routes. In the inter-tour version of *move* the position where the segment is inserted is chosen randomly. For a classification of *move* and *cross* see Kindervater and Savelsbergh (1997).

*2-opt** (Potvin and Rousseau 1995) is performed on two randomly chosen routes. Edges $(i, i+1)$ and $(j, j+1)$ are replaced by edges $(i, j+1)$ and $(j, i+1)$. Therefore, the customers located after customer i in the first route are reinserted to be served after customer j in the second route and the customers after j are inserted after customer i. Additional to the standard *2-opt**, we have also used a second possibility to connect the routes. More precisely, the second type of move replaces edges $(i, i+1)$ and $(j, j+1)$ by edges (i, j) and $(i+1, j+1)$ and reverses the direction of visit between j and the depot as well as between $i+1$ and the depot. An illustration of *2-opt** can be seen in Fig. 2. A move can only be accepted if it is feasible with respect to the tour length constraint.

The operator *change combination* assigns a new random visit combination to one or several customers. Then the customers are deleted from their previous positions and are inserted in the days of the new visit day combination with best insertion. While all the other operators perform on a route level only, i.e. they only change the VRP of a given day, this operator affects more than one day. Therefore, it is not used in the MDVRPI.

The parameter for neighborhood size for *move* and *cross* is given by the maximum number of customers in the route segments used within the operators, for *change combination* by the maximum number of customers for which the visit day combination is changed as shown in Tables 2 and 3. For *2-opt** we always use two randomly

Table 2 Set of Neighborhood Structures with $\kappa_{max} = 18$ for the PVRP-IF and the real world extensions

κ	Operator	min. customers	max. customers
1	*change combination*	1	1
2	*change combination*	1	2
3	*change combination*	1	3
4	*change combination*	1	4
5	*change combination*	1	5
6	*change combination*	1	6
		min. segment length	max. segment length
7	*intertrip move*	1	$\min(1, n)$
8	*intertrip move*	1	$\min(2, n)$
9	*intertrip move*	1	$\min(3, n)$
10	*intertrip cross*	1	$\min(1, n)$
11	*intertrip cross*	1	$\min(2, n)$
12	*intertrip cross*	1	$\min(3, n)$
13	*intertour move*	1	$\min(1, n)$
14	*intertour move*	1	$\min(2, n)$
15	*intertour move*	1	$\min(3, n)$
16	*intertour cross*	1	$\min(1, n)$
17	*intertour cross*	1	$\min(2, n)$
18	*intertour cross*	1	$\min(3, n)$

Table 3 Set of neighborhood structures with $\kappa_{max} = 13$ for the MDVRPI

κ	Operator	min. segment length	max. segment length
1	*intertrip move*	1	$\min(1, n)$
2	*intertrip move*	1	$\min(2, n)$
3	*intertrip move*	1	$\min(3, n)$
4	*intertrip cross*	1	$\min(1, n)$
5	*intertrip cross*	1	$\min(2, n)$
6	*intertrip cross*	1	$\min(3, n)$
7	*2-opt** for 2 random routes		
		min. segment length	max. segment length
8	*intertour move*	1	$\min(1, n)$
9	*intertour move*	1	$\min(2, n)$
10	*intertour move*	1	$\min(3, n)$
11	*intertour cross*	1	$\min(1, n)$
12	*intertour cross*	1	$\min(2, n)$
13	*intertour cross*	1	$\min(3, n)$

chosen routes. In each neighborhood we choose the number of customers or the segment length randomly between the minimum and the maximum value. Hence our choice of neighborhoods is biased toward smaller changes to focus the search rather close to the incumbent solution. The neighborhoods are ordered in an ascending way, i.e. we first start with the neighborhoods that only perturb a small part of our solution and increase this step by step.

4.3 Local search

For the local search we developed different components that can be combined. They are all based on *2-opt*, *3-opt*, on *DP* to insert the IFs exactly and on a greedy insertion procedure to insert the IFs.

The *2-opt* operator was introduced by Croes (1958). Two edges of a tour are deleted and combined in a different way connecting the first customers of those edges and the second customers of those edge, which also means that the sequence between those edges has to be inverted. The local search restarts immediately after an improving move was found.

We use *2-opt* in a way that treats IFs as normal customers and only moves that are feasible with respect to capacity are considered. We refer to it as 2-opt-only-feasible (*2-opt-of*). Therefore, a move creating a tour that does not fulfil the capacity restriction cannot be accepted. More precisely, moving too many customers between two IF visits leads to a violation of the capacity constraint.

To optimize the sequence of customer visits within a trip, we use *3-opt* and refer to it as *3-opt intratrip*. *3-opt* was introduced by Lin (1965). This operator tries all shifts of all subsequences to different positions in the same route. More precisely, three edges are deleted and replaced by three other edges. In our algorithm *3-opt* without sequence inversion, also often denoted as *3-opt**, is used.

In order to optimize the position of the IFs, we developed the following procedure. First we remove all the IFs. Then we have two possibilities of reinserting them. The first one is an exact procedure for inserting the IFs with *DP*. The second one is a greedy procedure that only inserts an IF whenever the capacity is about to be exceeded. This procedure is only used for the real world extensions.

For the exact procedure, we construct a directed graph that contains the customer nodes and the depot as the starting and the ending node. An arc represents a trip between two intermediate facility visits. More precisely, an arc ending in a node means that an IF is visited before that customer and therefore this customer is the first customer of the new trip. When we insert an IF we always choose the one that is closest to the two nodes between which it is inserted. Moreover, we only consider arcs that are feasible with respect to capacity. By calculating the shortest path we get the optimal insertion of IFs for a given customer sequence. This method is based on the one that was first described by Beasley (1983) for a route-first, cluster-second method for the VRP and it was also used in a genetic algorithm by Prins (2004) for the VRP.

Figure 3 shows an example with four nodes. The first customer and the depot can be merged to one node, since there is no need to visit an IF before node 1 is visited. The arcs show all the possible and feasible trips that can be enumerated. For example

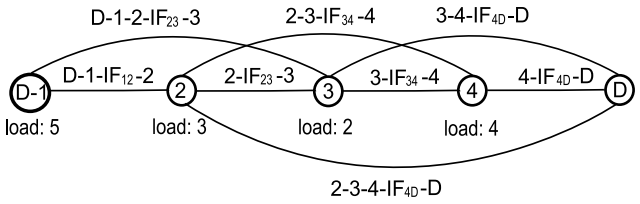

Fig. 3 Insert the intermediate facilities with dynamic programming. All feasible arcs for a truck capacity of 9 are enumerated

arc D-1-IF-2 means that we start from the depot, visit customer 1, then unload at the intermediate facility and proceed to customer 2 with an empty vehicle. IF_{12} is the intermediate facility with minimum insertion cost between 1 and 2. We assume that we have a truck capacity of 9, then we cannot enumerate trip D-1-2-3-IF and D-1-2-3-4-IF, because the load of 10 or 13 respectively would exceed the capacity. The shortest path from D-1 to D in the graph represents the optimal insertion of the IFs. Finding the shortest path is done by *DP*.

The general local search procedure works as follows. First the IFs are removed and are then reinserted with the *DP procedure*. Afterwards *2-opt-of* is used. This local search procedure is used for the PVRP-IF and the MDVRP. However, when real world constraints apply, it is necessary to adapt it, as described in Sect. 4.7.

4.4 Acceptance decision

The stopping condition in our implementation is the number of iterations. For solution acceptance we use a condition inspired by Simulated Annealing (SA) (Gelatt and Vecchi 1983). More precisely, solutions that yield a better objective function value are always accepted and inferior solutions are accepted with a probability $e^{\frac{-(f(x')-f(x))}{T}}$, where f is the objective function, x is the incumbent solution and x' is the new solution obtained after shaking and local search. So the acceptance of inferior solutions depends on the difference between the costs of the new solution and the incumbent solution and on a given temperature T. We decrease T linearly in η/k stages during the search process, where η represents the total number of iterations executed. Thus, every k iterations T is decreased by an amount $\frac{Tk}{\eta}$. We found that this linear annealing scheme works slightly better than the classical exponential one.

In feasibility in the tour length or capacity constraint is penalized in the objective function with a constant value.

4.5 Algorithm for the PVRP-IF

The algorithm for the PVRP-IF uses the neighborhoods described in Table 2. The local search procedure consists of the following steps. First we remove the IFs, then we insert them again with the *DP procedure* and afterwards we apply the *2-opt-of* operator.

4.6 Algorithm for the MDVRPI

The neighborhoods used for the MDVRPI are described in Table 3. The algorithm is similar to the one for the PVRP-IF, except for two neighborhood operators. The operator *change combination* is not used, since the planning period is just one day. Moreover, *2-opt** proved to be very effective for MDVRPI instances where the average number of customers per tour is very large.

4.7 Solving the real-world problem

As mentioned above, in the real world problem that we have studied, three modification of the PVRP-IF are considered. First, we have investigated the case where it is not necessary to unload at the end of the day. The vehicles can go back to the depot partly loaded and perform the unloading operation on the next day.

We do not perform *2-opt-of* to improve the tours, but instead we use *2-opt* before the IFs are inserted. For reinserting the IF, we use the *DP procedure*. In this case we construct the graph for the *DP procedure* by adding tours performed by the same vehicle until it is necessary to unload, which is at the end of the planning period. More precisely, we consider a giant tour consisting of the sequence of routes of a given vehicle over the whole planning horizon. This means that the depot node is also contained in the tour more often than only at the start and the end node.

Second, we consider the assignment of waste quantities to IFs. Changes in the algorithm were necessary to deal with that case. Since we have capacity constraints, we cannot use the *DP procedure* any longer in the way we used to. Therefore, the IFs are not inserted with the *DP procedure* anymore, but with a greedy procedure that considers the capacity limit at the IFs. When inserting an IF we always choose the one that is closest between the two customers where it is inserted. Once the capacity limit of one IF is exceeded, we close it and do not insert it any more. Afterwards *3-opt-intertrip* is performed to reoptimize the visit sequence of the customers. This is the only algorithm where we use *3-opt-intertrip*. The reason is that we do not want to destroy the assignment of waste to the intermediate facilities due to the capacity restrictions. Therefore, we only perform the local search for a small part of the solution, i.e. the part between two intermediate facilities.

Finally, the tour length may vary from vehicle to vehicle. This is solved by simply penalizing infeasible solutions in the objective function.

To sum it up, we use different components for the different problems. The algorithms for the PVRP-IF and the MDVRPI are almost the same. The only differences are the two following shaking operators. The operator *change combinations* is only used for the PVRP-IF. The MDVRPI is not a periodic problem and therefore this operator is not necessary. The operator *2-opt** proved to be very effective for the MDVRPI instances, while it did not bring an improvent for the PVRP-IF. Therefore, we decided to only use it for the MDVRPI.

The real world problems on the other hand have the same shaking phase as the PVRP-IF. However, due to the specific problem characteristics it was necessary to change the local search phase. The local search phase had to be changed, because with the current local search feasibility was not guaranteed anymore. Moreover, the idea

of our VNS implementations is that local search is only performed for a small part of the solution like a route in the MDVRPI and PVRP-IF implementations. Therefore, in the case of capacities at the IFs we decided to take a local search that does not destroy the current assignment of waste to an IF, but only optimizes the routing of each trip. This is why we chose *3-opt-intertrip*. For the case where the vehicle does not have to unload before going back to the depot, we chose to use *2-opt* for each route before the IFs are inserted, instead of using *2-opt-of* after they are inserted. In this case *2-opt-of* cannot be used anymore as a local search that only optimizes the routing of a given vehicle on a given day because feasibility has to be ensured over the whole planning period. However, when we add all the routes of a vehicle over the planning period and perform *2-opt* on it then it could happen that a customer is moved to another day. If this customer has a frequency that is higher than one, this would result in the necessity of moving all the customer visits to a new combination and would result in a much more complex and time consuming local search phase.

5 Computational results

To tune our algorithm, we adapted a set of instances from the literature to test the performance of our algorithm on the PVRP-IF. We also received data from the company containing the additional real world constraints mentioned above. Moreover, we will present the results on the standard benchmark instances for the MDVRPI. All results shown are averaged over 10 runs. Experiments were run on a computer with 2.4 GHz with 4GB RAM running under Linux.

5.1 Parameter settings

Only a few parameters are needed in our VNS implementation. The number of iterations used is 10^7. For penalizing tour length and capacity restriction violations a constant value of 1000 is used. The temperature for the SA acceptance criterion is set to 15 in the beginning for the MDVRPI and the real world instances and to 7 for the PVRP-IF instances and is decreased every 1000 iterations, in a way that it becomes 0 in the last 1000 iterations. We chose these parameter settings based on the performance of the VNS on previous work on the PVRP (see Hemmelmayr et al. 2009) and also on new extensive parameter tests.

5.2 Results for the PVRP-IF

To test our algorithm we adapted instances by Crevier et al. (2007) for the VRP-IF by adding visit day combinations of the instances by Cordeau et al. (1998) for the PVRP. This was possible because the set of customers is the same for both data sets. Table 4 shows the data for the instances. The number of customers (NumCust), the number of IF (NumIF), the maximum Duration (maxDur) and the maximum Capacity (maxCap) are taken from the instances by Crevier et al. (2007), while the number of days of the planning period (NumDays) and the service frequencies are taken from the instances

Table 4 Instance description for the PVRP-IF

Instance	NumVeh	NumCust	NumIF	NumDays	maxDur	maxCap	Service frequencies				
							f1	f2	f3	f4	f6
pr01	2	48	4	4	600	150	24	12		12	
pr02	2	96	4	4	1150	200	48	24		24	
pr03	2	144	4	4	1700	250	72	36		36	
pr04	2	192	4	4	2250	300	96	48		48	
pr05	2	240	4	4	2800	350	120	60		60	
pr06	2	288	4	4	3350	400	144	72		72	
pr07	2	72	6	6	950	175	18	18	18		18
pr08	2	144	6	6	1800	250	36	36	36		36
pr09	2	216	6	6	2650	325	54	54	54		54
pr10	2	288	6	6	3500	400	72	72	72		72

Table 5 Comparison of the VNS with the former neighborhoods used for the standard PVRP to the VNS that additionally uses neighborhoods designed for the PVRP-IF (i.e. intratrip move and cross). Runtimes are given in minutes

Instances	avg cost			CPU avg (minutes)	
	Old nbhs	New nbhs	% gap	Old nbhs	New nbhs
pr01	2273.45	2272.76	−0.03	1.74	1.37
pr02	3551.99	3556.70	0.13	4.80	4.33
pr03	4323.40	4327.40	0.09	11.17	9.55
pr04	4831.02	4802.40	−0.59	19.17	18.68
pr05	4923.22	4916.64	−0.13	35.42	37.21
pr06	5824.40	5786.38	−0.65	51.69	52.38
pr07	4817.19	4810.10	−0.15	4.10	3.33
pr08	5661.06	5661.95	0.02	16.17	15.37
pr09	7685.68	7638.62	−0.61	42.76	42.20
pr10	8722.86	8671.37	−0.59	83.39	80.05
avg.	5261.42	5244.43	−0.25	27.04	26.45

by Cordeau et al. (1998). Since it turned out that the number of vehicles (NumVeh) was too large, it was set to 2.

We have compared and analyzed different design decisions. Table 5 shows an analysis of the neighborhoods used in shaking of the VNS. The first approach only uses the neighborhoods that have been applied for the PVRP (Hemmelmayr et al. 2009). The second approach additionally uses two neighborhood structures designed for the problem at hand. More precisely, these are intertrip move and cross that move and swap customers segments between trips. The two approaches only differ in the set of neighborhood structures used, while the local search is the same for both approaches. In both cases, the local search is the new procedure that first removes the IFs, then reinserts them with *DP* and then performs *2-opt-of*.

Table 6 Computational Results on the Instances of Crevier et al. (2007) (CCL). Runtimes are reported in minutes

Instances	avg cost			avg CPU		Best cost		
	CCL	VNS	% gap	CCL	VNS	CCL	VNS	% gap
a2	1005.16	**997.94**	−0.72	6.39	1.23	**997.94**	**997.94**	0.00
b2	1333.20	**1291.19**	−3.15	14.72	6.41	1307.28	**1291.19**	−1.23
c2	1792.46	**1715.84**	−4.27	61.68	15.01	1747.61	**1715.60**	−1.83
d2	1898.21	**1860.92**	−1.96	40.54	30.14	1871.42	**1856.84**	−0.78
e2	1995.75	**1922.81**	−3.65	73.78	49.31	1942.85	**1919.38**	−1.21
f2	2312.15	**2233.43**	−3.40	162.22	71.24	2284.35	**2230.32**	−2.37
g2	1185.93	**1153.17**	−2.76	29.51	3.71	1162.58	**1152.92**	−0.83
h2	1611.75	**1575.28**	−2.26	160.79	15.66	1587.37	**1575.28**	−0.76
i2	1998.20	**1922.24**	−3.80	322.41	41.92	1972.00	**1919.74**	−2.65
j2	2325.18	**2250.21**	−3.22	256.85	73.38	2294.06	**2247.70**	−2.02
average	1745.80	**1692.30**	−2.92	112.89	30.80	1716.75	**1690.69**	−1.37

5.3 Results on the MDVRPI instances

We also compared the results of our algorithms to the results of Crevier et al. (2007) (CCL) and Tarantilis et al. (2008) (TZK) for the MDVRPI. Note that the best solutions of CCL were identified during sensitivity analysis and the best solutions of TZK were identified with the standard parameter settings that they have used. The best solutions of our algorithms indicated in the tables are the best solutions obtained in the 10 runs. A list of new best known solutions found throughout the sensitivity analysis can be found in the Appendix.

Concerning runtimes, CCL used a Prosys, 2 GHz computer and TZK performed their algorithm on a PentiumIV-2.4 GHz with 512MB of RAM under Windows XP. Our runs were executed on a computer with 2.4 GHz with 4GB RAM running under Linux.

There are three data sets. The first data set was proposed by Crevier et al. (2007). A comparison of our algorithm (VNS) with theirs (CCL) can be found in Table 6. We show the average solution quality over 10 runs, the CPU time in minutes and the best solution obtained during the 10 runs. We are able to improve their results by -2.92% on average. Also the runtime is faster on comparable machines.

Table 7 shows the results for the second data set. It was provided by Crevier et al. (2007) and solved by them (CCL) and by Tarantilis et al. (2008) (TZK). It also shows the average solution quality, the average runtime in minutes and the best solution found during the 10 runs. On average, we are able to improve on the best of the two benchmark algorithms by -0.99%. The best of our 10 runs improves on the best found solution of the benchmark algorithms by -0.22% in comparable runtimes. Moreover, for each instance we are either improving or there is a tie.

Finally, results for the third data set are shown in Table 8. Unfortunately Tarantilis et al. (2008) have only indicated the best solution found and the time when the best solution was found. We present the average percentage deviation to the best solutions

Table 7 Computational Results on the Instances of Crevier et al. (2007) with the algorithm by Crevier et al. (2007) (CCL) and Tarantilis et al. (2008) (TZK). Runtimes are reported in minutes

Instances	avg cost				avg CPU			Best cost			
	CCL	TZK	VNS	%gap to min	CCL	TZK	VNS	CCL	TZK	VNS	%gap to min
a1	1211.28	1189.70	**1180.57**	−0.77	4.58	3.38	1.42	**1179.79**	**1179.79**	**1179.79**	0.00
b1	1232.67	1225.08	**1217.07**	−0.65	9.17	7.80	6.39	**1217.07**	**1217.07**	**1217.07**	0.00
c1	1893.01	1898.92	**1867.96**	−1.32	36.22	34.21	20.40	1886.15	1883.05	**1866.76**	−0.87
d1	1076.31	1064.29	**1059.43**	−0.46	8.55	5.87	1.57	**1059.43**	**1059.43**	**1059.43**	0.00
e1	1311.6	**1309.12**	**1309.12**	0.00	13.52	8.62	6.22	**1309.12**	**1309.12**	**1309.12**	0.00
f1	1601.54	1585.83	**1573.05**	−0.81	41.41	38.81	25.60	1576.33	1572.17	**1570.41**	−0.11
g1	1202	1190.21	**1183.32**	−0.58	55.22	5.79	3.38	**1181.13**	**1181.13**	**1181.13**	0.00
h1	1598.51	1577.54	**1548.61**	−1.83	32.07	11.06	14.61	1547.25	1547.25	**1545.50**	−0.11
i1	1976.11	1956.17	**1923.52**	−1.67	51.01	42.50	33.58	1927.99	1925.99	**1922.18**	−0.20
j1	1161.77	1128.86	**1115.78**	−1.16	58.90	5.52	2.78	1120.65	1117.20	**1115.78**	−0.13
k1	1618.45	1591.74	**1577.96**	−0.87	64.61	12.07	14.56	1586.92	1580.39	**1576.36**	−0.26
l1	1917.08	1904.39	**1869.70**	−1.82	104.27	51.39	35.48	1884.92	1880.60	**1863.28**	−0.92
average	1483.36	1468.49	1452.17	−0.99	39.96	18.92	13.83	1456.40	1454.43	1450.57	−0.22

Table 8 Computational Results on the Instances of Tarantilis et al. (2008). Indicated is the percentage deviation of the best found solution during 10 runs of the VNS to the best solution obtained by TZK as well as the percentage deviation of the time when the best solution was found

Instance	% gap best	% gap runtime	Instance	% gap best	% gap runtime	Instance	% gap best	% gap runtime
50c3d2v	**0.00**	−99.88	100c3d3v	**0.00**	−93.72	150c4d3v	**0.00**	111.06
50c3d4v	**0.00**	−98.66	100c3d5v	**−0.11**	−68.30	150c4d5v	**−0.44**	−73.19
50c3d6v	**−0.05**	−98.36	100c3d7v	**−0.11**	−71.21	150c4d7v	**−1.12**	−38.15
50c5d2v	**0.00**	−99.89	100c5d3v	**0.00**	−99.39	150c6d3v	**0.00**	37.88
50c5d4v	**0.00**	−99.92	100c5d5v	**0.00**	−99.58	150c6d5v	**−0.34**	67.96
50c5d6v	**−0.09**	−95.18	100c5d7v	**−0.13**	−78.89	150c6d7v	**−1.27**	−47.03
50c7d2v	**0.00**	−94.83	100c7d3v	**−0.21**	137.98	150c8d3v	**−0.27**	−86.31
50c7d4v	**−0.04**	−97.76	100c7d5v	**−1.07**	−87.53	150c8d5v	**−0.95**	−71.14
50c7d6v	**−0.57**	−98.54	100c7d7v	**−1.01**	−79.56	150c8d7v	**−1.05**	−76.05
75c3d2v	**0.00**	−96.67	125c4d3v	**−0.10**	−88.40	175c4d4v	**−0.07**	21.59
75c3d4v	**0.00**	−98.91	125c4d5v	**−0.17**	−98.17	175c4d6v	**−0.39**	−73.64
75c3d6v	**−1.76**	−78.54	125c4d7v	0.13	−70.42	175c4d8v	**−1.90**	−55.28
75c5d2v	**0.00**	−76.38	125c6d3v	**−0.02**	−92.53	175c6d4v	**−0.52**	5.70
75c5d4v	**−0.42**	−87.84	125c6d5v	**−1.37**	−77.07	175c6d6v	**−2.55**	−68.40
75c5d6v	**−0.33**	−76.99	125c6d7v	**−3.03**	−79.81	175c6d8v	**−0.87**	−80.09
75c7d2v	**0.00**	−90.17	125c8d3v	**−0.42**	49.74	175c8d4v	**−2.23**	−35.27
75c7d4v	**−0.22**	−97.94	125c8d5v	**−1.96**	−52.59	175c8d6v	**−1.07**	−61.48
75c7d6v	0.55	−98.83	125c8d7v	**−1.50**	−62.45	175c8d8v	**−2.45**	−86.43

Table 9 Results for the not-unloading-at-the-end-of-the-day case, runtime in minutes and the gap to the case where unloading at the end of the day is obligatory

Instances	Do not unload	% gap to unload	Time
pr01	2233.56	−1.72	4
pr02	3490.53	−1.86	9
pr03	4307.45	−0.46	17
pr04	4813.1	0.22	28
pr05	4929.03	0.25	40
pr06	5788.94	0.04	59
pr07	4748.54	−1.28	9
pr08	5703.53	0.73	27
pr09	7582.26	−0.74	58
pr10	8575.17	−1.11	97
avg.	5217.21	−0.59	35

obtained by the VNS in 10 runs to the algorithm of Tarantilis et al. (2008). Also the deviation to the time when the best solution was found is indicated. Table 11 in the Appendix lists the results in detail.

A list of the best known solutions found with different parameter settings can be found in the appendix. Note that these best known solutions are even better than the solutions provided in Tables 6, 7 and 8.

5.4 Real world constraints

We now explain the findings from the experiments with the real world constraints. In order to identify the effects of these modifications more clearly, we first introduce these constraints in the standard benchmark instances and analyze the results. Then we briefly discuss the real world instances.

5.4.1 Return empty

First, we evaluated the impact of the special real-world constraint that the vehicles do not have to unload at the end of the day. When we allow that the vehicles return to the depot partly filled then we can further improve the solution quality by 0.59% as shown in Table 9. When the legal regulations allow to park the waste collection vehicle partly filled at the depot the solid waste collection company can further reduce the logistics costs. Although the trip scheduling problem, which is the problem of planning the allocation of the fleet to the different types of waste, can become more difficult. When the vehicles are partly filled in the morning it is only possible to collect the same type of waste with the vehicle on the current day as the day before.

5.4.2 Distribution of waste

In Table 10 we performed a sensitivity analysis of adding different capacity limits to the different intermediate facilities. The capacity limits are expressed in % of total

Table 10 Results for the capacity limit at IFs. In the equal distribution case all IFs have the same limit, in the biased distribution case one IF has a high limit. The gap is computed to the values reported in Table 5

Instance	Equal distribution		Biased distribution		CPU avg	
	Results	% gap to basic case	Results	% gap to basic case	Case 1	Case 2
pr01	2310.01	1.64	2323.66	2.24	5	4
pr02	3676.25	3.36	3833.62	7.79	14	14
pr03	4483.84	3.62	4888.69	12.97	29	30
pr04	4989.47	3.90	5303.45	10.43	53	55
pr05	5196.10	5.68	5406.20	9.96	73	78
pr06	6175.05	6.72	6334.07	9.47	143	148
pr07	4974.95	3.43	5539.50	15.16	13	14
pr08	6139.52	8.43	7185.77	26.91	41	44
pr09	8472.35	10.91	8499.20	11.27	106	107
pr10	9502.79	9.59	10351.32	19.37	183	199
average	5592.03	5.73	5966.55	12.56	66	69

demand. We have studied two cases and compared them to the basic case, which means that no capacity limit applies (reported in Table 5). The first case corresponds to an equal capacity limit at each IF. More precisely, we have divided the total demand by the number of IFs. For the second case we have given a high capacity limit to the first IF and a low limit to the remaining ones. More precisely, for the case with 4 IFs we allowed a limit of 70% for the first IF and 10% for the remaining and for the case with 6 IF we allowed a limit of 75% and 5% respectively. Then we have added 10% to each limit, because such a strict capacity limit does not make sense in the real life, it will only lead to bad or infeasible solutions.

The results indicate that capacity limits at the IFs (even if a 10% tolerance is permitted) increase the routing cost by about 6% on average. In case of a very biased distribution of the capacities the additional cost is even doubled.

5.4.3 Results on real-world instances

We were also provided with real world data by the company that we collaborated with. We received data from three different communities. More precisely, the instances range from small to large ones. The biggest one has 387 customers, 2 vehicles, 3 IFs and 5 days. The second instance has 184 customers, 1 vehicle, 1 IF and 5 days, while the last instance has 78 customers, 2 IFs and only one vehicle and one day.

With the algorithm that we developed, we were able to improve their results by 25% according to the data that we received.

For these three instances, some parameters were changed to perform a sensitivity analysis.

During these experiments expected results were obtained:

- Some cost reduction can be gained when more visit day combinations can be used, because the additional degree of freedom is beneficial.

- Moreover, when the depot acts as an IF, there is further improvement, because at least the trip from the last IF to the depot is saved. Of course this depends on the location of the depot and the IFs in the customer setting.
- An increased tour length can provide additional savings.
- Capacity limits at the IFs lead to cost increases

Since these real world experiments are based on very few instances, we refrain from giving any numbers.

6 Conclusion

In this paper an important problem in reverse logistics was considered—the collection of solid waste. More precisely, we analyzed the waste collection from public waste collection points, leading to a node routing problem. We introduced a formal model for the PVRP-IF and developed a new hybrid algorithm for this problem. The hybridization was done by combining a VNS algorithm with an exact procedure for inserting the intermediate facilities. We showed that a sophisticated insertion procedure for intermediate facilities can improve the solution quality if it is combined with a local search algorithm. Different design decisions were presented and evaluated.

Then, we extended the algorithm to consider additional aspects and constraints arising in real world solid waste collection. More precisely, we considered the case where a capacity limit at the IFs has to be observed. In case of an even distribution of capacities an average cost increase of 6% was identified, while heterogeneous capacities are more costly. Furthermore, when vehicles can return to the depot partly loaded at the end of the day minor cost savings can be achieved. When solving some real world instances with our algorithm, an average cost saving of about 25% was obtained compared to the manual planning. The algorithm is currently being integrated in a decision support system of a consulting company in Austria.

Finally, we have also shown that the basic concept of the algorithm is very robust for a larger class of node based waste collection problems. After minor modifications, it was also applied to a related but different problem, namely the MDVRPI, where just one day is considered and the depot can be used as an IF at the end of the day. In three different sets of benchmark instances we were able to improve average and best known results and outperformed two existing algorithms by showing an improvement of -0.84% on average. Concerning the new best known solutions found, our algorithm was able to find new best known solutions for 49 out of the 76 instances while we achieved a tie in the remaining 27 ones.

Future work will focus on the planning decisions that have to be made before the solution of the problems considered here. On the one hand, this concerns the optimal number of bins for each waste type at the collection points. Note that fewer bins require a higher visit frequency, which usually leads to higher routing costs. Due to the limited space available at most collection points also the total number of bins at these sites is limited and this scarce resource has to be distributed optimally over the different types of waste. Furthermore, also the fleet should be used efficiently and the decision, how many vehicles should collect which type of waste on which day, is challenging.

Acknowledgements The authors would like to thank our industrial partners Hans-Georg Frantz and Hans Koschuh from B.I.M. consulting and Gert Schweiger from FAGUS for providing the real-world data. Thanks are also due to Guenter Kiechle for the project management of the real world application. Furthermore Financial support from the Austrian Research Promotion Agency (FFG-project bridge No. 818058) and from the Austrian Science Fund (FWF-project No. P20342-N13) is gratefully acknowledged.

Appendix

Table 11 shows the best solutions obtained by our algorithm and by TZK as well as the average solution quality of the VNS compared to the best solution found by TZK. Note that even the average results of our VNS are able to improve their best solutions by −0.42%. We also show a comparison of the time when the best solution is found in seconds and the total runtime used by the VNS in seconds. It can be seen that the VNS also needs lower runtimes.

Table 11 Computational Results on the Instances of Tarantilis et al. (2008) compared to our algorithm. The average solution quality as well as the best solution found during 10 runs of the VNS are compared to the best solution found by TZK. Runtimes are in seconds

Instance	C best			C avg		Time best solution found		Total time
	TZK bks	VNS bks	% gap best	VNS avg	% gap avg.	TZK	VNS	VNS
50c3d2v	**2209.83**	**2209.83**	0.00	**2209.83**	0.00	170.80	0.20	397.20
50c3d4v	**2368.33**	**2368.33**	0.00	**2368.33**	0.00	133.90	1.80	105.50
50c3d6v	3000.88	**2999.29**	−0.05	**2999.29**	−0.05	164.20	2.70	68.90
50c5d2v	**2608.25**	**2608.25**	0.00	**2608.25**	0.00	92.40	0.10	458.20
50c5d4v	**3086.58**	**3086.58**	0.00	**3086.58**	0.00	124.00	0.10	122.60
50c5d6v	3552.00	**3548.88**	−0.09	3557.08	0.14	182.40	8.80	76.60
50c7d2v	**3353.08**	**3353.08**	0.00	**3353.08**	0.00	189.70	9.80	846.30
50c7d4v	3381.57	**3380.27**	−0.04	**3380.27**	−0.04	201.30	4.50	285.40
50c7d6v	4097.80	**4074.44**	−0.57	**4089.26**	−0.21	205.30	3.00	89.70
75c3d2v	**2678.80**	**2678.80**	0.00	**2678.80**	0.00	270.00	9.00	1427.60
75c3d4v	**2746.74**	**2746.74**	0.00	**2746.74**	0.00	202.70	2.20	428.00
75c3d6v	3454.71	**3393.89**	−1.76	**3393.89**	−1.76	293.60	63.00	198.30
75c5d2v	**3373.69**	**3373.69**	0.00	**3373.69**	0.00	197.30	46.60	1766.80
75c5d4v	3568.35	**3553.46**	−0.42	**3560.91**	−0.21	212.10	25.80	245.30
75c5d6v	4198.61	**4184.65**	−0.33	**4184.80**	−0.33	250.80	57.70	190.30
75c7d2v	**3569.02**	**3569.02**	0.00	**3569.02**	0.00	322.50	31.70	1550.60
75c7d4v	3830.43	**3822.10**	−0.22	3862.82	0.85	330.80	6.80	325.60
75c7d6v	**4239.76**	**4263.24**	0.55	4263.24	0.55	257.40	3.00	152.70
100c3d3v	**3123.51**	**3123.51**	0.00	**3123.51**	0.00	420.60	26.40	879.50
100c3d5v	3552.50	**3548.75**	−0.11	**3551.90**	−0.02	438.50	139.00	274.70
100c3d7v	4239.83	**4235.31**	−0.11	4247.37	0.18	397.00	114.30	241.50
100c5d3v	**4053.95**	**4053.95**	0.00	**4053.95**	0.00	472.50	2.90	1453.50

Table 11 (*Continued*)

Instance	C best			C avg		Time best solution found		Total time
	TZK bks	VNS bks	% gap best	VNS avg	% gap avg.	TZK	VNS	VNS
100c5d5v	**4413.17**	**4413.17**	0.00	**4413.17**	0.00	432.20	1.80	224.10
100c5d7v	5148.98	**5142.52**	−0.13	**5142.66**	−0.12	462.90	97.70	159.90
100c7d3v	4216.47	**4207.79**	−0.21	**4207.79**	−0.21	511.80	1218.00	1816.10
100c7d5v	4462.51	**4414.69**	−1.07	**4440.65**	−0.49	527.50	65.80	276.00
100c7d7v	4897.47	**4847.79**	−1.01	**4868.14**	−0.60	501.10	102.40	199.60
125c4d3v	3920.05	**3916.02**	−0.10	**3916.02**	−0.10	523.50	60.70	1877.50
125c4d5v	4315.68	**4308.44**	−0.17	**4308.44**	−0.17	540.20	9.90	457.10
125c4d7v	**4763.49**	**4769.75**	0.13	4770.99	0.16	503.80	149.00	285.90
125c6d3v	4064.20	**4063.25**	−0.02	**4063.25**	−0.02	551.20	41.20	1684.80
125c6d5v	4826.71	**4760.47**	−1.37	**4764.60**	−1.29	499.80	114.60	527.50
125c6d7v	5325.28	**5164.03**	−3.03	**5200.20**	−2.35	550.70	111.20	227.00
125c8d3v	4553.28	**4534.14**	−0.42	**4541.45**	−0.26	613.80	919.10	2379.00
125c8d5v	5045.65	**4947.00**	−1.96	**5041.75**	−0.08	578.20	274.10	619.60
125c8d7v	5416.96	**5335.79**	−1.50	**5339.66**	−1.43	560.30	210.40	344.60
150c4d3v	**4049.48**	**4049.48**	0.00	4054.57	0.13	582.40	1229.20	2765.70
150c4d5v	4638.72	**4618.23**	−0.44	**4625.54**	−0.28	491.20	131.70	871.10
150c4d7v	5176.50	**5118.41**	−1.12	**5141.36**	−0.68	480.00	296.90	450.30
150c6d3v	**4057.09**	**4057.09**	0.00	4057.70	0.01	597.70	824.10	2901.10
150c6d5v	4872.08	**4855.29**	−0.34	**4859.56**	−0.26	613.70	1030.80	1834.30
150c6d7v	5768.29	**5695.26**	−1.27	**5695.78**	−1.26	643.90	341.10	611.40
150c8d3v	4653.90	**4641.30**	−0.27	**4641.30**	−0.27	610.50	83.60	3108.60
150c8d5v	5113.77	**5065.11**	−0.95	**5065.11**	−0.95	697.20	201.20	688.60
150c8d7v	5665.23	**5605.83**	−1.05	**5614.64**	−0.89	720.30	172.50	303.30
175c4d4v	4706.76	**4703.25**	−0.07	4709.46	0.06	1304.50	1586.10	2524.90
175c4d6v	4835.64	**4816.54**	−0.39	**4816.99**	−0.39	1380.60	363.90	1045.50
175c4d8v	5943.28	**5830.63**	−1.90	**5851.61**	−1.54	1104.00	493.70	839.50
175c6d4v	5025.51	**4999.43**	−0.52	**5010.23**	−0.30	1290.70	1364.30	3349.50
175c6d6v	5431.34	**5292.88**	−2.55	**5298.15**	−2.45	1352.10	427.30	862.80
175c6d8v	6090.01	**6037.20**	−0.87	**6040.99**	−0.80	1548.70	308.40	425.50
175c8d4v	5878.58	**5747.73**	−2.23	**5748.19**	−2.22	1494.20	967.20	2653.60
175c8d6v	5989.63	**5925.40**	−1.07	**5931.18**	−0.98	1512.30	582.60	912.50
175c8d8v	6943.63	**6773.77**	−2.45	**6820.52**	−1.77	1601.80	217.30	406.30
average	4342.55	4311.74	−0.58	4319.71	−0.42	572.46	269.58	911.44

Table 12 shows the best solutions that were identified through sensitivity analysis for the PVRP-IF and Table 13 shows the new best known solutions for the MDVRPI. For the MDVRPI, the average deviation from the best known solutions found by other authors is −1.55% for set 1, −0.28% for set 2 and −0.78% for set 3. Note that for the MDVRPI we are always either improving the best known solution found in the literature or there is a tie.

Table 12 Best known solutions identified through sensitivity analysis for the PVRP-IF

Instance	bks found
pr01	2268.42
pr02	3528.86
pr03	4256.36
pr04	4766.71
pr05	4855.39
pr06	5738.52
pr07	4790.47
pr08	5624.35
pr09	7579.62
pr10	8615.69

Table 13 Best known solutions identified through sensitivity analysis for the MDVRPI

Instance	bks found	Instance	bks found	Instance	bks found	Instance	bks found
a2	997.94	j1	1115.78	75c7d4v	3822.10	125c8d7v	5334.92
b2	1291.19	k1	1573.21	75c7d6v	4239.76	150c4d3v	4049.48
c2	1715.6	l1	1863.28	100c3d3v	3123.51	150c4d5v	4612.05
d2	1854.04	50c3d2v	2209.83	100c3d5v	3548.44	150c4d7v	5118.41
e2	1916.69	50c3d4v	2368.33	100c3d7v	4235.31	150c6d3v	4057.09
f2	2230.32	50c3d6v	2999.29	100c5d3v	4053.95	150c6d5v	4855.29
g2	1152.92	50c5d2v	2608.25	100c5d5v	4413.17	150c6d7v	5695.26
h2	1575.28	50c5d4v	3086.58	100c5d7v	5142.52	150c8d3v	4641.3
i2	1919.74	50c5d6v	3548.88	100c7d3v	4207.79	150c8d5v	5065.11
j2	2247.7	50c7d2v	3353.08	100c7d5v	4412.7	150c8d7v	5605.83
a1	1179.79	50c7d4v	3380.27	100c7d7v	4847.79	175c4d4v	4692.54
b1	1217.07	50c7d6v	4074.44	125c4d3v	3916.02	175c4d6v	4816.54
c1	1866.76	75c3d2v	2678.80	125c4d5v	4308.44	175c4d8v	5830.63
d1	1059.43	75c3d4v	2746.74	125c4d7v	4666.9	175c6d4v	4996.29
e1	1309.12	75c3d6v	3393.89	125c6d3v	4063.25	175c6d6v	5291.63
f1	1570.41	75c5d2v	3373.69	125c6d5v	4760.47	175c6d8v	6024.99
g1	1181.13	75c5d4v	3553.46	125c6d7v	5164.03	175c8d4v	5747.73
h1	1545.50	75c5d6v	4184.65	125c8d3v	4534.14	175c8d6v	5914.01
i1	1922.18	75c7d2v	3569.02	125c8d5v	4947	175c8d8v	6772.69

References

Alegre, J., Laguna, M., Pacheco, J.: Optimizing the periodic pick-up of raw materials for a manufacturer of auto parts. Eur. J. Oper. Res. **179**, 736–746 (2007)

Angelelli, E., Speranza, M.: The application of a vehicle routing model to a waste-collection problem: two case studies. J. Oper. Res. Soc. **53**(9), 944–952 (2002a)

Angelelli, E., Speranza, M.G.: The periodic vehicle routing problem with intermediate facilities. Eur. J. Oper. Res. **137**(2), 233–247 (2002b)

Archetti, C., Speranza, M.: Vehicle routing in the 1-skip collection problem. J. Oper. Res. Soc. **55**(7), 717–727 (2004)
Baptista, S., Oliveira, R., Zuquete, E.: A period vehicle routing case study. Eur. J. Oper. Res. **139**(2), 220–229 (2002)
Beasley, J.: Route-first cluster-second methods for vehicle routing. Omega **11**(4), 403–408 (1983)
Beltrami, E., Bodin, L.: Networks and vehicle routing for municipal waste collection. Networks **4**(1) (1974)
Bodin, L., Mingozzi, A., Baldacci, R., Ball, M.: The rollon-rolloff vehicle routing problem. Transp. Sci. **34**(3), 271 (2000)
Christofides, N., Beasley, J.: The period routing problem. Networks **14**(2), 237–256 (1984)
Clarke, G., Wright, J.W.: Scheduling of vehicles from a central depot to a number of delivery points. Oper. Res. **12**, 568–581 (1964)
Cordeau, J., Gendreau, M., Laporte, G.: A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks **30**(2), 105–119 (1998)
Crevier, B., Cordeau, J., Laporte, G.: The multi-depot vehicle routing problem with inter-depot routes. Eur. J. Oper. Res. **176**(2), 756–773 (2007)
Croes, G.: A method for solving traveling salesman problems. Oper. Res. **6**, 791–812 (1958)
Eisenstein, D., Iyer, A.: Garbage collection in Chicago: a dynamic scheduling model. Manag. Sci. **43**(7), 922–933 (1997)
Gelatt, S., Vecchi, M.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
Golden, B., Assad, A., Wasil, E.: Routing vehicles in the real world: Applications in the solid waste, beverage, food, dairy and newspaper industry. In: Toth, P., Vigo, D. (eds.) The Vehicle Routing Problem, pp. 245–286. SIAM, Philadelphia (2001)
Hansen, P., Mladenovic, N.: Variable neighborhood search: Principles and applications. Eur. J. Oper. Res. **130**, 449–467 (2001)
Hemmelmayr, V., Doerner, K., Hartl, R.: A variable neighborhood search heuristic for periodic routing problems. Eur. J. Oper. Res. **195**(3), 791–802 (2009)
Kindervater, G.A.P., Savelsbergh, M.: Vehicle routing: Handling edges exchanges windows. In: Aarts, E., Lenstra, J. (eds.) Local Search in Combinatorial Optimization. Wiley, Chichester (1997)
Kulcar, T.: Optimizing solid waste collection in Brussels. Eur. J. Oper. Res. **90**(1), 71–77 (1996)
Lacomme, P., Prins, C., Ramdane-Cherif, W.: Evolutionary algorithms for periodic arc routing problems. Eur. J. Oper. Res. **165**(2), 535–553 (2005)
Lin, S.: Computer solutions of the traveling salesman problem. Bell Syst. Tech. J. **44**, 2245–2269 (1965)
Mourão, M., Almeida, M.: Lower-bounding and heuristic methods for a refuse collection vehicle routing problem. European Journal of Operational Research **121**(2), 420–434 (2000)
Mourão, M., Amado, L.: Heuristic method for a mixed capacitated arc routing problem: A refuse collection application. European Journal of Operational Research **160**(1), 139–153 (2005)
Potvin, J., Rousseau, J.: An exchange heuristic for routeing problems with time windows. J. Oper. Res. Soc. 1433–1446 (1995)
Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Comput. Oper. Res. **31**(12), 1985–2002 (2004)
Prins, C., Bouchenoua, S.: A memetic algorithm solving the VRP, the CARP and general routing problems with nodes, edges and arcs. In: Recent Advances in Memetic Algorithms, pp. 65–85 (2005)
Tarantilis, C., Zachariadis, E., Kiranoudis, C.: A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. INFORMS J. Comput. **20**(1), 154 (2008)
Teixeira, J., Antunes, A., de Sousa, J.: Recyclable waste collection planning—a case study. European Journal of Operational Research **158**(3), 543–554 (2004)
Tung, D., Pinnoi, A.: Vehicle routing–scheduling for waste collection in Hanoi. Eur. J. Oper. Res. **125**(3), 449–468 (2000)