OR Spectrum (2015) 37:353–387
DOI 10.1007/s00291-014-0376-5

REGULAR ARTICLE

An adaptive VNS algorithm for vehicle routing problems with intermediate stops

Michael Schneider · Andreas Stenger · Julian Hof

Published online: 2 October 2014

Abstract There are numerous practical vehicle routing applications in which vehicles have to stop at certain facilities along their routes to be able to continue their service. At these stops, the vehicles replenish or unload their cargo or they stop to refuel. In this paper, we study the vehicle routing problem with intermediate stops (VRPIS), which considers stopping requirements at intermediate facilities. Service times occur at these stops and may depend on the load level or fuel level on arrival. This is incorporated into the routing model to respect route duration constraints. We develop an adaptive variable neighborhood search (AVNS) to solve the VRPIS. The adaptive mechanism guides the shaking step of the AVNS by favoring the route and vertex selection methods according to their success within the search. The performance of the AVNS is demonstrated on test instances for VRPIS variants available in the literature. Furthermore, we conduct tests on newly generated instances of the electric vehicle routing problem with recharging facilities, which can also be modeled as VRPIS variant. In this problem, battery electric vehicles need to recharge their battery en route at respective recharging facilities.

Keywords Vehicle routing · Intermediate stops · Refueling · Recharging · Electric vehicles

M. Schneider (✉)
DB Schenker Endowed Assistant Professorship: Logistics Planning and Information Systems,
Department of Law and Economics, TU Darmstadt, Darmstadt, Germany
e-mail: schneider@bwl.tu-darmstadt.de

A. Stenger
Lufthansa Technik, Hamburg, Germany
e-mail: andistenger@gmail.com

J. Hof
Business Information Systems and Operations Research, University of Kaiserslautern,
Kaiserslautern, Germany
e-mail: julian.hof@wiwi.uni-kl.de

1 Introduction

Intermediate stops have to be considered in many practical vehicle routing applications, e.g., for (1) replenishment of the goods to be delivered, (2) refueling (or recharging in case of battery electric vehicles), or (3) unloading of collected goods or disposal of waste. These stops differ from regular customer stops in two aspects: first, they are optional, and second, they depend on the state of the vehicle with respect to load and fuel level (which decides the latest possible moment at which an intermediate stop has to occur). Contrary to optional customer stops, e.g., in vehicle routing problems with profits (Archetti et al. 2013), intermediate stops are not directly related to customer service or profit maximization, but aim at keeping the vehicle operational.

As mentioned above, major applications of intermediate stops are good replenishment, refueling, and waste disposal, which are detailed in the following. Intermediate replenishment stops are used in distribution systems with several facilities storing the products to be delivered (Angelelli and Speranza 2002; Crevier et al. 2007; Tarantilis et al. 2008). The aim is to avoid returning to a central depot to reload the delivery vehicle. Concrete applications can be found in the distribution of heating oil (Prescott-Gagnon et al. 2012), road maintenance (Amaya et al. 2007) or in city logistics, where city freighters may visit satellite facilities to be replenished (Crainic et al. 2009).

Intermediate stops for unloading operations are common in waste collection or snow ploughing. Here, intermediate disposal sites need to be visited, at the latest, when the maximal capacity of the vehicle is reached (see, e.g, Kim et al. 2006; Benjamin and Beasley 2010).

Intermediate refueling stops occur in several practical applications. For example, some companies keep contracts with gas station chains to get special rates at the respective stations, which makes it profitable to consider refueling stops in the route planning. Without such contracts, this is not the case because the network of gas stations is generally quite dense in developed countries. The refueling topic gains further relevance by the strong growth in alternative fuels, namely biodiesel, ethanol, hydrogen, methanol, natural gas or propane, for which only a sparse infrastructure is existent. Finally, battery electric vehicles (BEVs) need to stop to recharge during longer routes due to their limited driving range (Conrad and Figliozzi 2011; Schneider et al. 2014). BEV technologies have recently gained importance due to city logistics concepts, which aim at reducing the negative external effects of urban freight transportation. In this context, BEVs seem to be a very good choice as they have no local emissions, operate very efficiently at the stop-and-go level and have low noise levels (Wang and Lin 2013). Moreover, BEVs are defined to be emission free by EU regulation No 510/2011 and are, therefore, a major means to comply with laws and regulations on emissions.

This paper is the first to introduce the vehicle routing problem with intermediate stops (VRPIS), a routing model that considers visits to intermediate facilities to keep vehicles operational. The necessity to visit an intermediate facility depends on the fuel and/or the load level of a delivery vehicle. The time spent at a facility is defined as a function of the fuel and load level on arrival. By abstracting from the actual purpose of the intermediate stop, be it for replenishment, refueling or disposal, our problem definition comprises several problems with specific applications proposed in the literature. The objective of VRPIS is to minimize the total costs composed of travel costs and fixed costs for the deployment of vehicles.

VRPIS extends the NP-hard capacitated VRP (CVRP) by several combinatorial aspects, which makes exact methods unsuitable for solving large problem instances in fast computation time. We develop a heuristic solution approach, namely an adaptive variable neighborhood search (AVNS), which combines ideas of VNS (Hansen and Mladenović 2001) and adaptive large neighborhood search [ALNS, see (Pisinger and Ropke 2007)]. This method has been successfully applied to single- and multi-depot routing problems [see (Stenger et al. 2013a, b)]. To assess the performance of our AVNS, we perform computational studies on benchmark instances of VRPIS variants previously studied in the literature.

Moreover, we investigate the electric VRP with recharging facilities (EVRPRF) as a special case of VRPIS. EVRPRF models the routing decision of logistics service providers employing BEVs. The driving range of a vehicle is restricted by the maximum battery capacity and a distance-related energy consumption along the route, which determines the battery charge. We simplify several real-world characteristics and do not consider the influence of vehicle load, vehicle speed and grades on energy consumption. The battery can be recharged at any of the available recharging facilities. For this problem, we generate a set of small benchmark instances, which are used to assess the solution quality of our method in comparison to the commercial solver CPLEX. In addition, detailed results for a set of large instances are provided.

This paper is organized as follows. In Sect. 2, we review the related literature. Section 3 presents the problem description and the mathematical models of VRPIS and of the special case EVRPRF. The AVNS solution method is detailed in Sect. 4. Computational tests to assess the performance of the proposed method are described in Sect. 5. The paper is summarized and concluded in Sect. 6.

2 Literature review

This section gives short reviews of the following strands of literature related to VRPIS and EVRPRF: (1) VRP with intermediate replenishment or disposal stops, (2) VRP with refueling or recharging stops, and (3) refueling problems occurring in other application areas.

Crevier et al. (2007) introduce the multi-depot VRP with inter-depot routes (MDVRPI), which considers intermediate depots at which vehicles can be replenished with goods during the course of a route. The authors develop a heuristic procedure that combines ideas from adaptive memory programming (Rochat and Taillard 1995), tabu search (TS) and integer programming. Although the multi-depot case is described, all proposed benchmark instances consider only one depot at which the vehicle fleet is stationed. Therefore, Tarantilis et al. (2008) rename the problem to VRP with intermediate replenishment facilities (VRPIRF), and we adopt this acronym for the remainder of this paper. They propose a hybrid guided local search heuristic that follows a three-step procedure. First, an initial solution is constructed by means of a cost-savings heuristics. Second, a VNS algorithm is applied using a TS in the local search phase. Third, the solution is further improved by means of a guided local search.

Prescott-Gagnon et al. (2012) propose three metaheuristics to solve a VRP arising in heating oil distribution, considering intra-route replenishments, heterogeneous vehicles, optional customer visits and time windows. The authors design a TS, an LNS based on the TS and a column generation heuristic and report computational results obtained on test instances derived from a real-world dataset. Other problems similar to VRPIRF arise in the collection of waste (see, e.g., Angelelli and Speranza 2002; Kim et al. 2006; Coene et al. 2010; Benjamin and Beasley 2010), in snow clearance (Perrier et al. 2007), or in road maintenance and marking (Amaya et al. 2007; Salazar-Aguilar et al. 2013). A recent review of the literature on waste collection can be found in Beliën et al. (2014).

Hemmelmayr et al. (2013) study the periodic vehicle routing problem with intermediate facilities (PVRP-IF) in the context of waste collection. The authors introduce a hybrid solution approach consisting of a VNS using dynamic programming to insert intermediate facilities. The solution procedure is also applied to the VRPIRF problem instances provided by Crevier et al. (2007) and Tarantilis et al. (2008) and is able to outperform both approaches.

The literature on routing problems with refueling stops is still relatively scarce. Conrad and Figliozzi (2011) present the recharging VRP, in which vehicles with limited range have the possibility of recharging en route at certain customer locations. The recharging time is assumed to be fixed. The impact of maximum driving range, recharging time and time window existence is studied using a selection of the VRP with time windows (VRPTW) instances of Solomon (1987). Moreover, bounds are formulated to predict average tour lengths. Erdoğan and Miller-Hooks (2012) propose the green VRP (G-VRP), which considers a limited fuel capacity of the vehicles and the possibility of refueling at facilities along the route with a fixed refueling time. Neither capacity restrictions nor time window constraints are considered. The authors propose two heuristics to solve G-VRP. The first is a modified Clarke and Wright savings algorithm (MCWS) which creates routes by establishing feasibility through the insertion of refueling facilities, merging feasible routes according to savings and removing redundant facilities. The second heuristic is a density-based clustering algorithm (DBCA) designed as cluster-first and route-second approach.

Schneider et al. (2014) develop a hybrid heuristic approach that combines VNS with TS to address the electric vehicle routing problem with time windows and recharging stations (E-VRPTW). Contrary to the EVRPRF studied in this paper, their E-VRPTW includes time windows, but features no maximal route duration constraints. Moreover, their objective is hierarchical and inspired by the objective function used in heuristic methods for the VRPTW: they first minimize the number of employed vehicles and only minimize traveled distance second, whereas we follow the objective of minimizing total costs composed of travel costs and fixed vehicle costs. Their VNS/TS is able to significantly improve on the results of Erdoğan and Miller-Hooks (2012) on the G-VRP instances and achieves convincing results on the VRPIRF instances, although the method is not specifically designed for this type of problem.

Refueling problems are also investigated in other application areas, e.g., the refueling of aircraft or locomotives. In Barnes et al. (2004), tanker aircraft stationed at several bases have to be assigned to receiver aircraft to perform refueling in midair. Raviv and

Kaspi (2012) deal with the optimal refueling schedule of locomotives pulling trains, i.e., the determination of the yards at which the locomotives have to be refueled.

3 Problem definition

This section presents a mixed-integer program of the VRPIS (Sect. 3.1) and derives the formulation of the special case EVRPRF (Sect. 3.2).

3.1 The vehicle routing problem with intermediate stops (VRPIS)

We start with the introduction of some necessary notation. Let $C = \{1, \ldots, n\}$ denote the set of n customers and let F denote the set of facilities. The set F itself comprises the set of refueling facilities F_F, the set of replenishment or unloading/disposal facilities (in the following denoted as loading facilities) F_L, and the set F_{FL} of facilities where both refueling and loading are possible (from now on referred to as combined facilities), i.e., $F = F_F \cup F_L \cup F_{FL}$. We use a set of dummy vertices F' to allow several visits to the facilities in F (see, e.g., Bard et al. 1998; Schneider et al. 2014). Further, let vertices 0 and $n + 1$ denote instances of the same depot representing the start and end of each vehicle route. To indicate which depot instances are included in a fictive set X, the respective depot instances are used as indices, i.e., $X_0 = X \cup \{0\}$, $X_{n+1} = X \cup \{n + 1\}$ and $X_{0,n+1} = X \cup \{0\} \cup \{n + 1\}$. Finally, let $V' = C \cup F'$ denote the set of all customers and visits to facilities.

Then, VRPIS can be defined on the complete directed graph $G = (V'_{0,n+1}, A)$ with the set of arcs $A = \{(i, j) : i, j \in V'_{0,n+1}, i \neq j\}$. Arcs $(i, j) \in A$ are associated with a cost c_{ij}, a distance d_{ij} and a travel time t_{ij}. A homogeneous fleet of m vehicles with load capacity q, fuel capacity p and fixed costs per use c^{fix} is stationed at the depot. Fuel capacity is expressed in distance units and denotes the distance that can be traveled with maximum fuel level.

Each customer $i \in C$ has a positive demand u_i and service time t_i^s. Each facility visit $j \in F'$ is associated with a docking time t_j^d, which marks the time span between the arrival at the facility and the beginning of the actual refueling and/or loading process. The time span for the refueling and loading process is determined by functions $\Phi^f(f_j)$ and $\Phi^l(l_j)$, respectively. It may depend on the fuel level f_j (the cargo level l_j) on arrival at the facility. Visiting a facility $j \in F_F$ completely refills the fuel tank of a vehicle and vehicles are fully replenished or unloaded at facilities in F_L. For combined facilities, we assume that vehicles are fully loaded and are simultaneously refueled during the time span occupied by the loading process. The increase in fuel during that time span is given by the function $\Theta(l_j)$, which depends on the loading time and thus on the load level l_j on arrival at the facility. A refueling process at a combined facility that takes longer than the loading time is modeled by a visit to a refueling facility in F_F with the same location as the combined facility. Each facility is assumed to have an unlimited fuel and cargo capacity, respectively, and can be simultaneously used by any number of vehicles. This assumption seems adequate for many real-world scenarios,

Table 1 Parameters and decision variables of the VRPIS model

$0, n+1$	Instances of the depot
C	Set of customers $= \{1, \ldots, n\}$
F'_F	Set of visits to refueling facilities
F'_L	Set of visits to loading facilities
F'_{FL}	Set of visits to combined facilities
F'	Set of visits to all intermediate facilities, $F' = F'_F \cup F'_L \cup F'_{FL}$
V'	Set of all customers and visits to facilities $C \cup F'$
$V'_{0,n+1}$	Set of all vertices $V'_{0,n+1} = V' \cup \{0\} \cup \{n+1\}$
V'_0	Set of all vertices excluding depot instance $n+1$, $V'_0 = \{0\} \cup V'$
V'_{n+1}	Set of all vertices excluding depot instance 0, $V'_{n+1} = \{n+1\} \cup V'$
c^{fix}	Fixed cost per used vehicle
c_{ij}	Travel costs between vertices i and j
d_{ij}	Distance between vertices i and j
u_i	Demand of customer i ($u_i = 0$ if $i \notin C$)
m	Number of available vehicles
p	Maximal fuel capacity of a vehicle expressed as possible range without refueling
q	Maximal loading capacity of a vehicle
t_{ij}	Travel time between vertices i and j
t^{max}	Maximum route duration
t^d_i	Docking time at intermediate facility i
t^s_i	Service time at customer i ($t^s_i = 0$ if $i \notin C$)
$\Theta(l_i)$	Function returning the amount that is refueled at vertex i during the loading process, given the current load level l_i
$\Phi^f(f_i)$	Function returning the refueling time at a refueling facility depending on fuel level f_i
$\Phi^l(l_i)$	Function returning the loading time at a loading facility depending on the current load level l_i
a_i	Decision variable specifying the arrival time at vertex i
f_i	Decision variable specifying the fuel level at vertex i expressed in distance units
l_i	Decision variable specifying the load level at vertex i
x_{ij}	Binary decision variable indicating if arc $(i, j) \in A$ is traversed

but clearly represents a simplification for scenarios in which capacity and loading possibilities are constrained.

To represent working hour restrictions of real-world applications, we assume that the arrival time of all vehicles at depot instance $n+1$ may not exceed the maximum route duration t^{max}. The following variables are used in the model: a_j specifies the time on arrival at vertex j, f_j the fuel level, and l_j the load level. Binary decision variables x_{ij} take value 1 if vertex j is visited after vertex i and 0 otherwise. Thus, the mixed-integer program of VRPIS is as follows (Table 1 summarizes the notation):

$$\min \sum_{i \in V'_0} \sum_{j \in V'_{n+1}, i \neq j} c_{ij} x_{ij} + \sum_{j \in V'} c^{fix} x_{0j} \tag{1}$$

$$\sum_{i \in V'_0, i \neq j} x_{ij} = 1 \qquad \forall j \in C \tag{2}$$

$$\sum_{i \in V'_0, i \neq j} x_{ij} \leq 1 \qquad \forall j \in F' \tag{3}$$

$$\sum_{j \in V'} x_{0j} \leq m \tag{4}$$

$$\sum_{i \in V'_0, i \neq j} x_{ij} - \sum_{i \in V'_{n+1}, i \neq j} x_{ji} = 0 \qquad \forall j \in V' \tag{5}$$

$$0 \leq a_i \leq t^{max} \qquad \forall i \in V'_{0,n+1} \tag{6}$$

$$a_i + (t_{ij} + t_i^s)x_{ij} - t^{max}(1 - x_{ij}) \leq a_j \qquad \forall i \in C \cup \{0\}, j \in V'_{n+1}, i \neq j \tag{7}$$

$$a_i + (t_{ij} + t_i^d)x_{ij} + \Phi^f(f_i) - t^{max}(1 - x_{ij}) \leq a_j \qquad \forall i \in F'_F, j \in V'_{n+1}, i \neq j \tag{8}$$

$$a_i + (t_{ij} + t_i^d)x_{ij} + \Phi^l(l_i) - t^{max}(1 - x_{ij}) \leq a_j \qquad \forall i \in F'_L \cup F'_{FL}, j \in V'_{n+1}, i \neq j \tag{9}$$

$$0 \leq f_j \leq f_i - d_{ij}x_{ij} + p(1 - x_{ij}) \qquad \forall i \in C \cup F'_L, j \in V'_{n+1}, i \neq j \tag{10}$$

$$0 \leq f_j \leq p - d_{ij}x_{ij} \qquad \forall i \in F'_F \cup \{0\}, j \in V'_{n+1}, i \neq j \tag{11}$$

$$0 \leq f_j \leq f_i + \Theta(l_i) - d_{ij}x_{ij} + p(1 - x_{ij}) \qquad \forall i \in F'_{FL}, j \in V'_{n+1}, i \neq j \tag{12}$$

$$f_i + \Theta(l_i) \leq p \qquad \forall i \in F'_{FL} \tag{13}$$

$$0 \leq l_j \leq l_i - u_i x_{ij} + q(1 - x_{ij}) \qquad \forall i \in C \cup F'_F, j \in V'_{n+1}, i \neq j \tag{14}$$

$$0 \leq l_j \leq q - u_i x_{ij} \qquad \forall i \in \{0\} \cup F'_L \cup F'_{FL}, j \in V'_{n+1}, i \neq j \tag{15}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in V'_0, j \in V'_{n+1}, i \neq j \tag{16}$$

The goal of the VRPIS is to minimize the sum of the total travel cost and the fixed vehicle cost, expressed by the objective function (1). Constraints (2) ensure that every customer must be visited, while optional intermediate stops are ensured by Constraints (3). Constraints (4) guarantee that the number of routes does not exceed the number of available vehicles. Flow conservation is given by Constraints (5). Constraints (6) limit the arrival time at each vertex to the maximum route duration. Time feasibility for arcs leaving customers or the depot is defined by Constraints (7). The same is ensured for arcs leaving refueling facilities and loading facilities in Constraints (8) and (9). Constraints (10) control the fuel feasibility for arcs leaving customers or loading facilities and Constraints (11) guarantee that a refueling facility is left in a completely refueled state. The fuel increase during loading at combined facilities is defined in Constraints (12). Constraints (13) guarantee that no refueling beyond the maximal fuel capacity is possible at combined facilities. Constraints (14) control the load feasibility for arcs leaving customers or refueling facilities. Constraints (15) ensure that vehicles leave loading facilities and the depot in a fully loaded state. Binary decision variables are defined in Constraints (16).

3.2 New special VRPIS case: the electric vehicle routing problem with recharging facilities (EVRPRF)

As described above, we investigate the EVRPRF as a special case of the VRPIS. The VRPIS model is transformed into a formulation of the EVRPRF as follows:

1. No intermediate cargo loading takes place. Therefore, no loading facilities are present in the EVRPRF and sets F_L and F_{FL} are empty and Constraints (9), (12) and (13) can be neglected. To allow for recharging at the depot, dummy instances of 0 are now contained in the set F'_F.
2. We assume a linear recharging process of the vehicle battery, depending on a given average recharging speed g. Since the fuel capacity is expressed as the maximum travel range without refueling, g describes the increase of range per time unit. Thus, the function for the refueling time is defined as: $\Phi^f(f_i) = \frac{p-f_i}{g}$.

3.3 Special cases from the literature: G-VRP and VRPIRF

To assess the performance of our AVNS, we conduct tests on instances of the special VRPIS cases G-VRP and VRPIRF and compare our results to those presented in the literature (see Sect. 5).

G-VRP can be addressed as special case of VRPIS as follows:

- As no loading at intermediate facilities is considered in G-VRP, all related constraints of VRPIS are omitted.
- c^{fix} is set to zero because no vehicle cost is considered in G-VRP.
- The refueling time $\Phi^f(f_i)$ is set to zero and docking time t_i^d is set to the fixed service time of the G-VRP instances.

VRPIRF can be addressed as special case of VRPIS as follows:

- Since the VRPIRF instances do not consider refueling possibilities, all refueling-related constraints of VRPIS are omitted.
- Loading time $\Phi^l(l_i)$ is set to zero because only a fixed docking time t_i^d occurs when visiting a loading facility.
- The maximum route duration $t^{\max}$ is reduced by t_i^d to account for a docking operation at the depot at the end of a route, which is considered in the VRPIRF.
- No vehicle deployment costs are considered in the VRPIRF, so c^{fix} is set to zero.

Table 2 clarifies the relation between VRPIS and the special cases considered in this paper by comparing the properties of each problem.

Table 2 Relation between VRPIS, the introduced special case EVRPRF as well as the special cases from the literature VRPIRF and G-VRP

	EVRPRF	VRPIRF	G-VRP	VRPIS
Fixed vehicle cost	✓			✓
Refueling possible	✓		✓	✓
Loading possible		✓		✓
Fuel-/load-dependent service times	✓			✓
Fixed docking time	✓	✓	✓	✓

4 An adaptive variable neighborhood search algorithm for the VRPIS

In this section, we describe in detail our AVNS algorithm for solving VRPIS. AVNS follows the VNS diversification paradigm of searching in increasingly large neighborhoods [for a detailed introduction to standard VNS, see (Hansen and Mladenović 2001)]. However, routes and vertices involved in the shaking step of AVNS are not selected entirely at random, but are determined by problem-specific rules and the past search performance of these rules. AVNS has previously provided promising results for MDVRP, VRP with private fleet and common carriers (VRPPC), multi-depot VRPPC (Stenger et al. 2013a) and the prize collecting VRP with non-linear cost (Stenger et al. 2013b).

The choice of AVNS is motivated by two factors. First, the high complexity of VRPIS makes it necessary to use an algorithm with strong diversification possibilities. Pretests have shown that classical local-search-based algorithms, like TS, often get stuck in local traps from which they are not able to escape. By contrast, the shaking step of our AVNS modifies up to four routes and moves sequences of up to six nodes in one iteration, which proved to be of vital importance to find promising solutions. Moreover, VNS algorithms presented in the literature have previously shown convincing performance on VRPs with intermediate stops [see, e.g., (Tarantilis et al. 2008)]. Second, to ensure acceptable runtimes on large problem instances, a high efficiency of the search is required. The adaptive mechanism of AVNS takes into account the problem-specific characteristics of VRPIS and adapts based on the recent search performance. Thus, it efficiently guides the search to improving solutions. To sum up, combining the strong diversification of VNS with an adaptive mechanism results in a highly efficient heuristic, characterized by short computing times and high-quality results.

A pseudocode overview of the AVNS is given in Fig. 1. First, the set of neighborhood structures $\{N_\kappa \mid \kappa = 1, \ldots, \kappa^{\max}\}$ is defined. Next, an initial solution S is constructed by means of a modified version of the savings algorithm by Clarke and Wright (1964), which considers the insertion of intermediate stops (Sect. 4.1), and the solution is subsequently improved by a local search (see Sect. 4.3).

In the AVNS component, a guided shaking step is used to diversify the search, producing a random solution S' within the κ-th neighborhood of S (Sect. 4.2.1). The adaptive mechanism is characterized by problem-specific selection methods for the routes and vertices to be shaken instead of an entirely random selection. Besides methods which have proven their effectiveness in previous works, we design specific methods which take the characteristics of intermediate stops into account (Sect. 4.2.2). Each of the selection methods is chosen according to a probability, which is dynamically updated during the search depending on the performance of the method (Sect. 4.2.3).

Subsequently, a greedy local search procedure is applied to obtain the local optimum S'' (Sect. 4.3). In this step, classical operators as well as operators which are able to rearrange intermediate stops are used. If S'' is accepted, it replaces S and κ is reset to one. Otherwise, S'' is discarded and κ is increased by one, i.e, the next neighborhood is selected. We reset to the overall best solution after a certain number of iterations without improvement. The search is stopped after a given number of iterations without improvement of the best solution.

```
Define the neighborhood structures $\mathcal{N}_\kappa$ with $\kappa = 1, .., \kappa^{max}$
Generate initial solution $S$
Improve initial solution by local search
$\kappa \leftarrow 1$
repeat
   {Adaptive Shaking}
   Select route and vertex selection method and generate $S' \in \mathcal{N}_\kappa(S)$
   {Local Search}
   $S'' \leftarrow$ localSearch($S'$)
   {Acceptance Decision}
   if accept($S''$) then
      $S \leftarrow S''$
      $\kappa \leftarrow 1$
   else
      $\kappa \leftarrow \kappa \mod \kappa^{max} + 1$
   end if
   Update weights of route and customer selection methods
until given number of iterations without improvement reached
```

Fig. 1 Pseudocode of the AVNS heuristic for solving VRPIS

As described above, we adapt the algorithmic framework of AVNS to the specifics of VRPIS by incorporating problem-specific knowledge into the selection methods of our adaptive component and into the local search component. Numerical tests have proven the positive effects of these novel methods on the solution quality and runtime of our algorithm (see Appendix for details).

4.1 Initialization with modified savings algorithm

A modification of the savings algorithm, introduced by Clarke and Wright (1964), is used to quickly generate initial vehicle routes that include intermediate stops. We allow the initial solution to be infeasible with respect to fuel, load or duration constraints. The steps of our modified Savings Algorithm are the following:

1. Generate back-and-forth tours for all customers. If such a tour is already infeasible concerning fuel, perform the cost-optimal insertion of a refueling facility into the respective route (see Sect. 4.3 for details).
2. Evaluate potential cost savings for merging each pair of routes and sort the merge moves in decreasing order.
3. Out of the remaining merge moves, select the two routes with highest cost savings and merge them if the maximum route duration is not exceeded. If no merge with positive cost savings exists, stop.
4. Evaluate the resulting route:
   (a) If fuel or load violations emerge in the resulting route, try to resolve them by adding intermediate facilities at the optimal position.
   (b) If the facility insertion leads to a duration violation, cancel the previous merging and continue with Step 3.
   (c) If the resulting route starts or ends with an intermediate facility, i.e., no merging according to customer-related cost savings can be performed at this position,

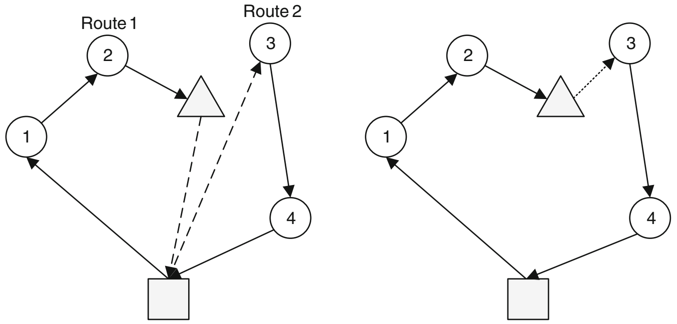

Fig. 2 Merging of routes that start or end with an intermediate facility. Removed arcs are shown with *dashed lines*, inserted arcs with *dotted lines*

try to connect the facility with one of the remaining routes such that the cost increase is minimized and all constraints are still met (see Fig. 2).

5. Continue with Step 3.

The resulting number of routes may exceed the number of available vehicles. In this case, the route with the smallest cumulated customer demand is dissolved and its customers are inserted into the remaining routes at the cost-optimal position. Load capacity, fuel capacity and duration violations are handled by means of a penalizing cost function, see Sect. 4.4. The process of dissolving routes is repeated until the required number of vehicles is reached. Subsequently, the solution is improved by a local search step (see Sect. 4.3).

4.2 The adaptive shaking

In the shaking step of our AVNS, new solutions are generated according to predefined neighborhood structures (Sect. 4.2.1). Problem-specific methods are used for the selection of the routes and vertices to be involved in the shaking (Sect. 4.2.2). The algorithm guides the shaking step by adapting the selection probabilities of these methods according to their previous performance during the search (Sect. 4.2.3).

4.2.1 Shaking neighborhoods

Similar to Stenger et al. (2013a), two operators are employed to generate neighboring solutions: a sequence relocation and a cyclic exchange operator, originally introduced by Thompson and Orlin (1989). The cyclic exchange moves vertices between routes in a cyclic fashion. It is characterized by two parameters: the number of routes involved Ω and the maximum number of vertices to be exchanged $\Gamma^{\max}$.

For each route k the vertex sequence $\Psi^{k}_{j_k,\Gamma_k}$ with start vertex j_k and length Γ_k is transferred to route $k+1$ at the former position of sequence $\Psi^{k+1}_{j_{k+1},\Gamma_{k+1}}$. In Figure 3, the cyclic exchange operator is depicted with $\Omega = 3$ routes, exchanging $\Gamma_1 = 1$, $\Gamma_2 = 2$ and $\Gamma_3 = 2$ vertices. Note that, if the total number of existing routes gets below the

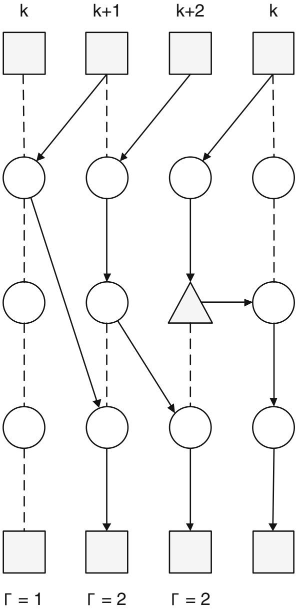

Fig. 3 Example of a cyclic exchange with three routes

number of routes to cycle, Ω is reduced accordingly. Similarly, Γ_k has to be adjusted if it exceeds the number of vertices of route k, denoted with $|V_k|$.

Sequence relocation represents a restriction of the cyclic exchange operator. A vertex sequence is relocated from one route to another, and the latter keeps all of its former vertices. Thus, $\Gamma^{\max} = 0$ applies for the second route.

Table 3 shows the neighborhood structures employed within the search. After six sequence relocation neighborhoods the search continues with 18 cyclic exchange neighborhoods, considering $\Omega = 2$ to $\Omega = 4$ routes between which up to $\Gamma^{\max} = 6$ vertices can be transferred. Sequence lengths with up to $\Gamma^{\max} = 4$ vertices are randomly chosen within the interval $[0, \min(\Gamma^{\max}, |V_k|)]$. Sequence lengths with more than four customers are defined to be fixed.

4.2.2 Selection methods

Instead of determining the routes and vertices to be involved in the shaking entirely at random, the AVNS algorithm guides the shaking step to a certain extent. For this purpose, several methods are implemented which bias the route and vertex sequence selection. On the one hand, we use methods which have proven their effectiveness in previous works on routing problems. On the other hand, we design problem-specific methods which take into account the new characteristics of intermediate stops, e.g., the associated detours required. Each of the methods is chosen with a certain probability, which is dynamically updated during the search depending on the success of the method in former iterations. The selection methods and the adaptive mechanism are detailed in the following.

Table 3 Neighborhood structures examined within the shaking step of the AVNS

κ	Type	$\boldsymbol{\Omega}$	$\boldsymbol{\Gamma}^{\mathbf{max}}$	κ	Type	$\boldsymbol{\Omega}$	$\boldsymbol{\Gamma}^{\mathbf{max}}$
1	Sequence relocation	2	1	13	Cyclic exchange	3	1
2	Sequence relocation	2	2	14	Cyclic exchange	3	2
3	Sequence relocation	2	3	15	Cyclic exchange	3	3
4	Sequence relocation	2	4	16	Cyclic exchange	3	4
5	Sequence relocation	2	5	17	Cyclic exchange	3	5
6	Sequence relocation	2	6	18	Cyclic exchange	3	6
7	Cyclic exchange	2	1	19	Cyclic exchange	4	1
8	Cyclic exchange	2	2	20	Cyclic exchange	4	2
9	Cyclic exchange	2	3	21	Cyclic exchange	4	3
10	Cyclic exchange	2	4	22	Cyclic exchange	4	4
11	Cyclic exchange	2	5	23	Cyclic exchange	4	5
12	Cyclic exchange	2	6	24	Cyclic exchange	4	6

Route selection The first of the Ω_κ routes of the current neighborhood N_κ is chosen according to one of the following five route selection methods:

1. *Random* The probability of being selected is equal for every route.
2. *Route length* The probability of a route for being selected is proportional to the associated travel distance. The intention is to remove vertices from long routes and reinsert them into shorter routes to reduce the total costs.
3. *Route length per demand unit* The selection probability of a route is proportional to the relation of the total distance and the cumulated demand of a route. This criterion shall lead to an improvement of inefficient routes.
4. *Facility density* The probability is proportional to the ratio of the number of intermediate stops to the number of customers within a route. The goal is to favor routes that possibly contain redundant facility visits.
5. *Facility detour* The probability is proportional to the total detour resulting from intermediate stops. This is intended to reduce the associated detours and thus the overall costs.

After choosing the first route by means of one of the procedures above, the other routes to be involved in the shaking are iteratively determined as follows: the next route is randomly chosen from the set of all remaining routes that are spatially closer than a predefined threshold $d^{\max}$ to the previously selected route [cp. (Stenger et al. 2013a)].

Vertex sequence selection Once the routes to be involved are determined, the vertex sequences to be removed from each route must be identified. The following three methods are used for this selection decision:

1. *Random* Each vertex sequence is chosen with the same probability.
2. *Distance to next route* The probability of selecting a vertex sequence is inversely proportional to the distance of the sequence to the route into which it will be

inserted. This is measured by the sum of the vertex distances to the center of gravity of the target route.

3. *Distance to neighboring vertices* The probability of selecting a sequence is proportional to the distance of the sequence to the surrounding vertices. It is given by the sum of the distance between the first vertex and its predecessor and the distance between the last vertex and its successor. Removing a sequence which is far apart from the other vertices of the route can reduce the total costs.

4.2.3 Adaptive mechanism

At each shaking step, the choice of the route and vertex selection methods is based on probabilities. Each method is assigned the same probability at the beginning of the search. The probability of each method is then dynamically updated in the course of the search depending on its success in improving the current solution. To select the methods, we use the roulette wheel selection procedure as proposed by Pisinger and Ropke (2007) for ALNS. Given h selection methods, each method s is assigned a weight w_s. The probability of selecting method s is then defined by $w_s / \sum_{i=1}^{h} w_i$. After γ AVNS iterations, the weight of each method is updated based on its success during these iterations. The performance of a method is measured by a scoring system. A score of nine is added to the total score of a method whenever it achieved a new overall best solution, a score of three if the current solution was improved and a score of one if the solution is worse than the current one, but accepted according to the acceptance criterion. If ϕ_s denotes the current score of method s and χ_s the number of applications of the method since the last weight update, then the new weight is calculated as $w_s = w_s(1 - \rho) + \rho \frac{\phi_s}{\chi_s}$. The system parameter $\rho \in [0, 1]$ allows to control to what extent the past value of the weight influences the new one. The values ϕ_s and χ_s are reset to zero after each update.

4.3 Local search

The solution generated within the shaking step is subsequently improved by several greedy local search procedures. All operators are implemented such that the first improving move is accepted.

First, potential fuel or load violations within a route are handled by adding visits to intermediate facilities. If the distance between two consecutive refueling facility visits exceeds the fuel capacity of the vehicle, the fuel level drops below zero at a certain point. Hence, at least one refueling facility must be visited before this point. Let ϕ denote the position of the last visited refueling facility and σ that of the last vertex reachable from there. The best insertion position is, therefore, determined within the path $\{\phi + 1, \ldots, \sigma + 1\}$. For each possible position, the cost for inserting the closest refueling facility $i \in F_F$ is calculated. The insertion with the lowest cost increase is performed, but in this step insertions leading to feasible solutions are always preferred to infeasible solutions. The insertion of loading facilities is carried out in analogous fashion.

In a second step, we aim to improve the routing by means of the following operators, which are applied in random order. The 2-opt operator replaces two edges by two new ones (Lin 1965). A restricted variant of the Or-Opt exchange (Or 1976) replaces three existing edges by three new ones such that a sequence of three vertices is relocated (Stenger et al. 2013a). The intra-route relocate operator moves a customer to a different position within a route (Savelsbergh 1992). This operator is also defined for moving facilities. Finally, a facility replacement operator evaluates for each facility visit of each route whether replacing the facility visit with a visit to a different facility decreases the routing costs.

This block is followed by an application of a facility removal operator, which aims at removing redundant facility visits. In a final step, we apply two inter-route operators. The inter-route relocate operator moves a customer from its current route to another, and the exchange operator interchanges two customers between two routes (Savelsbergh 1992).

4.4 Penalty determination

Tightly constrained problems often let the local search get stuck in local optima quickly. It is, therefore, reasonable to temporarily allow constraint violations and impose penalty costs on infeasible solutions (see, e.g., Cordeau et al. 1997; Vidal et al. 2012). We define the total penalty costs of a solution as $\text{Cost}^{\text{penalty}} = \delta^C \cdot \upsilon^C + \delta^D \cdot \upsilon^D + \delta^U \cdot \upsilon^U$, with δ^C denoting the penalty factor for capacity violations, υ^C the capacity violation of the solution, δ^D the duration penalty factor, υ^D the duration violation of the solution, δ^U the fuel penalty factor, and υ^U the fuel violation of the solution.

All penalty factors are initialized to δ^0 and dynamically varied within the interval $[\delta^{\min}, \delta^{\max}]$. After a given number of local search iterations η^+ with a violation of the respective constraint, the penalty factor is increased by factor δ^{update}. Analogously, after η^- feasible iterations, the penalty factor is reduced by factor δ^{update}. Preliminary tests showed that choosing different values for η^+ and η^- limits cycling of the local search, especially in small-sized problems.

4.5 Acceptance decision

The solution S'' obtained by the local search procedure is compared to the yet best solution S. If S'' is accepted, it replaces S as initial solution and κ is reset to one. Standard VNS implementations usually model the local search as a simple descent step, i.e., S'' is only accepted if it is improving on S. We use a criterion inspired by simulated annealing (SA) to control solution acceptance. This approach was originally proposed by Hemmelmayr et al. (2009) and also applied in Stenger et al. (2013a).

Improving solutions are always accepted, while non-improving ones are accepted with probability $e^{\frac{-(f(S'')-f(S))}{\vartheta}}$. The temperature parameter ϑ is decreased from its initial value ϑ^0 by factor ϑ^- after every AVNS iteration. After ϵ non-improving main iterations, the current solution is reset to the best solution found so far. Solution diversification is increased by resetting ϑ to ϑ^0 after ξ solution resets.

5 Computational studies

This section presents the computational studies to examine the effectiveness of the AVNS. We perform tests on available instances developed for the routing problems G-VRP (Erdoğan and Miller-Hooks 2012) and VRPIRF (Crevier et al. 2007; Tarantilis et al. 2008), which are both special cases of VRPIS. In addition, we design two sets of benchmark instances for the EVRPRF introduced in Sect. 3.2. A set of small instances is used to assess the quality of our solutions by comparing them to the solutions obtained with the commercial solver CPLEX. A set of large instances is used to prove the ability of our algorithm to deal with realistically sized problems in terms of computational effort. Detailed results are provided to enable a comparison with future methods developed for the EVRPRF. To the best of our knowledge, our numerical studies cover all special cases of the VRPIS investigated in the literature.

Section 5.1 describes the test environment and the parameter setting. The computational results obtained on the special cases of the VRPIS from the literature are presented in Sect. 5.2. Section 5.3 details the generation of EVRPRF instances and the results obtained on this benchmark.

5.1 Experimental environment and parameter settings

The AVNS is implemented as single-thread code in Java. Tests are conducted on a desktop computer with an Intel Core i5 2.67 GHz processor with 4 GB RAM, running Windows 7 Professional. All numerical tests are carried out with the same parameter setting, which was determined during the development and testing of our algorithm.

To determine this parameter setting, we follow the approach described in Ropke and Pisinger (2006). As test instances, we selected a reasonably large subset of the test instances of all VRPIS special cases. Then, we use the parameter setting that we have found during the development of our algorithm as basis for the tuning. Here, we stepwise refine the value of each parameter. In detail, we adjust the value of a single parameter while all remaining parameters are fixed. With every parameter setting, we perform 20 runs on the selected subset of test instances. The setting which produces the best average result is kept and the procedure is repeated with the next parameter. The resulting parameter setting is reported in Table 4.

In detail, the table shows the setting for the number of iterations after which the probabilities are updated (γ), the parameter ρ, which weighs the old weight and the new scores in the weight update of the selection methods within the adaptive mechanism, the initial (δ^0), minimal (δ^{min}) and maximal (δ^{max}) penalty factors, the penalty update factor (δ^{update}), the numbers of iterations after which the penalty costs are decreased (η^-) and increased (η^+), the initial temperature (ϑ^0), the temperature reduction factor (ϑ^-), and the number of resets of the current solution to the best solution found after which the temperature is reset to its initial value (ξ).

To achieve reasonable runtimes on the investigated test instances, we set the maximum number of iterations without improvement (ω) and the number of non-improving iterations after which the current solution is reset to the best solution found (ϵ) as follows: for EVRPRF, we set ω equal to 2,000 and ϵ to 50, for VRPIRF, we use

Table 4 Overview of the final parameter setting of AVNS chosen for the numerical studies

AVNS		Penalties		SA	
γ	30	δ^0	1,000	ϑ^0	50
ρ	0.3	$\delta^{\min}$	10	ϑ^-	0.9995
		$\delta^{\max}$	10,000	ξ	4
		δ^{update}	1.5		
		η^-	2		
		η^+	3		

$\omega = 500$, $\epsilon = 25$. Additional tests on VRPIRF showed that a higher iteration number does not significantly improve the solution quality.

5.2 Experiments on problems with intermediate stops from the literature

To assess the performance of our AVNS, we conduct tests on instances of the special VRPIS cases G-VRP and VRPIRF and compare our results to those presented in the literature.

5.2.1 Green VRP

The benchmark instances designed for the G-VRP (see Sect. 2) consist of five sets. Four sets contain ten instances each comprising 20 customers (which are either uniformly distributed or clustered) and between two and ten refueling facilities. The fifth set represents a case study conducted by the authors and consists of twelve instances involving between 111 and 500 customers and 21 to 28 facilities. Note that, customers that either cannot be served within the maximum route duration or whose service requires visiting more than one refueling stop must be identified and removed from the test instances. The geographical coordinates given in the instances have to be converted to distances between vertices by means of the Haversine formula using an average earth radius of 4,182.45 miles.

Tables 5 and 6 show the results of AVNS on the small and, respectively, large G-VRP instances. We compare our results to those of the MCWS and DBCA heuristics of Erdoğan and Miller-Hooks (2012) and the VNS/TS of Schneider et al. (2014). For each problem instance, we report the problem name and the best-known solution (BKS) provided by either Erdoğan and Miller-Hooks (2012) or Schneider et al. (2014). For the MCWS and DBCA of Erdoğan and Miller-Hooks (2012), we give only the result of the better of the two algorithms for each instance. It was originally determined as the best of multiple runs (L^{best} in the table), but the exact number of runs is not given in the paper. For the VNS/TS of Schneider et al. (2014) and the AVNS, L^{best} corresponds to the best solution found in ten runs. For all algorithms, we further provide the gap of L^{best} to the BKS (Δ^{best}) and the number of served customers (n). For VNS/TS and AVNS, we also display the average computing time of ten runs (t^{avg}) in minutes, for the algorithms of Erdoğan and Miller-Hooks (2012), no runtimes were reported. The runtimes of VNS/TS and AVNS are directly comparable as both algorithms are

Table 5 Results of AVNS on the small-sized G-VRP instances by Erdoğan and Miller-Hooks (2012)

Inst.	BKS	MCWS/DBCA			VNS/TS				AVNS				
		n	L^{best}	$\Delta^{\text{best}}(\%)$	n	L^{best}	$\Delta^{\text{best}}(\%)$	$t^{\text{avg}}(\text{min})$	n	L^{best}	$\Delta^{\text{best}}(\%)$	L^{avg}	$t^{\text{avg}}(\text{min})$
20c3sU1	**1797.49**	20	1,797.51	0.00	20	**1,797.49**	0.00	0.69	20	**1,797.49**	0.00	**1,797.49**	0.16
20c3sU2	**1,574.77**	20	1,613.53	2.46	20	**1,574.77**	0.00	0.64	20	**1,574.78**	0.00	**1,574.78**	0.15
20c3sU3	**1,704.48**	20	1,964.57	15.26	20	**1,704.48**	0.00	0.64	20	**1,704.48**	0.00	**1,704.48**	0.13
20c3sU4	**1,482.00**	20	1,487.15	0.35	20	**1,482.00**	0.00	0.65	20	**1,482.00**	0.00	**1,482.00**	0.17
20c3sU5	**1,689.37**	20	1,752.73	3.75	20	**1,689.37**	0.00	0.67	20	**1,689.37**	0.00	**1,689.37**	0.18
20c3sU6	**1,618.65**	20	1,668.16	3.06	20	**1,618.65**	0.00	0.67	20	**1,618.65**	0.00	**1,618.65**	0.15
20c3sU7	**1,713.66**	20	1,730.45	0.98	20	**1,713.66**	0.00	0.64	20	**1,713.66**	0.00	**1,713.66**	0.19
20c3sU8	**1,706.50**	20	1,718.67	0.71	20	**1,706.50**	0.00	0.67	20	**1,706.50**	0.00	**1,706.50**	0.16
20c3sU9	**1,708.81**	20	1,714.43	0.33	20	**1,708.81**	0.00	0.66	20	**1,708.82**	0.00	**1,708.82**	0.19
20c3sU10	**1,181.31**	20	1,309.52	10.85	20	**1,181.31**	0.00	0.64	20	**1,181.31**	0.00	**1,181.31**	0.23
20c3sC1	**1,173.57**	20	1,300.62	10.83	20	**1,173.57**	0.00	0.62	20	**1,173.57**	0.00	**1,173.57**	0.38
20c3sC2	**1,539.97**	19	1,553.53	0.88	19	**1,539.97**	0.00	0.58	19	**1,539.97**	0.00	**1,539.97**	0.21
20c3sC3	**880.20**	12	1,083.12	23.05	12	**880.20**	0.00	0.25	12	**880.20**	0.00	**880.20**	0.15
20c3sC4	**1,059.35**	18	1,091.78	3.06	18	**1,059.35**	0.00	0.53	18	**1,059.35**	0.00	1,077.71	0.23
20c3sC5	**2,156.01**	19	2,190.68	1.61	19	**2,156.01**	0.00	0.60	19	**2,156.01**	0.00	**2,156.01**	0.14
20c3sC6	**2,758.17**	17	2,883.71	4.55	17	**2,758.17**	0.00	0.71	17	**2,758.17**	0.00	**2,758.17**	0.14
20c3sC7	**1,393.99**	6	1,701.40	22.05	6	**1,393.99**	0.00	0.18	6	**1,393.99**	0.00	**1,393.99**	0.04
20c3sC8	**3,139.72**	18	3,319.74	5.73	18	**3,139.72**	0.00	0.62	18	**3,139.72**	0.00	**3,139.72**	0.08
20c3sC9	**1,799.94**	19	1,811.05	0.62	19	**1,799.94**	0.00	0.60	19	**1,799.94**	0.00	**1,799.94**	0.16
20c3sC10	**2,583.42**	15	2,644.11	2.35	15	**2,583.42**	0.00	0.45	15	**2,583.42**	0.00	2,600.39	0.09
S1_2i6s	**1,578.12**	20	1,614.15	2.28	20	**1,578.12**	0.00	0.71	20	**1,578.12**	0.00	**1,578.12**	0.16
S1_4i6s	**1,397.27**	20	1,541.46	10.32	20	**1,397.27**	0.00	0.75	20	**1,397.27**	0.00	**1,397.27**	0.16

Table 5 continued

Inst.	BKS	MCWS/DBCA			VNS/TS				AVNS				
		n	L^{best}	$\Delta^{\text{best}}(\%)$	n	L^{best}	$\Delta^{\text{best}}(\%)$	$t^{\text{avg}}(\text{min})$	n	L^{best}	$\Delta^{\text{best}}(\%)$	L^{avg}	$t^{\text{avg}}(\text{min})$
S1_6i6s	**1,560.49**	20	1,616.20	3.57	20	**1,560.49**	0.00	0.73	20	**1,560.49**	0.00	**1,560.49**	0.20
S1_8i6s	**1,692.32**	20	1,882.54	11.24	20	**1,692.32**	0.00	0.74	20	**1,692.32**	0.00	**1,692.32**	0.17
S1_10i6s	**1,173.48**	20	1,309.52	11.59	20	**1,173.48**	0.00	0.71	20	**1,173.48**	0.00	**1,173.48**	0.24
S2_2i6s	**1,633.10**	20	1,645.80	0.78	20	**1,633.10**	0.00	0.75	20	**1,633.10**	0.00	**1,633.10**	0.19
S2_4i6s	**1,505.06**	19	1,505.06	0.00	19	1,532.96	1.85	0.88	19	**1,505.07**	0.00	**1,505.07**	0.14
S2_6i6s	**2,431.33**	20	3,115.10	28.12	20	**2,431.33**	0.00	0.78	20	**2,431.33**	0.00	**2,431.33**	0.13
S2_8i6s	**2,158.35**	16	2,722.55	26.14	16	**2,158.35**	0.00	0.57	16	**2,158.35**	0.00	**2,158.35**	0.09
S2_10i6s	1,958.46	16	1,995.62	1.90	17	1,958.46	0.00	0.61	16	1585.46	−19.05	1,585.46	0.15
S1_4i2s	**1,582.20**	20	1,582.20	0.00	20	**1,582.21**	0.00	0.63	20	**1,582.21**	0.00	**1,582.21**	0.13
S1_4i4s	**1,460.09**	20	1,580.52	8.25	20	**1,460.09**	0.00	0.68	20	**1,460.09**	0.00	**1,460.09**	0.16
S1_4i6s	**1,397.27**	20	1,541.46	10.32	20	**1,397.27**	0.00	0.75	20	**1,397.27**	0.00	**1,397.27**	0.16
S1_4i8s	**1,397.27**	20	1,561.29	11.74	20	**1,397.27**	0.00	0.82	20	**1,397.27**	0.00	**1,397.27**	0.17
S1_4i10s	**1,396.02**	20	1,529.73	9.58	20	**1,396.02**	0.00	0.85	20	**1,396.02**	0.00	**1,396.02**	0.23
S2_4i2s	**1,059.35**	18	1,117.32	5.47	18	**1,059.35**	0.00	0.51	18	**1,059.35**	0.00	1,069.42	0.23
S2_4i4s	**1,446.08**	19	1,522.72	5.30	19	**1,446.08**	0.00	0.60	19	**1,446.08**	0.00	1,449.17	0.21
S2_4i6s	**1,434.14**	20	1,730.47	20.66	20	**1,434.14**	0.00	0.69	20	**1,434.14**	0.00	1,445.35	0.20
S2_4i8s	**1,434.14**	20	1,786.21	24.55	20	**1,434.14**	0.00	0.75	20	**1,434.14**	0.00	**1,434.14**	0.20
S2_4i10s	**1,434.13**	20	1,729.51	20.60	20	**1,434.13**	0.00	0.78	20	**1,434.13**	0.00	1,455.31	0.24
Avg.				8.12			0.05	0.65			−0.48		0.17

The results are compared to those of the MCWS/DCBA of Erdoğan and Miller-Hooks (2012) and the VNS/TS of Schneider et al. (2014). BKS denotes the previously best-known solution. L^{best} denotes the best solution found (for VNS/TS and AVNS in ten runs), Δ^{best} the gap to the BKS and n the number of served customers. For VNS/TS and AVNS, the average computing time in minutes is given (t^{avg}). For AVNS, we additionally report the average solution quality of ten runs (L^{avg}). Numbers in bold indicate the best solution found

Table 6 Results of AVNS on the large-sized G-VRP instances by Erdoğan and Miller-Hooks (2012)

Inst.	BKS	MCWS/DBCA			VNS/TS				AVNS				
		n	L^{best}	$\Delta^{\text{best}}(\%)$	n	L^{best}	$\Delta^{\text{best}}(\%)$	$t^{\text{avg}}(\text{min})$	n	L^{best}	$\Delta^{\text{best}}(\%)$	L^{avg}	$t^{\text{avg}}(\text{min})$
111c_21s	4,797.15	109	5,626.64	17.29	109	4,797.15	0.00	21.76	109	**4,770.47**	−0.56	4,791.53	1.78
111c_22s	4,802.16	109	5,610.57	16.83	109	4,802.16	0.00	23.56	109	**4,776.81**	−0.53	4,797.31	1.94
111c_24s	4,786.96	109	5,412.48	13.07	109	4,786.96	0.00	21.90	109	**4,767.14**	−0.41	4,790.84	2.16
111c_26s	4,778.62	109	5,408.38	13.18	109	4,778.62	0.00	25.12	109	**4,767.14**	−0.24	4,782.60	2.04
111c_28s	4,799.15	109	5,331.93	11.10	109	4,799.15	0.00	24.17	109	**4,765.52**	−0.70	4,781.26	1.73
200c	8,963.46	190	10,413.59	16.18	192	8,963.46	0.00	76.65	192	**8,886.00**	−0.86	8,970.14	3.61
250c	10,800.18	235	11,886.61	10.06	237	10,800.18	0.00	120.90	237	**10,487.15**	−2.90	10,531.20	3.67
300c	12,594.77	281	14,229.92	12.98	283	12,594.77	0.00	182.23	283	**12,374.49**	−1.75	12,514.78	4.94
350c	14,323.02	329	16,460.30	14.92	329	14,323.02	0.00	232.03	329	**14,103.66**	−1.53	14,271.56	7.11
400c	16,850.21	378	19,099.04	13.35	378	16,850.21	0.00	305.12	378	**16,697.21**	−0.91	16,839.23	12.70
450c	18,521.23	424	21,854.17	18.00	424	18,521.23	0.00	525.52	424	**18,310.60**	−1.14	18,512.47	13.19
500c	21,170.90	471	24,517.08	15.81	471	21,170.90	0.00	356.01	471	**20,609.67**	−2.65	20,874.50	19.51
Avg.				14.40			0.00	159.58			−1.18		6.20

The results are compared to those of the MCWS/DCBA of Erdoğan and Miller-Hooks (2012) and the VNS/TS of Schneider et al. (2014). BKS denotes the previously best-known solution. L^{best} denotes the best solution found (for VNS/TS and AVNS in ten runs), Δ^{best} the gap to the BKS and n the number of served customers. For VNS/TS and AVNS, the average computing time in minutes is given (t^{avg}). For AVNS, we additionally report the average solution quality of ten runs (L^{avg}). Numbers in bold indicate the best solution found coded in Java and executed on the same computer. Moreover, we report the average solution quality of the ten runs for the AVNS (L^{avg}). Finally, averages of the runtimes and relative gaps to the BKS over the complete set of instances are given at the end of the table.

For the small G-VRP instances (Table 5), AVNS finds the best-known solution (BKS) for all instances. Thus, the AVNS is able to clearly outperform the methods of Erdoğan and Miller-Hooks (2012), even if for each instance only the best result provided by either MCWS or DBCA is considered. For the comparison with the VNS/TS of Schneider et al. (2014), note that the large gap of -19.05 % to the BKS for instance S2_10i6s is not meaningful because Schneider et al. (2014) identify one more customer to be reachable for this instance. However, even disregarding this instance, AVNS yields better solution quality than VNS/TS for one instance and is able to match the quality for all other instances. Moreover, compared to the VNS/TS approach, it runs nearly four times as fast on average. The results further prove the robustness of the developed algorithm: For the large majority of instances, the average solution quality of the ten runs L^{avg} is equal to the quality of the best run L^{best}; for the remaining instances, the gap is quite small.

On the large-sized G-VRP instances (Table 6), the AVNS algorithm finds new best-known solution solutions for all instances and achieves an average gap to the previous BKS of more than 1 %. Moreover, the speed of the AVNS is remarkable, using approximately 4 % of the runtime of the VNS/TS of Schneider et al. (2014).

5.2.2 VRP with intermediate replenishment facilities

The VRPIRF (see Sect. 2) considers intermediate replenishment facilities for the goods to be delivered. We run tests on two instance sets. The set of Crevier et al. (2007) comprises 22 instances consisting of 48–216 customers, three to six intermediate facilities and four to six available vehicles. The customers are clustered around the facilities. The set of Tarantilis et al. (2008) contains 54 instances with 50–175 customers, three to eight facilities and two to eight vehicles.

Tables 7 and 8 show the results of our AVNS on the instances of Crevier et al. (2007), compared to the results of Crevier, Cordeau, and Laporte (2007) (CCL), those of Tarantilis, Zachariadis, and Kiranoudis (2008) (TZK), of the VNS/TS of Schneider et al. (2014), and the VNS of Hemmelmayr, Doerner, Hartl, and Rath (2013) (HDHR). For each instance, we report the instance name and the previously known BKS as determined by the four comparison methods. In Table 7, we report for all algorithms the average solution quality of ten runs (L^{avg}), the gap of the average solution to the BKS (Δ^{avg}) and the average computing time in minutes (t^{avg}). Finally, averages of the runtimes and the gaps to the BKS over the complete set of instances are given at the end of the table. Note that, a direct comparison of runtimes is only valid for AVNS and VNS/TS. The other methods are partly coded in different programming languages and were run on different platforms to obtain the reported computation times. The best solution quality obtained by any of the methods on each instance is indicated in bold.

In Table 8, we report the best solution found (L^{best}) and the gap of the best solution to the BKS (Δ^{best}). The best solutions reported by Crevier et al. (2007), Schneider et al. (2014), Hemmelmayr et al. (2013) and our AVNS are based on ten runs, those

Table 7 Average solution quality of AVNS on the VRPIRF instances of Crevier et al. (2007)

Inst.	BKS	CCL			TZK			VNS/TS			HDHR			AVNS		
		L^{avg}	Δ^{avg} (%)	t^{avg} (min)	L^{avg}	Δ^{avg} (%)	t^{avg} (min)	L^{avg}	Δ^{avg} (%)	t^{avg} (min)	L^{avg}	Δ^{avg} (%)	t^{avg} (min)	L^{avg}	Δ^{avg} (%)	t^{avg} (min)
a1	**1,179.79**	1,211.28	2.67	4.58	1,189.70	0.84	3.40	1,197.59	1.51	1.82	1,180.57	0.07	1.42	1,184.57	0.41	0.64
b1	**1,217.07**	1,232.67	1.28	9.17	1,225.08	0.66	7.80	1,226.80	0.80	7.14	**1,217.07**	0.00	6.39	1,218.21	0.09	4.19
c1	**1,866.76**	1,893.01	1.41	36.22	1,898.92	1.72	34.20	1,925.21	3.13	33.93	1,867.96	0.06	20.40	1,925.41	3.14	32.98
d1	**1,059.43**	1,076.31	1.59	8.55	1,064.29	0.46	5.90	1,063.08	0.35	1.82	**1,059.43**	0.00	1.57	1,061.50	0.20	0.55
e1	**1,309.12**	1,311.60	0.19	13.52	1,309.12	0.00	8.70	1,343.90	2.66	7.29	**1,309.12**	0.00	6.22	1,312.75	0.28	5.08
f1	**1,570.41**	1,601.54	1.98	41.41	1,585.83	0.98	38.80	1,619.80	3.14	34.61	1,573.05	0.17	25.60	1,601.40	1.97	34.99
g1	**1,181.13**	1,202.00	1.77	55.22	1,190.21	0.77	5.80	1,190.72	0.81	4.21	1,183.32	0.19	3.38	1,183.75	0.22	1.69
h1	**1,545.50**	1,598.51	3.43	32.07	1,577.54	2.07	11.10	1,582.33	2.38	18.03	1,548.61	0.20	14.61	1,567.22	1.41	14.08
i1	**1,922.18**	1,976.11	2.81	51.01	1,956.17	1.77	42.50	2,004.35	4.27	45.62	1,923.52	0.07	33.58	1,974.97	2.75	35.11
j1	**1,115.78**	1,161.77	4.12	58.90	1,128.86	1.17	5.50	1,120.65	0.44	4.24	**1,115.78**	0.00	2.78	1,116.82	0.09	2.02
k1	**1,576.36**	1,618.45	2.67	64.61	1,591.74	0.98	12.10	1,601.67	1.61	18.11	1,577.96	0.10	14.56	1,600.42	1.53	10.74
l1	**1,863.28**	1,917.08	2.89	104.27	1,904.39	2.21	51.40	1,932.82	3.73	46.14	1,869.70	0.34	35.48	1,916.07	2.83	40.59
Avg.			2.23	39.96		1.14	18.93		2.07	18.58		0.10	13.83		1.24	15.22
a2	**997.94**	1,005.16	0.72	6.40				1,002.58	0.47	1.80	**997.94**	0.00	1.23	**997.94**	0.00	0.72
b2	**1,291.19**	1,333.20	3.25	14.70				1,324.84	2.61	7.35	**1,291.19**	0.00	6.41	1,300.42	0.72	4.83
c2	**1,715.60**	1,792.46	4.48	61.70				1,760.81	2.63	18.05	1,715.84	0.01	15.01	1,741.55	1.51	18.32
d2	**1,856.84**	1,898.21	2.23	40.50				1,908.34	2.77	35.10	1,860.92	0.22	30.14	1,903.15	2.49	30.64
e2	**1,919.38**	1,995.75	3.98	73.80				1,993.47	3.86	59.12	1,922.81	0.18	49.31	1,957.80	2.00	41.60
f2	**2,230.32**	2,312.15	3.67	162.20				2,325.34	4.26	89.86	2,233.43	0.14	71.24	2,313.08	3.71	42.80
g2	**1,152.92**	1,185.93	2.86	29.50				1,161.50	0.74	4.14	1,153.17	0.02	3.71	1,158.21	0.46	2.20
h2	**1,575.28**	1,611.75	2.32	160.80				1,610.48	2.23	18.35	**1,575.28**	0.00	15.66	1,586.24	0.70	21.20

Table 7 continued

Inst.	BKS	CCL			TZK			VNS/TS			HDHR			AVNS		
		L^{avg}	Δ^{avg} (%)	t^{avg} (min)	L^{avg}	Δ^{avg} (%)	t^{avg} (min)	L^{avg}	Δ^{avg} (%)	t^{avg} (min)	L^{avg}	Δ^{avg} (%)	t^{avg} (min)	L^{avg}	Δ^{avg} (%)	t^{avg} (min)
i2	**1,919.74**	1,998.20	4.09	322.40				1,969.98	2.62	47.58	1,922.24	0.13	41.92	1,971.27	2.68	41.10
j2	**2,247.70**	2,325.18	3.45	256.90				2,330.34	3.68	91.30	2,250.21	0.11	73.38	2,303.67	2.49	41.93
Avg.			3.10	112.89					2.59	37.27		0.08	30.80		1.68	24.53
Tot. Avg.			2.63	73.11					2.30	27.07		0.09	21.55		1.44	19.46

Results are compared to those of Crevier, Cordeau, and Laporte (2007) (CCL), Tarantilis, Zachariadis, and Kiranoudis (2008) (TZK), of the VNS/TS of Schneider et al. (2014) and the VNS of Hemmelmayr, Doerner, Hartl, and Rath (2013) (HDHR). L^{avg} denotes the average solution found in ten runs, Δ^{avg} the gap to BKS and t^{avg} the average computing time in minutes. Numbers in bold indicate the best solution found

Table 8 Comparison of the results on the VRPIRF instances of Crevier et al. (2007) to those of TZK, CCL, VNS/TS and HDHR based on the best run

Inst.	BKS	CCL		TZK		VNS/TS		HDHR		AVNS		$\overline{\textbf{AVNS}}$	
		L^{best}	$\Delta^{\text{best}}(\%)$	$L^{\text{best}}*$	$\Delta^{\text{best}}*(\%)$	L^{best}	$\Delta^{\text{best}}(\%)$	L^{best}	$\Delta^{\text{best}}(\%)$	L^{best}	$\Delta^{\text{best}}(\%)$	L	$\Delta L(\%)$
a1	**1,179.79**	1,203.39	2.00	**1,179.79**	0.00	**1,179.79**	0.00	**1,179.79**	0.00	**1,179.79**	0.00	**1,179.79**	0.00
b1	**1,217.07**	**1,217.07**	0.00	**1,217.07**	0.00	**1,217.07**	0.00	**1,217.07**	0.00	**1,217.07**	0.00	**1,217.07**	0.00
c1	**1,866.76**	1,888.22	1.15	1,883.05	0.87	1,897.30	1.64	**1,866.76**	0.00	1,893.53	1.43	1,882.46	0.84
d1	**1,059.43**	**1,059.43**	0.00	**1,059.43**	0.00	1,060.10	0.06	**1,059.43**	0.00	**1,059.43**	0.00	**1,059.43**	0.00
e1	**1,309.12**	**1,309.12**	0.00	**1,309.12**	0.00	**1,309.12**	0.00	**1,309.12**	0.00	**1,309.12**	0.00	**1,309.12**	0.00
f1	**1,570.41**	1,592.25	1.39	1,572.17	0.11	1,584.06	0.87	**1,570.41**	0.00	1,579.89	0.60	1,577.63	0.46
g1	**1,181.13**	1,190.93	0.83	**1,181.13**	0.00	1,181.99	0.07	**1,181.13**	0.00	**1,181.13**	0.00	**1,181.13**	0.00
h1	**1,545.50**	1,566.75	1.37	1,547.24	0.11	1,566.19	1.34	**1,545.50**	0.00	1,555.52	0.65	1,553.75	0.53
i1	**1,922.18**	1,945.73	1.23	1,925.99	0.20	1,953.39	1.62	**1,922.18**	0.00	1,956.70	1.80	1,934.08	0.62
j1	**1,115.78**	1,144.41	2.57	1,117.20	0.13	**1,115.78**	0.00	**1,115.78**	0.00	**1,115.78**	0.00	**1,115.78**	0.00
k1	**1,576.36**	1,586.92	0.67	1,580.39	0.26	1,586.64	0.65	**1,576.36**	0.00	1,591.81	0.98	1,577.98	0.10
l1	**1,863.28**	1,897.74	1.85	1,880.60	0.93	1,902.72	2.12	**1,863.28**	0.00	1,907.15	2.35	1,894.69	1.69
Avg.			1.09		0.22		0.70		0.00		0.65		0.35
a2	**997.94**	1,000.24	0.23			**997.94**	0.00	**997.94**	0.00	**997.94**	0.00	**997.94**	0.00
b2	**1,291.19**	1,307.28	1.25			1,301.21	0.78	**1,291.19**	0.00	**1,291.19**	0.00	**1,291.19**	0.00
c2	**1,715.60**	1,751.45	2.09			1,732.19	0.97	**1,715.60**	0.00	1,730.14	0.85	**1,715.60**	0.00
d2	**1,856.84**	1,877.03	1.09			1,892.62	1.93	**1,856.84**	0.00	1,878.89	1.19	1,874.12	0.93
e2	**1,919.38**	1,974.13	2.85			1,940.52	1.10	**1,919.38**	0.00	1,943.61	1.26	1,937.84	0.96
f2	**2,230.32**	2,298.51	3.06			2,292.40	2.78	**2,230.32**	0.00	2,292.84	2.80	2,268.54	1.71
g2	**1,152.92**	1,162.58	0.84			1,158.21	0.46	**1,152.92**	0.00	1,158.21	0.46	**1,152.92**	0.00

Table 8 continued

Inst.	BKS	CCL		TZK		VNS/TS		HDHR		AVNS		$\overline{\textbf{AVNS}}$	
		L^{best}	$\Delta^{\text{best}}(\%)$	$L^{\text{best}}*$	$\Delta^{\text{best}}*(\%)$	L^{best}	$\Delta^{\text{best}}(\%)$	L^{best}	$\Delta^{\text{best}}(\%)$	L^{best}	$\Delta^{\text{best}}(\%)$	L	$\Delta L(\%)$
h2	**1,575.28**	1,593.40	1.15			1,597.41	1.41	**1,575.28**	0.00	1,576.86	0.10	1,576.86	0.10
i2	**1,919.74**	1,978.70	3.07			1,934.09	0.75	**1,919.74**	0.00	1,945.24	1.33	1,944.74	1.30
j2	**2,247.70**	2,303.01	2.46			2,293.40	2.03	**2,247.70**	0.00	2,281.86	1.52	2,281.86	1.52
Avg.			1.81				1.22		0.00		0.95		0.65
Tot. Avg.			1.42				0.94		0.00		0.79		0.49

Note that, the value given in column $L^{\text{best}}*$ for TZK corresponds to the best solution ever found with the final parameter setting. For all other methods, L^{best} refers to the best out of ten runs. Δ^{best} denotes the gap to BKS. We also provide the best solutions found during our overall testing in column $\overline{\textbf{AVNS}}$. Numbers in bold indicate the best solution found

Table 9 Comparison of the AVNS results on the VRPIRF instances by Tarantilis et al. (2008) to those of TZK, VNS/TS and HDHR. BKS denotes the previously best-known solution

		TZK				VNS/TS				HDHR				AVNS				$\overline{\text{AVNS}}$	
Instance	BKS	L^{best*}	Δ^{best*}	Δ^{avg} (%)	t^{avg} (min)	L^{best}	Δ^{best} (%)	Δ^{avg} (%)	t^{avg} (min)	L^{best}	Δ^{best} (%)	Δ^{avg} (%)	t^{avg} (min)	L^{best}	Δ^{best} (%)	Δ^{avg} (%)	t^{avg} (min)	L	ΔL(%)
50c3d2v	**2209.83**	**2209.83**	0.00	2.27	2.85	**2209.83**	0.00	0.10	1.82	**2209.83**	0.00	0.00	6.62	**2209.83**	0.00	0.00	0.80	**2209.83**	0.00
50c3d4v	**2368.33**	**2368.33**	0.00	2.18	2.23	**2368.33**	0.00	0.89	1.84	**2368.33**	0.00	0.00	1.76	**2368.33**	0.00	0.00	0.83	**2368.33**	0.00
50c3d6v	**2999.29**	3000.88	0.05	2.18	2.74	**2999.29**	0.00	0.81	1.88	**2999.29**	0.00	0.00	1.15	**2999.29**	0.00	0.01	0.52	**2999.29**	0.00
50c5d2v	**2608.25**	**2608.25**	0.00	2.87	1.54	**2608.25**	0.00	1.01	2.00	**2608.25**	0.00	0.00	7.64	**2608.25**	0.00	0.00	0.73	**2608.25**	0.00
50c5d4v	**3086.58**	**3086.58**	0.00	1.23	2.07	**3086.58**	0.00	0.00	1.98	**3086.58**	0.00	0.00	2.04	**3086.58**	0.00	0.11	0.88	**3086.58**	0.00
50c5d6v	**3548.88**	3552.00	0.09	0.99	3.04	3552.00	0.09	0.37	2.02	**3548.88**	0.00	0.23	1.28	**3548.88**	0.00	0.27	0.41	**3548.88**	0.00
50c7d2v	**3353.08**	**3353.08**	0.00	2.37	3.16	**3353.83**	0.02	3.10	2.38	**3353.08**	0.00	0.00	14.11	3370.85	0.53	1.29	0.86	3370.85	0.53
50c7d4v	**3380.27**	3381.57	0.04	2.67	3.36	**3380.27**	0.00	0.58	2.10	**3380.27**	0.00	0.00	4.76	3398.28	0.53	0.93	1.25	3398.28	0.53
50c7d6v	**4074.44**	4097.80	0.57	0.83	3.42	**4074.44**	0.00	0.95	2.07	**4074.44**	0.00	0.36	1.50	**4074.44**	0.00	0.24	0.41	**4074.44**	0.00
75c3d2v	**2678.80**	**2678.80**	0.00	0.57	4.50	2692.76	0.52	1.68	4.67	**2678.80**	0.00	0.00	23.79	**2678.80**	0.00	0.53	2.89	**2678.8**	0.00
75c3d4v	**2746.74**	**2746.74**	0.00	1.95	3.38	**2746.74**	0.00	0.17	4.33	**2746.74**	0.00	0.00	7.13	**2746.74**	0.00	0.00	3.86	**2746.74**	0.00
75c3d6v	**3393.89**	3454.71	1.79	3.11	4.89	3448.64	1.61	2.21	4.34	**3393.89**	0.00	0.00	3.31	3419.49	0.75	1.33	1.97	3419.49	0.75
75c5d2v	**3373.69**	**3373.69**	0.00	2.98	3.29	3386.64	0.38	2.34	5.09	**3373.69**	0.00	0.00	29.45	3374.76	0.03	1.18	3.27	**3373.69**	0.00
75c5d4v	**3553.46**	3568.35	0.42	2.86	3.54	3569.82	0.46	1.05	4.42	**3553.46**	0.00	0.21	4.09	**3553.46**	0.00	1.16	2.62	**3553.46**	0.00
75c5d6v	**4184.65**	4198.61	0.33	2.00	4.18	4215.30	0.73	2.38	4.55	**4184.65**	0.00	0.00	3.17	4193.86	0.22	0.25	1.56	4193.86	0.22
75c7d2v	**3569.02**	**3569.02**	0.00	2.41	5.38	3581.32	0.34	1.63	5.06	**3569.02**	0.00	0.00	25.84	**3569.02**	0.00	0.24	2.47	**3569.02**	0.00
75c7d4v	**3822.10**	3830.43	0.22	2.35	5.51	3830.43	0.22	1.92	4.61	**3822.10**	0.00	1.07	5.43	**3822.10**	0.00	0.13	2.60	**3822.1**	0.00
75c7d6v	**4239.76**	**4239.76**	0.00	2.02	4.29	4244.35	0.11	0.75	4.80	4263.24	0.55	0.55	2.55	4263.24	0.55	0.97	1.54	**4239.76**	0.00
100c3d3v	**3123.51**	**3123.51**	0.00	1.10	7.01	3127.65	0.13	2.33	7.94	**3123.51**	0.00	0.00	14.66	3180.66	1.83	2.51	9.64	3176.39	1.69
100c3d5v	**3548.44**	3552.50	0.11	2.48	7.31	3548.75	0.01	0.30	7.62	**3548.75**	0.01	0.10	4.58	**3548.44**	0.00	0.06	4.18	**3548.44**	0.00
100c3d7v	**4235.31**	**4239.83**	0.11	0.93	6.62	4268.34	0.78	2.45	7.92	**4235.31**	0.00	0.28	4.03	4274.72	0.93	1.51	3.49	4274.72	0.93
100c5d3v	**4053.95**	**4053.95**	0.00	1.06	7.88	**4053.95**	0.00	1.58	8.49	**4053.95**	0.00	0.00	24.23	**4053.95**	0.00	0.57	10.59	**4053.95**	0.00
100c5d5v	**4413.17**	**4413.17**	0.00	2.69	7.20	4424.81	0.26	5.52	7.70	**4413.17**	0.00	0.00	3.74	**4413.17**	0.00	0.00	2.62	**4413.17**	0.00
100c5d7v	**5142.52**	5148.98	0.13	0.69	7.72	**5142.52**	0.00	0.28	7.93	**5142.52**	0.00	0.00	2.67	**5142.52**	0.00	0.15	2.09	**5142.52**	0.00
100c7d3v	**4207.79**	4216.47	0.21	0.82	8.53	4242.38	0.82	1.83	8.87	**4207.79**	0.00	0.00	30.27	**4207.79**	0.00	0.40	14.83	**4207.79**	0.00
100c7d5v	**4414.69**	4462.51	1.08	2.45	8.79	4448.15	0.76	1.76	8.00	**4414.69**	0.00	0.59	4.60	4440.36	0.58	1.17	5.12	4439.55	0.56
100c7d7v	**4847.79**	4897.47	1.02	2.59	8.35	4916.62	1.42	4.89	8.10	**4847.79**	0.00	0.42	3.33	4850.21	0.05	1.20	1.90	4850.21	0.05
125c4d3v	**3916.02**	3920.05	0.10	1.28	8.73	3966.61	1.29	3.73	13.23	**3916.02**	0.00	0.00	31.29	**3916.02**	0.00	0.75	11.13	**3916.02**	0.00
125c4d5v	**4308.44**	4315.68	0.17	1.47	9.00	**4308.44**	0.00	0.45	12.33	**4308.44**	0.00	0.00	7.62	**4308.44**	0.00	0.00	5.89	**4308.44**	0.00
125c4d7v	4668.77	4763.49	2.03	3.54	8.40	4694.32	0.55	2.61	12.54	4769.75	2.16	2.19	4.77	**4664.38**	**-0.09**	2.67	3.52	**4664.38**	**-0.09**
125c6d3v	**4063.25**	**4064.20**	0.02	0.80	9.19	4117.41	1.33	3.42	13.56	**4063.25**	0.00	0.00	28.08	**4063.25**	0.00	1.37	12.51	**4063.25**	0.00
125c6d5v	**4760.47**	4826.71	1.39	4.13	8.33	4786.74	0.55	1.61	13.09	**4760.47**	0.00	0.09	8.79	4780.61	0.42	1.47	7.53	4770.52	0.21
125c6d7v	**5164.03**	5325.28	3.12	5.86	9.18	5221.52	1.11	2.55	12.89	**5164.03**	0.00	0.70	3.78	5358.54	3.77	4.44	3.57	5203.67	0.77
125c8d3v	**4534.14**	4553.28	0.42	3.16	10.23	4574.82	0.90	1.94	14.98	**4534.14**	0.00	0.16	39.65	4545.44	0.25	1.11	23.82	4541.82	0.17
125c8d5v	**4947.00**	5045.65	1.99	3.40	9.64	**4958.26**	0.23	3.88	13.38	**4947.00**	0.00	1.92	10.33	5071.39	2.51	3.17	10.79	4954.11	0.14
125c8d7v	**5335.79**	5416.96	1.52	2.16	9.34	5397.86	1.16	2.59	13.38	**5335.79**	0.00	0.07	5.74	5372.92	0.70	1.59	4.94	5359.52	0.44

Table 9 continued

Instance	BKS	TZK				VNS/TS				HDHR				AVNS				$\overline{\text{AVNS}}$	
		L^{best*}	Δ^{best*}	$\Delta^{avg}(\%)$	$t^{avg}(\text{min})$	L^{best}	$\Delta^{best}(\%)$	$\Delta^{avg}(\%)$	$t^{avg}(\text{min})$	L^{best}	$\Delta^{best}(\%)$	$\Delta^{avg}(\%)$	$t^{avg}(\text{min})$	L^{best}	$\Delta^{best}(\%)$	$\Delta^{avg}(\%)$	$t^{avg}(\text{min})$	L	$\Delta L(\%)$
150c4d3v	**4049.48**	**4049.48**	0.00	0.01	9.71	4072.95	0.58	3.05	21.84	**4049.48**	0.00	0.13	46.10	4100.31	1.26	1.79	37.67	4095.45	1.14
150c4d5v	4618.23	4638.72	0.44	1.89	8.19	4622.77	0.10	1.05	19.11	**4618.23**	0.00	0.16	14.52	**4612.05**	-0.13	0.49	15.92	**4612.05**	-0.13
150c4d7v	**5118.41**	5176.50	1.13	2.45	8.00	5163.02	0.87	1.70	19.06	**5118.41**	0.00	0.45	7.51	5147.84	0.57	1.54	6.83	5138.05	0.38
150c6d3v	**4057.09**	**4057.09**	0.00	0.15	9.96	4066.71	0.24	1.45	22.07	**4057.09**	0.00	0.02	48.35	4062.22	0.13	1.07	36.59	4062.22	0.13
150c6d5v	**4855.29**	**4872.08**	0.35	0.89	10.23	4931.13	1.56	2.77	21.16	**4855.29**	0.00	0.09	30.57	4943.20	1.81	3.94	26.30	4928.68	1.51
150c6d7v	**5695.26**	5768.29	1.28	3.89	10.73	5840.52	2.55	3.31	20.40	**5695.26**	0.00	0.01	10.19	5722.09	0.47	2.98	11.37	5722.09	0.47
150c8d3v	**4641.30**	4653.90	0.27	2.07	10.18	4689.13	1.03	3.94	22.67	**4641.30**	0.00	0.00	51.81	4708.21	1.44	3.30	34.50	4652.13	0.23
150c8d5v	**5065.11**	5113.77	0.96	2.07	11.62	5116.55	1.02	2.67	19.60	**5065.11**	0.00	0.00	11.48	**5065.11**	0.00	0.68	16.01	**5065.11**	0.00
150c8d7v	**5605.83**	5665.23	1.06	1.06	12.01	5648.32	0.76	1.56	19.67	**5605.83**	0.00	0.16	5.06	5620.73	0.27	1.33	8.44	5613.07	0.13
175c4d4v	**4703.25**	**4706.76**	0.07	1.68	21.74	4720.36	0.36	1.67	28.69	**4703.25**	0.00	0.13	42.08	4747.63	0.94	1.55	35.10	4730.6	0.58
175c4d6v	**4816.54**	**4835.64**	0.40	2.99	23.01	4863.88	0.98	2.91	26.71	**4816.54**	0.00	0.01	17.43	4888.01	1.48	4.81	20.26	4888.01	1.48
175c4d8v	**5830.63**	5943.28	1.93	3.49	18.40	5853.90	0.40	1.78	27.35	**5830.63**	0.00	0.36	13.99	5836.97	0.11	2.08	19.13	5836.97	0.11
175c6d4v	**4999.43**	5025.51	0.52	2.17	21.51	5011.01	0.23	2.43	29.28	**4999.43**	0.00	0.22	55.83	5000.48	0.02	1.25	35.82	5000.48	0.02
175c6d6v	**5292.88**	5431.34	2.62	2.73	22.54	5382.57	1.69	3.60	27.43	**5292.88**	0.00	0.10	14.38	5446.37	2.90	3.82	13.49	5323.95	0.59
175c6d8v	6037.20	6090.01	0.87	2.16	25.81	6066.10	0.48	1.97	27.97	6037.20	0.00	0.06	7.09	6045.58	0.14	1.54	10.19	**6025.08**	-0.20
175c8d4v	**5747.73**	5878.58	2.28	4.93	24.90	5840.25	1.61	3.61	29.83	**5747.73**	0.00	0.01	44.23	5866.54	2.07	2.92	30.52	5866.54	2.07
175c8d6v	**5925.40**	5989.63	1.08	3.91	25.21	5968.99	0.74	3.35	27.78	**5925.40**	0.00	0.10	15.21	5985.42	1.01	4.48	12.50	5940.74	0.26
175c8d8v	6773.77	6943.63	2.51	4.45	26.70	6840.04	0.98	4.15	27.98	6773.77	0.00	0.69	6.77	6792.15	0.27	3.75	8.58	**6766.54**	-0.11
Average			0.64	2.25	9.54		0.59	2.09	12.79		0.05	0.22	15.19		0.52	1.41	10.13		0.29

Note that, the value given in column $L^{\text{best}}*$ for TZK corresponds to the best solution ever found with the final parameter setting. For all other methods, L^{best} refers to the best solution out of ten runs and Δ^{best} to the gap to BKS. The gap of the average solution found to the BKS is given by Δ^{avg} and t^{avg} denotes to the average computing time in minutes. In addition, we provide the best solutions found during our overall testing in column $\overline{\textbf{AVNS}}$. Numbers in bold indicate the best solution found of Tarantilis et al. (2008) are the best solutions ever found with the final parameter setting, which we indicated with an asterisk. Finally, we report for our AVNS the best solutions found during the overall testing in column $\overline{\text{AVNS}}$.

Concerning the entire instance set of Crevier et al. (2007), a comparison of AVNS is only possible with CCL, VNS/TS and HDHR because TZK only provide solutions for the first subset of the instances. AVNS is able to improve on the solution quality of CCL and VNS/TS concerning the best as well as the average quality. Moreover, the runtimes of AVNS are clearly faster than those of VNS/TS. AVNS is not able to match the solution quality of HDHR, which is superior to all comparison methods in terms of solution quality.

In Table 9, the results of AVNS on the test instances of Tarantilis et al. (2008) are compared to those of TZK, the VNS/TS of Schneider et al. (2014) and HDHR. The reported measures are the same as in Tables 7 and 8. Concerning the average gap to the BKS, the AVNS is able to improve on the results of TZK and VNS/TS based on average as well as best solution quality. AVNS is not able to match the solution quality of HDHR, which again outperforms all other methods concerning solution quality. AVNS is able to provide two new BKS during the ten test runs and four new BKS during the overall testing. Runtimes are observably faster than those of VNS/TS.

5.3 Experiments on EVRPRF instances

In this section, we conduct numerical studies on EVRPRF instances. Section 5.3.1 describes the generation of the EVRPRF instances in more detail. Section 5.3.2 presents the computational results of our AVNS on the new instances.

5.3.1 Generation of EVRPRF instances

Our EVRPRF instances are based on the benchmark instances for the CVRP introduced by Christofides and Eilon (1969) and Golden et al. (1998). To generate valid EVRPRF instances, the following adjustments are made: The service time t_i^s of each customer $i \in C$ is set to ten time units. The battery capacity p of each vehicle is equal to the amount of electricity required to travel 60 % of the average route length of a high-quality solution of the respective CVRP instance. The CVRP solutions are taken from the website http://neumann.hec.ca/chairedistributique/data/vrp/old/ for the instances of Christofides and Eilon (1969) and from the paper of Mester and Bräysy (2007) for the instances of Golden et al. (1998). The fixed costs per vehicle c^{fix} are calculated by dividing the objective function value of the respective high-quality CVRP solution by the number of vehicles employed in this solution (rounded up to the next multiple of 20). The recharging speed g is set such that recharging the amount p takes 30 time units. Due to the additional time-consumption of visiting recharging facilities, the maximum route durations given by Christofides and Eilon (1969) and Golden et al. (1998) are increased by t_i^s multiplied with the average number of customers per route in the corresponding high-quality CVRP solution.

Further, each problem instance is complemented with eleven recharging facilities, of which one is located at the depot. The docking time t_i^d of each recharging facility

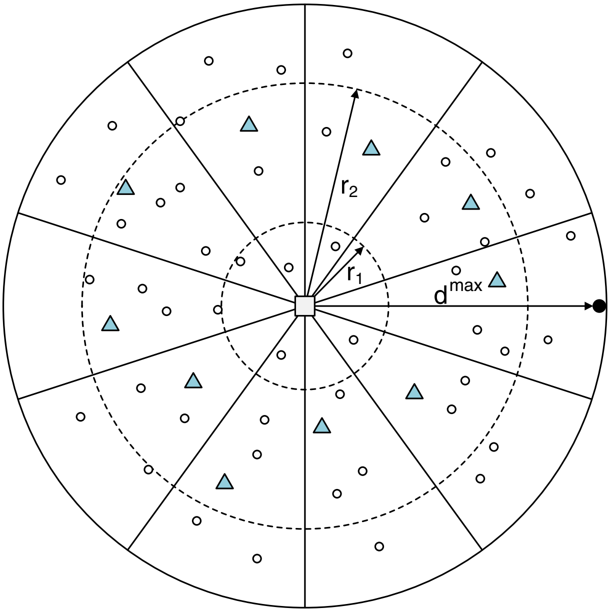

Fig. 4 Locating recharging facilities

$i \in F_F$ is set to five time units. The procedure for locating the recharging facilities is illustrated in Fig. 4. First, we generate a circle around the depot with a radius that corresponds to the maximal distance d^{max} of any customer to the depot. From this circle, we create a circular ring using the radii $r_1 = 0.3 \cdot d^{max}$ and $r_2 = 0.8 \cdot d^{max}$ and divide this ring into ten sectors of identical size. Within each of this circle ring sectors, we iteratively draw possible locations for the recharging facility in a random fashion until the following two criteria are met: (1) the location does not coincide with a customer location and (2) the distance of the possible location to all previously placed recharging facilities exceeds a given threshold. This threshold is continuously decreased after a certain number of the generated random points have not met this criterion.

In this way, we generate a total of 34 large EVRPRF instances, 14 based on the instances of Christofides and Eilon (1969) and 20 based on those of Golden et al. (1998). In addition, we create a set of small problem instances as follows: for each of the large EVRPRF instances of Christofides and Eilon (1969), we generate four small instances by (1) drawing 5, 10, 15 and 20 customers of the original instances and removing the remaining ones and (2) solving the thus generated instances with our AVNS and removing the recharging facilities that are not used in the produced solutions. In this way, 56 small instances are generated, which are denoted by the identifier of the underlying CVRP instance (CE plus instance number) followed by the number of customers (#C) and the number of facilities (#F) in the instance. For example, CE-01-05C2F denotes the instance obtained from the CVRP instance CE-01, containing five customers and two facilities.

Table 10 Comparison of the AVNS results on the generated small-sized EVRPRF instances with CPLEX

Inst.	CPLEX		AVNS				
	L	t(s)	L^{best}	$\Delta^{\text{best}}(\%)$	L^{avg}	$\Delta^{\text{avg}}(\%)$	t^{avg}(s)
CE-01-05C2F	292.55	914	292.55	0.00	292.55	0.00	0.23
CE-02-05C1F	195.78	24	195.78	0.00	195.78	0.00	0.17
CE-03-05C1F	225.49	34	225.49	0.00	225.49	0.00	0.16
CE-04-05C2F	207.09	3,988	207.09	0.00	207.09	0.00	0.23
CE-05-05C2F	261.72	68	261.72	0.00	261.72	0.00	0.39
CE-06-05C2F	382.72	51	382.72	0.00	382.72	0.00	0.36
CE-07-05C2F	508.96	91	508.96	0.00	508.96	0.00	0.26
CE-08-05C2F	227.70	75	227.70	0.00	227.70	0.00	0.19
CE-09-05C3F	389.60	217	389.60	0.00	389.60	0.00	0.33
CE-10-05C2F	322.42	17	322.42	0.00	322.42	0.00	0.40
CE-11-05C3F	435.71	72	435.71	0.00	435.71	0.00	0.33
CE-12-05C4F	286.50	7,200	286.50	0.00	286.50	0.00	0.37
CE-13-05C4F	494.33	7,200	494.33	0.00	494.33	0.00	0.47
CE-14-05C5F	322.63	7,200	322.63	0.00	322.63	0.00	0.61
CE-01-10C2F	439.92	7,200	439.92	0.00	439.92	0.00	0.99
CE-02-10C4F	433.39	7,200	433.39	0.00	433.39	0.00	0.67
CE-03-10C4F	312.87	7,200	312.87	0.00	312.87	0.00	0.76
CE-04-10C5F	338.90	7,200	338.90	0.00	338.90	0.00	1.98
CE-05-10C6F	397.11	1,343	397.11	0.00	397.11	0.00	2.21
CE-06-10C4F	544.12	7,200	544.12	0.00	544.12	0.00	0.38
CE-07-10C3F	533.65	7,200	533.65	0.00	533.65	0.00	0.35
CE-08-10C4F	429.21	7,200	429.21	0.00	430.70	0.35	0.48
CE-09-10C5F	736.75	7,200	736.75	0.00	736.75	0.00	0.47
CE-10-10C5F	645.73	7,200	645.73	0.00	645.73	0.00	0.41
CE-11-10C3F	428.14	7,200	428.14	0.00	428.14	0.00	1.03
CE-12-10C6F	499.53	7,200	499.53	0.00	499.66	0.03	1.00
CE-13-10C4F	447.75	7,200	447.75	0.00	447.75	0.00	1.38
CE-14-10C7F	417.64	7,200	417.64	0.00	417.64	0.00	2.68
CE-01-15C4F	491.70	7,200	491.70	0.00	491.70	0.00	1.00
CE-02-15C5F	586.13	7,200	586.13	0.00	587.43	0.22	0.69
CE-03-15C3F	498.00	7,200	498.00	0.00	498.00	0.00	1.22
CE-04-15C5F	488.38	7,200	488.38	0.00	488.38	0.00	1.87
CE-05-15C6F	489.54	7,200	489.54	0.00	489.54	0.00	1.40
CE-06-15C5F	611.58	7,200	611.58	0.00	611.58	0.00	0.64
CE-07-15C5F	876.38	7,200	876.38	0.00	876.38	0.00	0.46
CE-08-15C4F	605.26	7,200	605.26	0.00	607.97	0.45	0.86
CE-09-15C5F	754.09	7,200	754.09	0.00	754.09	0.00	0.54
CE-10-15C5F	515.30	7,200	515.30	0.00	515.30	0.00	0.58
CE-11-15C3F	483.14	7,200	483.14	0.00	483.14	0.00	2.10

Table 10 continued

Inst.	CPLEX		AVNS				
	L	t(s)	L^{best}	Δ^{best}(%)	L^{avg}	Δ^{avg}(%)	t^{avg}(s)
CE-12-15C7F	595.82	7,200	595.82	0.00	595.82	0.00	1.94
CE-13-15C4F	459.06	7,200	459.06	0.00	459.06	0.00	3.19
CE-14-15C6F	428.83	7,200	428.83	0.00	428.83	0.00	1.80
CE-01-20C6F	718.64	7,200	718.64	0.00	718.64	0.00	1.05
CE-02-20C6F	637.97	7,200	637.41	−0.09	637.41	−0.09	1.08
CE-03-20C4F	526.62	7,200	526.62	0.00	526.62	0.00	2.26
CE-04-20C6F	509.02	7,200	509.02	0.00	509.02	0.00	2.74
CE-05-20C5F	526.41	7,200	526.41	0.00	526.41	0.00	2.19
CE-06-20C5F	602.24	7,200	602.24	0.00	602.24	0.00	1.00
CE-07-20C6F	909.96	7,200	895.52	−1.59	915.45	0.60	0.53
CE-08-20C4F	806.08	7,200	802.46	−0.45	803.19	−0.36	0.71
CE-09-20C5F	919.69	7,200	773.20	−15.93	773.20	−15.93	0.57
CE-10-20C7F	893.96	7,200	889.74	−0.47	889.74	−0.47	0.96
CE-11-20C3F	662.05	7,200	662.05	0.00	662.05	0.00	2.96
CE-12-20C7F	622.46	7,200	622.46	0.00	622.46	0.00	2.18
CE-13-20C4F	697.26	7,200	697.26	0.00	697.26	0.00	2.57
CE-14-20C8F	614.58	7,200	600.98	−2.21	602.98	−1.89	2.53
Avg.		5,780.25		−0.37		−0.31	1.09

For CPLEX, L denotes the objective function value and t the total runtime in seconds. The maximum duration for CPLEX was set to 7,200 s. For AVNS, L^{best} denotes the best solution found in ten runs, L^{avg} the average solution of ten runs, Δ^{best} and Δ^{avg} the gaps to the CPLEX solution, and t^{avg} the average computing time in seconds

5.3.2 Results on the EVRPRF instances

We solve the small EVRPRF instances by means of our AVNS and compare our results to those of the commercial solver CPLEX. Ten AVNS runs are conducted for each problem instance. CPLEX is given a time limit of 7,200 s for each instance and we generate three dummy vertices for each recharging facility to represent visits to the facility. The results are presented in Table 10. For CPLEX, we report the solution L and the runtime t in seconds. If CPLEX terminates before the end of the time limit, the given solution is optimal. Otherwise, the result corresponds to the best upper bound found within the time limit. For the AVNS, we give the best solution found in the ten runs (L^{best}), the relative gap of this solution to the CPLEX solution (Δ^{best}), the average solution (L^{avg}), the gap of the average solution to CPLEX (Δ^{avg}) and the average runtime in seconds (t^{avg}).

While CPLEX is only able to solve twelve out of 56 instances to optimality, AVNS is able to provide high-quality solutions with an average runtime of just above one second. Concerning the best solution, the quality of all optimal CPLEX solutions and all CPLEX upper bounds is matched or improved. The robustness of our AVNS is again proven by a negative average gap of the average AVNS solution on the CPLEX solution.

Finally, we provide the results of our AVNS on the large EVRPRF instances in Table 11. The instances are denoted by the identifier of the underlying CVRP instance (CE or G, respectively, plus instance number) followed by the number of customers (#C) in the instance. The same measures as for the AVNS results on the small instances are reported. Here, no results for comparison are available, however, we want to provide researchers tackling the EVRPRF in the future with the possibility to compare their results with those of our AVNS. Moreover, we can show that our AVNS is able to provide solutions to the large instances within reasonable runtimes.

6 Conclusion

This paper presents an adaptive variable neighborhood search (AVNS) to address the vehicle routing problem with intermediate stops (VRPIS), in which vehicles are

Table 11 Results of AVNS on large EVRPRF instances

Inst.	AVNS				AVNS		
	L^{best}	L^{avg}	t^{avg}(min)	Inst.	L^{best}	L^{avg}	t^{avg}(min)
CE-01-050C	1,148.73	1,148.73	3.31	G-01-240C	12,363.17	12,476.44	23.81
CE-02-075C	1,895.52	1,903.06	3.60	G-02-320C	18,253.91	18,390.20	40.34
CE-03-100C	1,822.43	1,825.80	11.81	G-03-400C	22,504.96	24,069.90	41.68
CE-04-150C	2,344.28	2,353.74	14.19	G-04-480C	29,334.38	29,787.80	41.67
CE-05-199C	3,164.90	3,180.26	19.01	G-05-200C	12,995.39	13,135.66	39.31
CE-06-050C	1,053.82	1,056.98	1.70	G-06-280C	17,135.69	17,206.53	41.67
CE-07-075C	2,040.79	2,046.01	1.47	G-07-360C	20,650.82	21,833.96	41.68
CE-08-100C	1,703.03	1,709.49	4.74	G-08-440C	25,211.37	25,861.50	41.69
CE-09-150C	2,354.09	2,375.32	5.67	G-09-255C	1,564.96	1,573.69	41.62
CE-10-199C	2,856.10	2,885.45	8.20	G-10-323C	1,863.69	1,880.54	39.63
CE-11-120C	2,250.00	2,273.21	13.86	G-11-399C	2,232.24	2,259.98	41.74
CE-12-100C	1,960.85	1,976.11	3.98	G-12-483C	2,540.14	2,583.65	41.80
CE-13-120C	2,232.42	2,318.04	9.53	G-13-252C	2,000.50	2,034.28	15.00
CE-14-100C	1,808.34	1,834.67	6.06	G-14-320C	2,455.81	2,477.32	19.95
				G-15-396C	3,591.95	3,633.86	28.64
				G-16-480C	4,264.65	4,294.46	35.73
				G-17-240C	1,667.67	1,682.53	14.77
				G-18-300C	2,251.71	2,273.29	18.27
				G-19-360C	3,655.12	3,690.33	34.29
				G-20-420C	4,603.23	4,633.80	34.82
Avg.			7.65				33.90

L^{best} corresponds to the best solution found in ten runs and L^{avg} denotes the average solution of ten runs. The average computing time in minutes is given by t^{avg}

required to stop at certain facilities along their route to remain operational. The competitiveness of the proposed approach is demonstrated on benchmark instances from the literature designed for the green VRP and the VRP with intermediate replenishment facilities, which both represent special cases of VRPIS. Our AVNS algorithm shows a convincing performance compared to the methods from the literature and is able to obtain numerous new best solutions.

As a special case of the VRPIS, we additionally consider the electric vehicle routing problem with recharging facilities (EVRPRF). We design two sets of small and large

Table 12 Comparison of the results of our final AVNS implementation to those obtained by our AVNS without the problem-specific methods explicitly addressing intermediate stops (AVNS-without) on the VRPIRF instances of Crevier et al. (2007)

Inst.	BKS	AVNS			AVNS–without		
		L^{avg}	$\Delta^{avg}(\%)$	$t^{avg}(\text{min})$	L^{avg}	$\Delta^{avg}(\%)$	$t^{avg}(\text{min})$
a1	**1,179.79**	1,184.57	0.41	0.64	1,186.85	0.60	0.78
b1	**1,217.07**	1,218.21	0.09	4.19	1,219.47	0.20	8.99
c1	**1,866.76**	1,925.41	3.14	32.98	1,923.19	3.02	37.85
d1	**1,059.43**	1,061.50	0.20	0.55	1,062.24	0.27	1.08
e1	**1,309.12**	1,312.75	0.28	5.08	1,312.46	0.25	5.95
f1	**1,570.41**	1,601.40	1.97	34.99	1,611.08	2.59	39.62
g1	**1,181.13**	1,183.75	0.22	1.69	1,187.66	0.55	2.67
h1	**1,545.50**	1,567.22	1.41	14.08	1,571.66	1.69	25.43
i1	**1,922.18**	1,974.97	2.75	35.11	1,977.47	2.88	42.05
j1	**1,115.78**	1,116.82	0.09	2.02	1,119.00	0.29	2.73
k1	**1,576.36**	1,600.42	1.53	10.74	1,598.56	1.41	27.30
l1	**1,863.28**	1,916.07	2.83	40.59	1,921.14	3.11	41.95
Avg.			1.24	15.22		1.40	19.70
a2	**997.94**	**997.94**	0.00	0.72	998.39	0.05	1.06
b2	**1,291.19**	1,300.42	0.72	4.83	1,297.52	0.49	10.18
c2	**1,715.60**	1,741.55	1.51	18.32	1,747.94	1.89	29.12
d2	**1,856.84**	1,903.15	2.49	30.64	1,916.31	3.20	38.91
e2	**1,919.38**	1,957.80	2.00	41.60	1,988.88	3.62	41.33
f2	**2,230.32**	2,313.08	3.71	42.80	2,330.63	4.50	42.07
g2	**1,152.92**	1,158.21	0.46	2.20	1,158.57	0.49	3.63
h2	**1,575.28**	1,586.24	0.70	21.20	1,608.81	2.13	26.98
i2	**1,919.74**	1,971.27	2.68	41.10	1,982.83	3.29	40.28
j2	**2,247.70**	2,303.67	2.49	41.93	2,344.55	4.31	41.77
Avg.			1.68	24.53		2.40	27.53
Tot. Avg.			1.44	19.46		1.86	23.26

L^{avg} denotes the average solution quality of ten runs. The gap of the average solution found to the BKS is given by Δ^{avg} and the average computing time in minutes by t^{avg}. Numbers in bold indicate the best solution found

EVRPRF instances based on well-known CVRP benchmarks. On the small instances, our AVNS, using a runtime of approximately one second, is able to match or improve all results obtained by the commercial solver CPLEX within a time limit of 2 h.

Appendix: Influence of problem-specific components

Table 12 shows a comparison of our AVNS to an AVNS without the problem-specific components addressing intermediate stops (denoted as AVNS-without) on the VRPIRF instances of Crevier et al. (2007). For each instance, we report the instance name and the BKS. Moreover, we provide the average solution found in the ten runs (L^{avg}), the gap of the average solution to the BKS (Δ^{avg}) and the average runtime in minutes (t^{avg}) for both methods. Finally, averages of the runtimes and the gaps to the BKS over the complete set of instances are given at the end of the table. Results show that adding the problem-specific components clearly improves the solution quality while notably reducing runtimes.

References

Amaya A, Langevin A, Trépanier M (2007) The capacitated arc routing problem with refill points. Oper Res Lett 35(1):45–53

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

Archetti C, Speranza MG, Vigo D (2013) Vehicle routing problems with profits. Tech. Rep. WPDEM 2013/3, Department of Economics and Management, University of Brescia

Bard J, Huang L, Dror M, Jaillet P (1998) A branch and cut algorithm for the VRP with satellite facilities. IIE Trans 30(9):821–834

Barnes J, Wiley V, Moore J, Ryer D (2004) Solving the aerial fleet refueling problem using group theoretic tabu search. Math Comput Model 39(6–8):617–640

Beliën J, Boeck LD, Ackere JV (2014) Municipal solid waste collection and management problems: A literature review. Trans Sci 48(1):78–102

Benjamin A, Beasley J (2010) Metaheuristics for the waste collection vehicle routing problem with time windows, driver rest period and multiple disposal facilities. Comput Oper Res 37(12):2270–2280

Christofides N, Eilon S (1969) An algorithm for the vehicle-dispatching problem. Oper Res Quart 20(3):309–318

Clarke G, Wright JW (1964) Scheduling of vehicles from a central depot to a number of delivery points. Oper Res 12(4):568–581

Coene S, Arnout A, Spieksma FCR (2010) On a periodic vehicle routing problem. J Oper Res Soc 61:1719–1728

Conrad RG, Figliozzi MA (2011) The recharging vehicle routing problem. In: Doolen T, Aken EV (eds) Proceedings Industrial Engineering Research Conference

Cordeau JF, Gendreau M, Laporte G (1997) A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 30(2):105–119

Crainic TG, Ricciardi N, Storchi G (2009) Models for evaluating and planning city logistics systems. Transp Sci 43(4):432–454

Crevier B, Cordeau JF, Laporte G (2007) The multi-depot vehicle routing problem with inter-depot routes. Eur J Oper Res 176:756–773

Erdoğan S, Miller-Hooks E (2012) A green vehicle routing problem. Transp Res Part E Logist Transp Rev 48(1):100–114

Golden BL, Wasil EA, Kelly JP, Chao IM (1998) The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets and computational results. In: Crainic T, Laporte G (eds) Fleet management and logistics. Kluwer, Boston, pp 33–56

Hansen P, Mladenović N (2001) Variable neighborhood search: principles and applications. Eur J Oper Res 130(3):449–467

Hemmelmayr V, Doerner K, Hartl R, Rath S (2013) A heuristic solution method for node routing based solid waste collection problems. J Heuristics 19(2):129–156

Hemmelmayr VC, Doerner KF, Hartl RF (2009) A variable neighborhood search heuristic for periodic routing problems. Eur J Oper Res 195(3):791–802

Kim BI, Kim S, Sahoo S (2006) Waste collection vehicle routing problem with time windows. Comput Oper Res 33:3624–3642

Lin S (1965) Computer solutions to the traveling-salesman problem. Bell Syst Techn J 44(10):2245–2269

Mester D, Bräysy O (2007) Active-guided evolution strategies for large-scale capacitated vehicle routing problems. Comput Oper Res 34(10):2964–2975

Or I (1976) Traveling salesman-type problems and their relation to the logistics of regional blood banking. Ph.D. thesis, Department of Industrial Engineering and Management Sciences, Northwestern University Evanston, USA

Perrier N, Langevin A, Campbell JF (2007) A survey of models and algorithms for winter road maintenance. Part IV: Vehicle routing and fleet sizing for plowing and snow disposal. Comput Oper Res 34(1):258–294

Pisinger D, Ropke S (2007) A general heuristic for vehicle routing problems. Comput Oper Res 34(8):2403–2435

Prescott-Gagnon E, Desaulniers G, Rousseau LM (2012) Heuristics for an oil delivery vehicle routing problem, forthcoming in Flexible Services and Manufacturing Journal

Raviv T, Kaspi M (2012) The locomotive fleet fueling problem. Oper Res Lett 40(1):39–45

Rochat Y, Taillard E (1995) Probabilistic diversification and intensification in local search for vehicle routing. J Heuristics 1(1):147–167

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

Salazar-Aguilar MA, Langevin A, Laporte G (2013) The synchronized arc and node routing problem: application to road marking. Comput Oper Res 40(7):1708–1715

Savelsbergh M (1992) The vehicle routing problem with time windows: minimizing route duration. ORSA J Comput 4(2):146–154

Schneider M, Stenger A, Goeke D (2014) The electric vehicle routing problem with time windows and recharging stations. Transp Sci Artic Adv. doi:10.1287/trsc.2013.0490

Solomon MM (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper Res 35(2):254–265

Stenger A, Schneider M, Goeke D (2013a) The prize-collecting vehicle routing problem with single and multiple depots and non-linear cost. EURO J Transp Logis 2(1–2):57–87

Stenger A, Vigo D, Enz S, Schwind M (2013b) An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. Transp Sci 47(1):64–80

Tarantilis CD, Zachariadis EE, Kiranoudis CT (2008) A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. INFORMS J Comput 20(1):154–168

Thompson PM, Orlin JB (1989) Theory of cyclic transfers. Working Paper, MIT, Operations Research Center, Cambridge, USA

Vidal T, Crainic TG, Gendreau M, Lahrichi N, Rei W (2012) A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Oper Res 60(3):611–624

Wang YW, Lin CC (2013) Locating multiple types of recharging stations for battery-powered electric vehicle transport. Transp Res Part E Logis Transp Rev 58:76–87