EURO J Transp Logist (2013) 2:57–87
DOI 10.1007/s13676-013-0022-4

RESEARCH PAPER

The prize-collecting vehicle routing problem with single and multiple depots and non-linear cost

Andreas Stenger · Michael Schneider · Dominik Goeke

Received: 1 March 2012 / Accepted: 29 January 2013 / Published online: 1 March 2013

Abstract In this paper, we propose a new routing problem to model a highly relevant planning task in small package shipping. We consider the Prize-Collecting Vehicle Routing Problem with Non-Linear cost in its single and multi-depot version, which integrates the option of outsourcing customers to subcontractors instead of serving them with the private fleet. Thereby, a lower bound on the total customer demand to be served by the private fleet guarantees a high utilization of the fleet capacity. To represent the practical situation, where a discount is given by a subcontractor if larger amounts of packages are outsourced, subcontracting costs follow a non-linear function. The considered problem is NP-hard and we propose an Adaptive Variable Neighborhood Search algorithm to solve instances of realistic size. We propose new benchmark sets for the single and the multi-depot problem, which are adapted from test instances of the capacitated VRP and the closely related Multi-Depot VRP with Private fleet and Common carrier. In numerical studies, we investigate the performance of our algorithm on the newly generated test instances and on standard benchmark problems of related problems. Moreover, we study the effect of different cost functions and different values of the minimal demand to be served by the private fleet on the routing solutions obtained.

Keywords Vehicle routing · Prize-collecting · Heuristic · Non-linear cost

A. Stenger (✉)
Chair of IT-based Logistics, Goethe University, Grueneburgplatz 1, 60323 Frankfurt, Germany
e-mail: stenger@wiwi.uni-frankfurt.de

M. Schneider · D. Goeke
Chair of Business Information Systems and Operations Research,
TU Kaiserslautern, Erwin-Schrödinger-Straße, 67653 Kaiserslautern, Germany
e-mail: schneider@wiwi.uni-kl.de

D. Goeke
e-mail: dominik.goeke@wiwi.uni-kl.de

Introduction

The market situation for small package shippers (SPS) has drastically changed since the deregulation in the EU and the US. Before, the formerly big players like DHL[1] used to operate huge vehicle fleets and perform all last-mile deliveries with their own employees. However, rising competition has forced them to adopt the business model of using subcontractors for last-mile delivery as done by companies like DPD[2]. The subcontractors are paid per parcel delivered, saving the SPS the high fixed costs for vehicles and employees. Beside the outsourcing of entire unprofitable delivery areas, subcontractors are often used on the operational level to balance high demand fluctuations, in particular when the capacity of the self-operated vehicle fleet is not sufficient to serve all customers on a given day. On such days, the operational task is to decide which customers should be served by the private fleet and which customers should be subcontracted. The decision has to balance the trade-off between the cost of serving a customer [based on the solution of a Vehicle Routing Problem (VRP)] and the costs for subcontracting the customer. Chu (2005) modeled this planning problem, relaxing several practical constraints, as an extension of the Capacitated VRP (CVRP), which was later named VRP with Private fleet and Common carriers (VRPPC) (Bolduc et al. 2008). Extending the VRPPC, Stenger et al. (2012) proposed the Multi-Depot VRPPC (MDVRPPC) that considers not only multiple self-owned depots but also different subcontractors with individual subcontracting costs and limited delivery radiuses.

However, both problems disregard important real-world characteristics. First, a lower bound on the customer demand served by the private fleet is generally mandatory in order to maintain the profitability of the vehicle fleet. Second, the costs charged by a subcontractor for serving additional customers follow a non-linear cost function. Quantity discounts are given if the number of customers to outsource falls into a certain discount range. In this way, the subcontractor tries to improve the capacity utilization of his fleet.

We contribute by incorporating the above-described real-world characteristics in a planning problem called Prize-Collecting Multi-Depot Vehicle Routing Problem with Non-Linear costs (PCMDVRPNL). It extends the MDVRPPC and the well-known Prize-Collecting Traveling Salesman Problem (PCTSP), in which a prize is collected when visiting a customer and penalties are incurred for each unvisited customer. The task is to collect at least a given prize while minimizing the sum of distances traveled and penalty costs for unvisited customers. In the PCMDVRPNL, penalty costs are equal to subcontracting costs while at least a given customer demand (prize) has to be served by the private fleet. Besides the problem with multiple self-owned depots and multiple subcontractors, we also consider the single-depot case of the problem, denoted as Prize-Collecting Vehicle Routing Problem with Non-Linear cost (PCVRPNL). This problem is modelled as a special case of the PCMDVRPNL, in which only one self-owned depot and one subcontractor with unlimited delivery radius are available.

[1] http://www.dhl.com.

[2] http://www.dpd.com.

As we are, to the best of our knowledge, the first dealing with these problems in their given form, we propose new benchmark sets for both problems. The benchmark for single-depot PCVRPNL is adapted from test instances of the classical VRP and the PCMDVRPNL instances are based on test problems of the closely related MDVRPPC. As solution method, we present an Adaptive Variable Neighborhood Search (AVNS) algorithm that is inspired by the AVNS of Stenger et al. (2012) but employs modified neighborhood structures, a random mechanism for ordering these neighborhoods and route and customer selection rules specifically adapted to the non-linear problem addressed. The performance of the method is evaluated on the newly generated test instances and on available benchmark problems of the related VRPPC and MDVRPPC. Moreover, we study the effect of different cost functions on the route design and the subcontracting decisions as well as the influence of the minimum amount of demand to be served by the private fleet.

The remainder of the paper is structured as follows. First, we review the literature related to our work (see "Literature review"). Subsequently, we formulate the mixed integer program of the problem at hand (see "Mathematical model of the PCMDVRPNL"). In section "Solution method for the PCMDVRPNL", we present the problem-specific AVNS. The mathematical model and the details of our solution approach are described for PCMDVRPNL, of which the single depot problem can be considered a special case as described above. In section "Computational studies", we discuss the extensive numerical tests performed with our AVNS algorithm, followed by some concluding remarks in section "Conclusion".

Literature review

In this section, we provide a brief review of the literature that is of importance to our work. The idea of prize-collecting first arose in the context of the iron and steel industry, where a PCTSP was used to model the operational scheduling of a steel rolling mill. Balas (1989) transferred this idea to the general case of a traveling salesman and studied structural properties. As mentioned above, a traveling salesman collects a prize for each city visited and has to pay a penalty for each city that remains unvisited. The objective is to minimize the total distance traveled and penalty costs incurred for unvisited cities while collecting at least a given amount of prize money.

Several solution methods for the PCTSP have been proposed. For example, Dell'Amico et al. (1998) proposed a heuristic using lagrangian relaxation to produce an initial solution and subsequently apply an extension and collaborate procedure in the improvement phase. Recently, Chaves and Lorena (2008) presented a hybrid metaheuristic for generating initial solutions using a combination of greedy randomized search procedure and VNS. Based on the generated solutions, clusters are formed and promising clusters are identified and further improved by means of a local search procedure. However, no generally used set of PCTSP benchmark problems exists and thus the quality of the proposed solution methods cannot be evaluated in a straightforward fashion. For an extended literature review, the reader is referred to Feillet et al. (2005), who provide an overview of the literature on traveling salesman problems with profits.

The PCTSP was extended to a Prize-Collecting Vehicle Routing Problem (PCVRP) by Tang and Wang (2006) in order to model the hot rolling production scheduling problem. Here, every customer represents an order to be scheduled that has a given length that corresponds to the demand of the customer. Each vehicle route describes a turn and the vehicle capacity corresponds to the maximum length of a turn. The objective is to find the schedule that minimizes the variable production and fixed set-up costs and maximizes the profits of scheduled orders.

Chu (2005) presented an extension of the CVRP, in which customers can either be served by the private fleet or be outsourced to a common carrier. The costs for deliveries by the private fleet depend on the traveled distance and fixed vehicle cost. The common carrier is paid a fixed price per outsourced customer. The objective is to minimize the total costs including fixed vehicle costs, variable travel costs of the private fleet and the costs of assigning deliveries to the common carrier. To solve the problem (later named VRPPC by Bolduc et al. (2008), Chu (2005) proposed a simple heuristic based on the well-known Clarke and Wright algorithm (Clarke and Wright 1964). Another heuristic that outperforms the approach of Chu (2005) was developed by Bolduc et al. (2007). They modeled the VRPPC as heterogeneous VRP and presented a randomized construction-improvement-perturbation heuristic. Furthermore, they generated two large sets of benchmark instances for the VRPPC with up to 480 customers, which are based on classical VRP instances. Recently, two Tabu Search (TS) heuristics have been developed for the VRPPC. Côté and Potvin (2009) presented a heuristic which is mainly based on the unified TS framework proposed by Cordeau et al. (1997). The solutions obtained by this heuristic were further improved by the TS of Potvin and Naud (2011) which enhances the earlier TS by the concept of ejection chains. Numerical studies show that ejection chains help to significantly improve the solution quality, in particular on instances with a heterogeneous vehicle fleet, but also strongly increase computing time.

As described above, this paper considers a multi-depot problem that extends the classical Multi-Depot Vehicle Routing Problem (MDVRP) first described by Wren and Holliday (1972). Compared to the single-depot VRP, only few heuristic solution methods were presented for the MDVRP. Chao et al. (1993) proposed a multi-phase extension of the record-to-record travel method of Dueck (1993) and were able to solve benchmark instances with up to 360 customers and 9 depots. Renaud et al. (1996) presented a tabu-search heuristic which first assigns customers to their closest depot and next determines an initial routing solution using the improved petal heuristic of Renaud et al. (1996). Subsequently, a fast improvement phase, an intensification and a diversification phase are applied, all based on a set of local search operators which exchange customers of up to three routes. Cordeau et al. (1997) proposed another tabu search heuristic for MDVRP that is also able to solve Periodic VRP (PVRP) and periodic TSP problems. To diversify the search, attributes that are added to solutions are analyzed and penalties are modified accordingly in order to efficiently guide the search. Pisinger and Ropke (2007) used an Adaptive Large Neighborhood Search (ALNS), which was designed for a wide range of routing problems, to solve the MDVRP. The solutions obtained by ALNS were of higher quality than those found by Cordeau et al. (1997) but required larger computing time. Recently, Vidal et al. (2012) presented a Hybrid Genetic Search

with Adaptive Diversity Control (HGSADC) to solve MDVRP, PVRP and periodic MDVRP. The computational studies performed show considerable improvements for all problem classes. Stenger et al. (2012) proposed the MDVRPPC and developed an AVNS algorithm that biases the random shaking step. The approach obtains high quality solutions in short computing time for MDVRPPC and closely related problems, such as the MDVRP and the single-depot VRPPC.

Mathematical model of the PCMDVRPNL

In this section, we provide the mathematical formulation of the PCMDVRPNL, which includes the PCVRPNL as a special case.

We model the problem as an extension of the MDVRPPC proposed in Stenger et al. (2012). To define PCMDVRPNL, let $G = (V, E)$ be an undirected, complete graph. The set of nodes $V = J \cup H$ is composed of the set J of customers and the set H of depots. Set H itself is partitioned into a subset I of self-owned depots and a subset L of subcontractor depots. The total capacity of a depot $i \in H$, i.e., the maximum total demand it can serve, is limited to w_i units. A customer can either be served by a vehicle k of the set of private vehicles K or by a subcontractor. At most $k_{\max}$ identical vehicles of the private fleet K with a restricted capacity Q are available at each self-owned depot, where $|K| \leq k_{\max} \cdot |I|$ holds. For each employed vehicle a fixed cost of F is charged as well as the variable costs that are identical to the travel times c_{ij} for traversing arc $(i, j) \in E$. With each customer $j \in J$ a service time of t_j units is associated. The maximum duration of a route is limited to $t_{\max}$. Furthermore, $r_{\max}$ denotes the maximum delivery radius of subcontractor depots, i.e., customers outside this range are not served by the respective subcontractor.

The cost of subcontracting customer $j \in J$ to depot $l \in L$ is p_{lj}. This cost is discounted with factor $(1 - e_l(g_l^{\text{sub}}))$, where $e_l(g_l^{\text{sub}})$ denotes the discount given by subcontractor l for subcontracted demand g_l^{sub}. Moreover, at least T demand units have to be delivered by the private fleet.

The model uses two sets of decision variables. If node i follows node j on tour k, the binary variable x_{ijk} takes value 1, and value 0 otherwise. In addition, if customer $j \in J$ is assigned to depot $i \in H$, binary variable z_{ij} is equal to 1 and 0 otherwise. With the above definitions, the mathematical model of PCMDVRPNL can be stated as follows:

$$\min \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} F x_{ijk} + \sum_{i \in (I \cup J)} \sum_{j \in (I \cup J)} \sum_{k \in K} c_{ij} x_{ijk} + \sum_{j \in J} \sum_{l \in L} (1 - e_l(g_l^{\text{sub}})) p_{lj} z_{lj} \quad (1)$$

$$\sum_{k \in K} \sum_{i \in (I \cup J)} x_{ijk} = 1 - \sum_{l \in L} z_{lj} \quad \forall j \in J \quad (2)$$

$$\sum_{j \in J} \sum_{i \in (I \cup J)} q_j x_{ijk} \leq Q \quad \forall k \in K \quad (3)$$

$$\sum_{j \in J} q_j z_{ij} \leq w_i \quad \forall i \in H \quad (4)$$

$$\sum_{j\in(I\cup J)} x_{ijk} - \sum_{j\in(I\cup J)} x_{jik} = 0 \quad \forall i \in (I \cup J), \quad k \in K \tag{5}$$

$$\sum_{i\in I}\sum_{j\in J} x_{ijk} \le 1 \quad \forall k \in K \tag{6}$$

$$\sum_{i\in J}\sum_{j\in I\cup J}\sum_{k\in K} x_{ijk} \cdot q_i \ge T \tag{7}$$

$$\sum_{i\in R}\sum_{j\in R} x_{ijk} \le |R| - 1 \quad \forall R \subseteq J, \quad k \in K \tag{8}$$

$$\sum_{u\in J} x_{iuk} + \sum_{u\in(I\cup J)\setminus\{j\}} x_{ujk} \le 1 + z_{ij} \quad \forall i \in I, \quad j \in J, \quad k \in K \tag{9}$$

$$\sum_{i\in H} z_{ij} = 1 \quad \forall j \in J \tag{10}$$

$$\sum_{k\in K}\sum_{j\in J} x_{ijk} \le k_{\max} \quad \forall i \in I \tag{11}$$

$$\sum_{i\in(I\cup J)}\sum_{j\in(I\cup J)} (c_{ij} + t_i)x_{ijk} \le t_{\max} \quad \forall k \in K \tag{12}$$

$$c_{lj} z_{lj} \le r_{\max} \quad \forall j \in J, \quad l \in L \tag{13}$$

$$\sum_{j\in J} z_{lj} q_j = g_l^{\text{sub}} \quad \forall l \in L \tag{14}$$

$$x_{ijk} \in \{0,1\} \quad \forall i \in (I \cup J), \quad j \in (I \cup J), \quad k \in K \tag{15}$$

$$z_{ij} \in \{0,1\} \quad \forall i \in H, \quad j \in V \tag{16}$$

with

$$e_l(g_l^{\text{sub}}) = \begin{cases} 0 & \text{if } g_l^{\text{sub}} \le \vartheta \cdot Q \\ \gamma \cdot \frac{e_{\max}}{\lfloor \sum_{j\in J} q_j - T/Q \rfloor} & \text{if } ((\gamma-1)+\vartheta)\cdot Q < g_l^{\text{sub}} \le (\gamma+\vartheta)\cdot Q \\ & \gamma = 1, \ldots, \left\lfloor \left(\sum_{j\in J} q_j - T \right) / Q \right\rfloor \\ e_{\max} & \text{otherwise} \end{cases} \tag{17}$$

The objective (1) is to minimize the sum of fixed costs of employed vehicles, travel and subcontracting costs. Constraints (2) ensure that each customer is either visited exactly once by a vehicle of the private fleet or it is subcontracted. Constraints (3) and (4) describe the capacity restrictions of vehicles and depots. Constraints (5) and (6) define that each node entered by a vehicle has to be left and that each route has to end at the depot from which it originated. Constraint (7) determines the minimum customer demand T to be served by the private fleet. Subtour elimination is ensured by Constraints (8). Constraints (9) guarantee that a customer is only assigned to a certain depot if there exist a route starting from this depot that visits the customer. Constraints (10) guarantee that each

customer is assigned to a depot. The number of employed vehicles is restricted by (11). Compliance with route duration constraints is ensured in (12). Due to Constraints (13), a customer can only be subcontracted to depot l if it is not farther from l than $r_{\max}$ units. Equation (14) defines the subcontracted demand for each subcontractor. Binary variables are defined in Constraints (15) and (16) (Fig. 1).

In (17), we define the nonlinear cost function, which is inspired by real-life considerations. In industry practice, a subcontractor is paid per package volume, i.e., the demand units assigned to it by the SPS. However, the price per demand unit is usually not fixed but depends on the total subcontracted demand. More precisely, subcontractors grant discounts if the demand transferred by the SPS is sufficient to fill an entire vehicle as this renders the subcontractor's delivery operations most efficient. To model this circumstance, the price charged by the subcontractor follows a step function, i.e., the discount factor e_l is increased from 0 every time the total demand assigned to the subcontractor g_l^{sub} exceeds a given percentage ϑ of the vehicle capacity Q (also assumed for the fleet of the subcontractor). The subcontractor discount is limited to $e_{\max}$. Further, e_l is linked to the minimum demand T to be delivered by the private fleet such that e_l reaches its maximum value at the latest when the subcontracted demand is equal to $\sum_{j\in J} q_j - T$. In detail, the number of discount steps are $\rho = \lfloor (\sum_{j\in J} q_j - T)/Q \rceil$, where $\lfloor \cdot \rceil$ denotes rounding to the nearest integer. Thus, if the subcontracted demand exceeds $((\gamma - 1) + \vartheta) \cdot Q$ demand units, with $\gamma = 1, \ldots, \rho$, factor e_l is increased by $\gamma \cdot e_{\max}/\rho$. Figure 3 depicts an example with three discount steps, $e_{\max} = 0.4$ and $\vartheta = 0.8$.

The PCVRPNL is defined as special case of PCMDVRPNL, with $|I| = 1$, $|L| = 1$, $r_{\max} = \infty$.

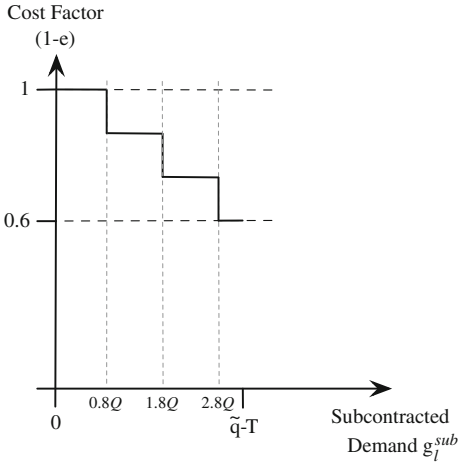

Fig. 1 Example of the stepwise subcontracting cost function with three discount steps. Symbol $\tilde{q}$ is used to denote the total demand of all customers, i.e., $\tilde{q} = \sum_{i\in J} q_i$

Solution method for the PCMDVRPNL

The PCMDVRPNL extends the already NP-hard CVRP by real-world characteristics of SPS. Consequently, only small problems can be solved by means of an exact approach. In order to handle problem instances of realistic size, we propose an Adaptive VNS (AVNS) metaheuristic approach to solve both the single and multi-depot version of the problem. AVNS, proposed by Stenger et al. (2012), is a metaheuristic that successfully combines ideas of Adaptive Large Neighborhood Search (ALNS) (Pisinger and Ropke 2007) and VNS. For a detailed description of the well-known metaheuristic concepts VNS and ALNS, we refer the reader to Mladenović and Hansen (1997) and Pisinger and Ropke (2007).

AVNS follows the solution process of a VNS and, starting from an initial solution, performs local search on systematically changing neighborhoods. In detail, a new neighboring solution is determined in each shaking step by exchanging customers among routes according to a set of predefined neighborhood structures. In contrast to standard VNS, the routes and customers involved in this shaking are not selected randomly in AVNS. Instead, a set of methods that bias the route and customer selection are applied to explore the solution space more efficiently. AVNS incorporates these selection methods in a fashion similar to the treatment of the removal and insertion heuristics in ALNS. In each iteration, one route and one customer selection method are chosen, while the probability of each method depends on its success in former iterations. In this way, the heuristic adapts to the characteristics and requirements of the problem instance to be solved and to the current state of the solution process. In the next step, AVNS performs a greedy local search on the generated neighboring solution and the resulting solution is finally accepted according to a simulated annealing criterion. If the solution is accepted, it replaces the current starting solution and a new shaking step using the first neighborhood structure is performed. Otherwise, the solution found is rejected and the shaking procedure is repeated with a more distant neighborhood and the former starting solution.

The major advantage of the AVNS framework is its high flexibility, which allows to design a problem-specific heuristic by simple modifications. Stenger et al. (2012) propose several route and customer selection methods that are specifically designed for solving problems including subcontractors, which are hence also applicable for the PCMDVRPNL. However, these methods are not sufficient to address the peculiarities induced by a non-linear cost-function. Therefore, to enhance the quality of our approach, we develop additional methods that consider these special requirements, like e.g., methods that aim at subcontracting several customers in one step in order to reach the next discount level. Furthermore, we do not use a fixed sequence of neighborhood structures, but generate a random ordering of the neighborhoods every time a new solution is accepted or after one iteration with all neighborhoods is complete (cp. Pirkwieser and Raidl 2008). Both modifications have proven their positive effect on the solution quality of the method in numerical tests. Finally, the local search is based on a subset of the operators that were employed in Stenger et al. (2012). In detail, tests on PCMDVRPNL instances revealed that we obtain similar solution quality without using the Or-Opt exchange,

Define the neighborhood structures $\mathcal{N}_\kappa$ with $\kappa = 1, .., \kappa_{max}$
Generate initial solution x
Set $\kappa := 1$
repeat
  if $\kappa = 1$ **then**
    $\pi \leftarrow$ generate random permutation of $\{1, .., \kappa_{max}\}$
  end if
  {*Adaptive Shaking*}
  Select route and customer selection method and generate $x' \in \mathcal{N}_{\pi(\kappa)(x)}$
  {*Local Search*}
  Find local optimum x'' with local search algorithm starting from initial solution x'
  {*Acceptance Decision*}
  if accept(x'') **then**
    $x \leftarrow x''$
    $\kappa \leftarrow 1$
  else
    $\kappa \leftarrow \kappa + 1 \mod \kappa_{max}$
  end if
  Update weights of route and customer selection methods
  $j = 1$
until given number of iterations without improvement reached

Fig. 2 Pseudocode of the AVNS-RN heuristic for solving PCMDVRPNL

while the computing time for the local search phase significantly decreases. A pseudocode overview of our AVNS with random neighborhood ordering (AVNS-RN) for solving PCMDVRPNL is given in Fig. 2.

In order to handle the outsourcing of customers to subcontracting depots, our heuristic uses the concept of *virtual vehicles* as proposed by Prosser and Shaw (1996) (see also Tang and Wang 2006; Bolduc et al. 2007). Here, a virtual vehicle with a capacity equal to the depot capacity w_i is assigned to each subcontractor depot $i \in L$ and all customers served by i are inserted into a corresponding virtual route. The length of the virtual subcontracting route is not considered in cost calculations as the subcontracting cost is based on outsourced quantities and not on the routing decision of the subcontractor.

In the following, we provide the details of our solution method: the "Initialization" phase , the "Adaptive shaking" phase including the route and customer selection methods, the "Local search" and the applied "Acceptance criterion".

Initialization

In the initialization phase, we decide for each customer whether it shall be outsourced or served by the private fleet and in the latter case to which depot the customer shall be assigned. Subsequently, we determine the vehicle routes for the private fleet. Our goal is to quickly compute a solution that serves as a starting point for the subsequent AVNS improvement. To decide whether a customer is outsourced or not, we use a modified version of the method proposed by Côté and Potvin (2009) for the closely related VRPPC, which is applied by Stenger et al. (2012) in similar fashion. We start by selecting customers to subcontract until only

the mandatory demand T remains to be served by the private fleet. To this end, we order the customers according to the quotient of subcontracting cost and demand and the most suitable customers are chosen for subcontracting as detailed in the following three-step procedure:

1. Determine for each customer $j \in J$ and each subcontractor depot $l \in L$ the quotient $\xi_{jl} = \frac{p_{jl}}{q_j}$ and insert ξ_{jl} into the list P sorted in increasing order.
2. Choose the first element in P denoted by $\xi_{j\_1l\_1}$ and check whether (1) the distance between j_1 and l_1 is smaller than the maximal service radius $r_{\max}$ of the subcontractor, (2) the remaining capacity of depot l_1 is greater than the demand q_{j_1} and (3) the demand of subcontracted customers including the investigated customer is smaller or equal to the maximum demand to be outsourced, i.e., $\sum_{i \in O \cup \{j1\}} q_i \leq \sum_{j \in J} q_j - T$, where T denotes the minimum customer demand to be served by the private fleet (see section "Mathematical model of the PCMDVRPNL") and O the list of already subcontracted customers. If all conditions are met, customer j_1 is subcontracted, added to the list O and all ξ_{j1l}, $l \in L$ are removed from P. Otherwise, only entry ξ_{j1l1} is removed from P.
3. Repeat step 2 until P is empty.

The customers remaining after the customers to be subcontracted have been selected are each assigned to the closest self-owned depot with free capacity. Based on this allocation, initial vehicle routes are created by means of the well-known Clarke and Wright algorithm (Clarke and Wright 1964). For all depots where the number of generated routes exceeds the number of available vehicles $k_{\max}$, we iteratively apply the following repair mechanism:

1. Select the route with the lowest demand at a randomly chosen depot with violations.
2. Remove all customers from this route and iteratively insert the customers at the cost optimal position concerning all routes. To be more precise, the insertion position is chosen such that the increase of total cost is minimized.
3. Repeat until the number of vehicles is lower or equal $k_{\max}$ at all depots.

Note that the obtained solution respects fleet restrictions but capacity constraints of the generated routes may be violated. In a final step, the solution obtained is improved by means the greedy local search described in section "Local search".

Adaptive shaking

In the adaptive shaking phase, we perturb the initial solution based on a given set of neighborhood structures, which are applied following a random ordering. To avoid the evaluation of unprofitable neighboring solutions, we bias the shaking by means of problem-specific selection methods that determine the routes and customers involved in the selected neighborhood moves. In each iteration, the route and customer selection method to be applied is chosen by a roulette wheel selection mechanism. In the process, the probability of each method adapts according to the

success of the selection method in former iterations, which is evaluated based on a scoring system.

In the following, we detail the utilized "Neighborhood structures", the "Route and customer selection methods" and the "Adaptive mechanism".

Neighborhood structures

The neighborhood structures used in our AVNS-RN algorithm are either defined by a move-exchange or a cyclic-exchange neighborhood operator (Thompson and Psaraftis 1993). For both operators, we define two sets of neighborhoods, one that only considers routes of the private fleet originating from identical depots and the other one allowing exchanges between all routes of the private fleet and the subcontractors. For all neighborhood operators and sets, the sequence length ω to be exchanged by neighborhood κ is randomly selected from interval $[0, \min(((\kappa - 1) \bmod 6) + 1, |N|)]$, where $|N|$ denotes the number of customers in the route. This means that for each neighborhood operator, we consider exchanges of up to six customers, while the number is obviously restricted by the number of customers in the route.

In detail, the following neighborhood structures are used:

- **Move customer sequence between vehicle routes of the private fleet originating from same depot** Neighborhood structures $\kappa = 1, \ldots, 6$ move a sequence of length ω customers from one route to another.
- **Move customer sequence between all routes of the private fleet and subcontractor** Structures $\kappa = 7, \ldots, 12$ are similar to the first set, however, customer sequences can be inserted into or removed from both routes that originate at a different depot and virtual routes of a subcontractor.
- **Cyclic exchange of customer sequence between routes of the private fleet originating from same depot** This set of neighborhood structures exchanges customer sequences between up to four different routes in a cyclic fashion. For example, a neighborhood considering three routes r_1, r_2, r_3 removes a customer sequence from r_1, inserts it into r_2, from where a sequence of customers is removed and transferred to route r_3. The sequence removed from r_3 is moved to r_1, thus closing the cycle. Neighborhood structures $\kappa = 13, \ldots, 18$ apply exchanges between two routes, $\kappa = 19, \ldots, 24$ between up to three and $\kappa = 25, \ldots, 30$ between up to four routes.
- **Cyclic exchange of customer sequence between all routes of the private fleet and subcontractor** This set of 18 neighborhood structures is similar to the third set but customer sequences can be inserted into or removed from both routes that originate at a different depot and virtual routes of a subcontractor.

Route and customer selection methods

Given a neighborhood structure κ to apply for the shaking step, our AVNS-RN does not select the routes and customers involved in the shaking step randomly as done in standard VNS. Instead, route selection and customer selection methods are applied

to bias the shaking step towards profitable regions of the search space. For example, this allows us to encourage shaking moves that force subcontracting in order to reach the next discount level at a certain subcontractor. The general procedure for performing an adaptive shaking move in our AVNS-RN algorithm is as follows (cp. Stenger et al. 2012):

1. Use roulette wheel selection to determine route selection method.
2. Determine the first route k_1 to be involved according to chosen route selection method.
3. Determine the remaining routes to be involved randomly, considering closeness measures.
4. Use roulette wheel selection to choose customer selection method.
5. For all involved routes, select randomly the number of customers ω to be exchanged and select the customers to be exchanged according to chosen selection method.
6. Perform the shaking move.

For selecting the first route involved in the shaking step, our AVNS-RN applies the following selection methods:

- Random: The probability of all routes to be selected is equal. This is the standard procedure in VNS.
- Longest route: The selection probability of a route is defined as being proportional to its total travel distance.
- Unit longest route: The probability of a route to be selected is proportional to the ratio between total travel distance and total demand of the customers in the route. Routes with long distance and low demand are generally considered as inefficient and are therefore more likely to be chosen for modification.
- Demand: The probability of a route is proportional to the average customer demand in this route. Biasing towards routes of the private fleet with high demand increases the chance of reaching the next discount step of a subcontractor.
- Distance to Discount: The probability of a subcontracted route to be chosen is inversely proportional to the gap between the total demand in this route and the demand required for reaching the next discount step.

The first three methods correspond to the route selection methods successfully applied in Stenger et al. (2012) and the last two methods constitute additional methods that consider the problem-specific characteristics of the PCMDVRPNL. Note that methods *longest route*, *unit longest route* and *demand* are only applicable for routes of the private vehicle fleet and method *distance to discount* can only be applied to routes of a subcontractor. As described above, the selection method is exclusively used to determine the first route involved in the shaking step, the other routes are chosen randomly. However, to avoid the evaluation of unpromising solutions, we restrict the potential routes using the concept of an embedding rectangle as proposed by Stenger et al. (2012). In this way, we consider merely exchanges of customers among routes that are close to each other.

In the next step, a customer selection method is chosen and subsequently applied to each of the routes involved in the shaking step in order to determine the customers to be exchanged. Our AVNS-RN applies the general selection methods described in Stenger et al. (2012) plus two problem-specific methods that aim at reaching higher discount levels. In case of routes of the private fleet, we exchange customer sequences of length ω (determined by selecting the first customer), while for subcontracted routes ω individual customers are selected by iteratively applying the selection method. We present the customer selection methods structured according to their applicability to the different types of routes.

Applicable to all routes:

- Random: The selection probability is equal for all customers.
- Distance to next route: The probability of a customer or a customer sequence being selected is inversely proportional to its distance to the center of gravity of the target route, i.e., the route in which the customers will be inserted. In this way, we bias towards customers that are close to the target route and thus more likely to be part of distance-efficient routes.

Only applicable to subcontracting routes:

- Subcontracting cost: The selection probability of a customer is proportional to its subcontracting cost. Customers with high subcontracting cost are more likely to be served efficiently by the private fleet.
- Unit subcontracting cost: The selection probability of a customer is proportional to the ratio between subcontracting cost and demand. This method targets at biasing customers with high subcontracting cost and/or low demand towards routes of the private fleet as they are likely to be efficiently served by them.
- Inverse demand: The selection probability of a customer is inversely proportional to its demand. Since customers with low demand can be inserted more easily into existing routes of the private fleet, we favor the removal of such customers from subcontracted routes.

Only applicable to routes of self-owned vehicles:

- Distance to neighboring customers: The probability of selecting a customer is inversely proportional to the sum of the distance between the first customer of the sequence and its predecessor and the last customer and its successor in the route. Customer sequences that are distant to other customers in a route are most likely to be exchanged in a profitable way.
- Demand: The probability of a customer sequence of being selected in a route of a self-owned vehicle is proportional to its demand. This favors the subcontracting of customers with high demand, which potentially leads to higher discount levels.

It is worth mentioning that a shaking move is capable of removing all customers of a route. In case such a move is accepted by the algorithm, the number of routes is reduced. In our computational studies, we found that this happens quite frequently, in particular due to the possibility of profitably moving customers into

subcontracted routes. Without further measures allowing the reopening of routes, the search often gets stuck in low quality regions of the search space. To counteract, we keep empty routes in the set of potential routes involved in the shaking. To be more precise, for every depot that has at least one vehicle left, we add an empty route. This allows the algorithm to open an additional route by inserting customers in an empty route during shaking.

Adaptive mechanism

Our AVNS-RN involves a quite extensive set of route and customer selection methods that clearly differ in their objective, i.e., they bias the shaking step in different directions. Depending on the problem instance addressed and the current state of the solution process, the success of individual methods will vary. Therefore, AVNS-RN follows the approach of ALNS and chooses the selection method to be used in each iteration based on associated probabilities that depend on the success of the respective method in former iterations. To be more precise, each selection method i is assigned a weight ζ_i that represents the success of the method. Given a set of s different methods, roulette wheel selection is applied in each shaking step to determine the method to be used, where the probability of selecting method i is equal to $\frac{\zeta_i}{\sum_{j=1}^{s} \zeta_j}$. Note that route selection and customer selection are treated separately in this process.

At the beginning of the search, all methods are assigned an equal weight ζ_0. During ϕ iterations, the success of a method is evaluated by a scoring mechanism. In case the shaking step using method i leads to a new overall best solution, the score of the method is increased by the amount δ_1. If the obtained solution improves on the incumbent, a somewhat lower score of δ_2 is assigned. Finally, if a solution is worse than the incumbent but nevertheless accepted by the algorithm, the score is increased by δ_3. After the evaluation period, the weights of all methods are updated as follows: Let ς_i denote the number of times method i was used during the evaluation period and let μ_i describe the obtained score of method i, then the new weight ζ_i is equal to $\zeta_i^{\text{old}}(1-\varrho) + \varrho\frac{\mu}{\varsigma_i}$. Factor $\varrho \in [0,1]$ is used to balance between the recent and the past success of a method when updating the weight (Pisinger and Ropke 2007).

Local search

After the shaking, greedy local search is used to determine a local optimum with respect to the applied neighborhood operators. Note that the local search is only performed on the routes involved in shaking. For intra-route moves, we use the well-known edge-exchange operator 2-opt, that exchanges two existing edges with two new ones (Lin 1965). For inter-route moves, a relocate and a swap operator are applied. Relocate removes a single customer from a route and inserts it into the other route at the cost-optimal position. The swap operator simply exchanges the position of two customers from different routes. The inter-route moves are

$x \leftarrow$ solution after shaking
$K_{shake} \leftarrow$ set of all routes involved in shaking
$K_{subc} \leftarrow$ set of subcontracted routes involved in shaking
$x_1 \leftarrow$ apply intra-route local search to all routes $K_{shake} \setminus K_{subc}$ in x using 2-opt
Initialize route pairs: $Pairs \leftarrow \{(k_a, k_b) \mid k_a, k_b \in K_{shake} \wedge k_a \neq k_b\}$
while $(|Pairs| > 0)$ **do**
  Randomly select a pair of routes $(k_a, k_b) \in Pairs$
  $x_2 \leftarrow$ solution obtained by best relocate move between k_a and k_b
  if (x_2 improves x_1) **then**
    $x_1 \leftarrow x_2$
    Reinitialize route pairs
    continue
  end if
  $x_3 \leftarrow$ solution obtained by best swap move between k_a and k_b
  if (x_3 improves x_1) **then**
    $x_1 \leftarrow x_3$
    Reinitialize route pairs
    continue
  end if
  $Pairs \leftarrow Pairs \setminus \{(k_a, k_b)\}$
end while
return x_1

Fig. 3 Pseudocode of the local search

applied to all routes of the private fleet and the virtual routes of the subcontractors. A more detailed description of the local search process is provided in pseudocode in Fig. 3.

During the local search and the shaking phase, we accept infeasible solutions in order to diversify the search. Solutions can be infeasible because they violate capacity limits of vehicles or the demand served by the private fleet falls below the mandatory lower limit. For both types of violations, a penalty term is added to the objective function:

$$f(s) = c(s) + \alpha \cdot overCap + \beta \cdot prizeDeficit,$$

where $c(s)$ is the original value of the objective function, α and β are penalty factors, which are positive weights in the interval $[Pen_{\min}, Pen_{\max}]$, *overCap* denotes the overcapacity and *prizeDeficit* the difference between the mandatory demand to be served by the private fleet and the actual value. Penalty factors are initialized with value Pen_{init}, and updated after each iteration without violation (with violation) of the respective constraint by dividing (multiplying) by 1.5. Note that we do not allow to violate the constraint on the maximum delivery radius of a subcontractor.

Acceptance criterion

Contrary to standard VNS approaches, where only improving solutions are accepted, we use a simulated annealing-based acceptance criterion, whose improvement potential has been shown in recent works (see, e.g., Pirkwieser and

Raidl 2008; see, e.g., Hemmelmayr et al. 2009; see, e.g., Stenger et al. 2012). The solution x'' obtained in the local search is compared to the currently best solution x and accepted if it improves the latter. Additionally, we accept deteriorating moves according to the Metropolis probability $e^{\frac{-(f(x'')-f(x))}{\theta}}$, where $f(\cdot)$ denotes the objective function value and θ the current temperature, which is used to control the degree of diversification. Starting from an initial value $\theta_{\text{init}} > 0$, the temperature is decreased by the factor η_{dec} after each AVNS iteration. In this way, the probability of accepting deteriorating solutions is reduced during the search ending in an intensification phase rejecting all non-improving solutions.

Computational studies

We perform extensive numerical tests to assess the performance of the proposed AVNS-RN algorithm and to study the influence of several problem parameters. We design new benchmark sets for PCMDVRPNL and its single-depot version as we are, to the best of our knowledge, the first dealing with these problems in their given form (see section "Generation of benchmark instances"). To find the best parameter values for our algorithm, we performed numerous tests on new benchmark instances (see section "Parameter setting"). Our first experiments assess the performance of our AVNS-RN on the generated benchmark instances and investigate the influence of the non-linear cost function on the routing solutions obtained (see section "Performance of AVNS-RN on PC(MD)VRPNL and the influence of non-linear subcontracting cost"). This is achieved by a comparison with the results realized with a linear cost function. In Section "Varying the minimum demand to be delivered by the private fleet", we study the impact of different values of the mandatory demand to be served by the private fleet T and show that this parameter significantly influences the subcontracting decisions and hence the overall solutions. Finally, to substantiate the competitive performance of our AVNS-RN, we present results obtained on benchmark instances proposed for the MDVRPPC by Stenger et al. (2012), which is closely related to the PCMDVRPNL (see section "Evaluating the algorithmic performance on benchmark instances of related problems") and on standard test instances of the VRPPC, which is closely related to the single-depot PCVRPNL.

All numerical tests were conducted on a desktop computer with an Intel i5 Processor clocked at 2.67 GHz and 4GB RAM. The algorithm is implemented in Java.

Generation of benchmark instances

To generate test problems for the PCMDVRPNL, we use the MDVRPPC instances proposed by Stenger et al. (2012) as a base. The subcontracting price p_{lj} of these instances depends mainly on the customer demand and can thus be used for our problem. The minimum demand T to be served by the private fleet is set to $0.7 \cdot \tilde{q}$. In addition, we give an upper bound which represents a high quality solution without

subcontracting, i.e., a high quality MDVRP solution (Cordeau et al. 1997). The upper bound provides a simple comparison value to evaluate the solution quality of our algorithm on the newly generated instances.

As PCVRPNL is an extension of the CVRP, we use the VRP benchmark instances proposed by Christofides and Eilon (1969) and Golden et al. (1998) as basis for designing a new benchmark set for our single depot problem. The benchmark design is inspired by the procedure for generating VRPPC instances described in Bolduc et al. (2008). However, utilization of the VRPPC instances presented there is not appropriate since their subcontracting costs mainly depend on the customers' distance to the depot, whereas in real-world small package shipping, prices charged by a subcontractor are based on customer demand.

Of the original CVRP instances, we utilize the depot and customer coordinates, the customer demand values and the vehicle capacities. The fixed vehicle cost F and the standard subcontracting price p_i for each customer are computed as follows. Let $C(x^*)$ be the objective function value and k^* the number of vehicles of a high-quality solution to the respective CVRP base instance.

The fixed usage cost of a vehicle is then computed as $F = C(x^*)/k^*$ rounded down to the nearest integer. The standard subcontracting price of customer $j \in J$ is calculated to

$$p_j = 1.5 \cdot q_j \cdot \frac{(F \cdot k^*) + C(x^*)}{\tilde{q}},$$

where q_j denotes the demand of customer j and $\tilde{q} = \sum_{i \in J} q_i$ the total demand of all customers of the problem instance. We restrict the number of vehicles available at the depot to k^* and the minimum demand T to be served by the private fleet is set to $0.7 \cdot \tilde{q}$. In this way, we generate 34 single-depot benchmark problems with up to 483 customers, which are divided into two sets. Set CEP is based on the CVRP instances of Christofides and Eilon (1969), and the second set, named GP, contains large-scale instances adapted from the CVRP problems presented by Golden et al. (1998). Finally, similar to the multi-depot case, we compute a simple upper bound for all benchmark instances by adding the vehicle fixed cost, calculated as described above, to a high-quality solution of the corresponding CVRP instance. More precisely, for the instances proposed by Christofides and Eilon (1969), detailed solutions are available at http://neumann.hec.ca/chairedistributique/data/vrp/old/, while for the instances of Golden et al. (1998), we use the solutions presented by Mester and Bräysy (2007). These VRP solutions correspond to high-quality solutions without subcontracting.

Parameter setting

We conducted a reasonable amount of parameter tuning using a randomly chosen subset of the PCMDVRPNL instances described above. The tuning procedure follows the approach described in Ropke and Pisinger (2006): Starting from a base parameter setting adopted from the AVNS in Stenger et al. (2012), we fine tune each parameter in turn, always keeping the best setting found for a parameter and

Table 1 Results obtained on a subset of PCMDVRPNL instances with different parameter settings

Simulated annealing			
θ_{init}	20	0	**50**
Δ_{avg}	0.0 %	0.87 %	−0.17 %
η_{dec}	0.9995	0.999	**0.9998**
Δ_{avg}	0.0 %	0.03 %	−0.34 %
Penalties			
Pen_{min}	**1**	10	100
Δ_{avg}	0.0 %	0.11 %	0.45 %
Pen_{max}	1,000	500	**2,500**
Δ_{avg}	0.0 %	−0.05 %	−0.10 %
Adaptive mechanism			
ϱ	0.3	**0.5**	0.7
Δ_{avg}	0.0 %	−0.16 %	−0.08 %
ϕ	30	**15**	45
Δ_{avg}	0.0 %	−0.55 %	−0.02 %
$\delta_1/\delta_2/\delta_3$	9/2/1	9/5/1	**9/2/0**
Δ_{avg}	0.0 %	−0.13 %	−0.14 %

proceeding with the next one. In this way, we conducted tests with different values for the simulated annealing starting temperature θ_{init}, the cooling factor η_{dec}, the lower (Pen_{min}) and upper bound (Pen_{max}) of the penalty factors used in the cost function, the ϱ reaction factor of the adaptive mechanism, the number of iterations ϕ in an evaluation period of the adaptive mechanism and finally different value combinations for the scores δ_1, δ_2 and δ_3. For each parameter setting, we conducted ten runs and used average solution costs to assess the performance. Table 1 reports for all parameter settings the percentage deviation Δ_{avg} to the average result obtained with the base parameter setting (shown in the first column). The best results obtained and thus the final setting are marked in bold. Moreover, the final setting uses a penalty initialization factor of $Pen_{init} = 100$. The search is stopped after 1,500 iterations without improvement or after 2,500 s of computing time. This provided a good tradeoff between solution quality and computation times.

Performance of AVNS-RN on PC(MD)VRPNL and the influence of non-linear subcontracting cost

Our problem considers a stepwise cost function that models the real-world scenario of the small packaging market, where subcontractors offer discounts depending on the assigned package volume (see section "Mathematical model of the PCMDVRPNL"). In our benchmark instances for PCMDVRPNL and PCVRPNL, we set the maximal discount e_{max} given by a subcontractor to 0.4. A new discount step is reached every time the subcontracted demand g_l^{sub} exceeds $\vartheta = 0.8$ of the vehicle capacity Q. In order to evaluate the impact of the non-linear cost function,

we additionally performed tests on the benchmark instances using a linear cost function $1 - \frac{e_{\max}}{\tilde{q}-T} \cdot g_l^{\text{sub}}$ (see Fig. 4).

In the following, we present the results obtained with our AVNS-RN on the newly designed benchmark sets of PCMDVRPNL and PCVRPNL in their original form and with a linearized cost function. For all problem instances, we present the upper bound computed as explained above (Cost_{UB} and the number of vehicles k_{UB}), the best solution found in ten runs ($\text{Cost}_{\text{best}}$ and k_{best}), the deviation of this solution from the upper bound in percentage (Δ_{best}), the average solution cost over the ten runs plus the deviation from the upper bound (Cost_{avg} and Δ_{avg}) and finally the average percentage of subcontracted customers (SC_{avg}). For the original instances, we additionally report the average computing time in seconds (CPU(s)), computing times for the linearized instances are not reported due to their similarity.

Table 2 reports the results obtained on the benchmark instances for the PCMDVRPNL. Since the values of Cost_{UB} and k_{UB} correspond to a very competitive solution of the specific instance without subcontracting, the results show that our algorithm is clearly able to identify those customers that can be profitably subcontracted and to determine efficient vehicle routes. On average, our algorithm proves able to find solutions that improve the upper bound by more than 11 %. This result also holds if the average solution quality Cost_{avg} is considered, which proves the robustness of AVNS-RN. The deviations from the upper bound as well as the number of vehicles required are almost equal for both cost functions, while the number of subcontracted customers is higher for linear costs. This can be explained by the fact that in case of the stepwise function, the algorithm tries to reach a discount step. If only a few customers are left within the delivery radius, subcontracting additional customers is most likely not profitable. By contrast, the linear function rewards each additionally subcontracted customer by an increased discount.

Table 3 depicts the results obtained on the single-depot PCVRPNL instances.

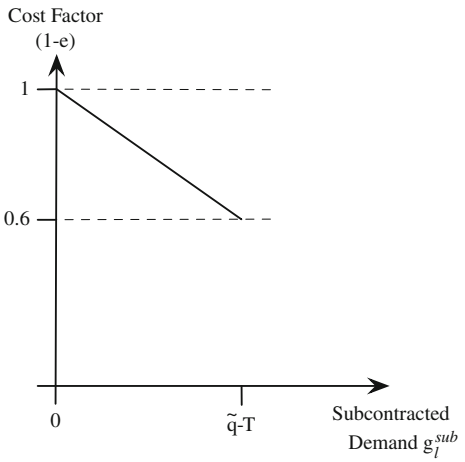

Fig. 4 Example of the linear discount function used as comparison method

Table 2 Results obtained on the original benchmark set for the PCMDVRPNL and the set with a linearized cost function

Instance	$Cost_{UB}$	k_{UB}	PCMDVRPNL-original							PCMDVRPNL-linearized						
			$Cost_{best}$	$\Delta_{best}(\%)$	k_{best}	$Cost_{avg}$	$\Delta_{avg}(\%)$	$SC_{avg}(\%)$	CPU(s)	$Cost_{best}$	$\Delta_{best}(\%)$	k_{best}	$Cost_{avg}$	$\Delta_{avg}(\%)$	$SC_{avg}(\%)$	CPU(s)
S-p01	1,236.87	11	1,100.48	−11.03	7	1,103.52	−10.78	29.20	17.4	1,109.77	−10.28	7	1,113.31	−9.99	31.80	34.6
S-p02	973.53	5	937.21	−3.73	4	939.89	−3.46	21.80	23.5	924.80	−5.01	4	925.39	−4.94	23.00	77.2
S-p03	1,301.19	11	1,153.46	−1,1.35	7	1,162.33	−10.67	32.40	32.6	1,169.48	−10.12	7	1,175.79	−9.64	34.00	141.8
S-p04	1,901.04	15	1,767.80	−7.01	11	1,772.70	−6.75	26.20	62.7	1,765.13	−7.15	11	1,771.40	−6.82	25.70	127.6
S-p05	1,550.03	8	1,371.05	−11.55	6	1,371.38	−11.53	24.00	125.5	1,369.13	−11.67	6	1,369.32	−11.66	24.90	245.3
S-p06	1,836.50	16	1,626.84	−11.42	11	1,632.94	−11.08	30.50	77.7	1,633.25	−11.07	11	1,639.85	−10.71	31.50	136.7
S-p07	1,841.97	16	1,647.37	−10.56	11	1,655.12	−10.14	27.70	48.0	1,634.94	−11.24	11	1,647.24	−10.57	28.10	69.0
S-p08	8,872.78	25	8,309.24	−6.35	18	8,391.77	−5.42	24.78	786.9	8,341.30	−5.99	18	8,401.82	−5.31	24.38	1,169.9
S-p09	7,498.66	26	7,177.88	−4.28	18	7,213.60	−3.80	30.64	621.9	7,061.09	−5.84	18	7,157.66	−4.55	30.52	1,007.7
S-p10	7,271.22	26	6,506.43	−10.52	18	6,553.59	−9.87	32.73	522.2	6,488.34	−10.77	18	6,534.17	−10.14	33.69	879.3
S-p11	7,186.06	26	6,313.71	−12.14	18	6,362.00	−11.47	36.51	461.9	6,292.37	−12.44	18	6,322.78	−12.01	36.22	692.5
S-p12	2,598.95	8	2,247.61	−13.52	6	2,252.07	−13.35	35.00	61.6	2,255.24	−13.22	6	2,256.68	−13.17	33.13	111.8
S-p13	2,598.95	8	2,219.79	−14.59	6	2,229.15	−14.23	40.63	73.3	2,201.04	−15.31	6	2,209.11	−15.00	41.38	92.0
S-p14	2,800.12	8	2,363.92	−15.58	6	2,363.92	−15.58	40.00	68.1	2,354.37	−15.92	6	2,393.80	−14.51	41.25	110.8
S-p15	5,065.42	16	4,363.31	−13.86	11	4,374.82	−13.63	38.63	130.9	4,350.50	−14.11	11	4,352.80	−14.07	41.06	223.2
S-p16	5,132.23	16	4,354.42	−15.16	11	4,358.93	−15.07	40.88	137.6	4,340.03	−15.44	11	4,360.61	−15.03	41.56	205.4
S-p17	5,269.09	16	4,372.57	−17.01	11	4,377.25	−16.93	46.94	237.8	4,358.64	−17.28	11	4,373.26	−17.00	46.63	257.4
S-p18	7,382.85	23	6,204.83	−15.96	16	6,218.06	−15.78	47.79	264.9	6,181.17	−16.28	16	6,213.61	−15.84	48.04	345.6
S-p19	7,667.06	24	6,126.71	−20.09	16	6,149.59	−19.79	54.58	323.6	6,108.85	−20.32	16	6,137.21	−19.95	54.63	495.8
S-p20	7,898.07	24	6,194.82	−21.57	16	6204.68	−21.44	55.17	265.5	6,170.75	−21.87	16	6,177.82	−21.78	54.67	439.0
S-p21	1,0914.84	34	8,955.33	−17.95	24	8,972.64	−17.79	56.44	399.9	8,939.03	−18.10	24	8,960.02	−17.91	56.08	1056.3
S-p22	1,1462.16	36	9,238.15	−19.40	24	9,259.96	−19.21	55.14	328.6	9,228.07	−19.49	24	9,258.82	−19.22	54.97	895.7
S-p23	11,838.75	36	9,458.06	−20.11	24	9,500.82	−19.75	53.42	502.3	9,436.26	−20.29	24	9,477.05	−19.95	53.53	958.7

Table 2 continued

Instance	$Cost_{UB}$	k_{UB}	PCMDVRPNL-original							PCMDVRPNL-linearized						
			$Cost_{best}$	$\Delta_{best}(\%)$	k_{best}	$Cost_{avg}$	$\Delta_{avg}(\%)$	$SC_{avg}(\%)$	CPU(s)	$Cost_{best}$	$\Delta_{best}(\%)$	k_{best}	$Cost_{avg}$	$\Delta_{avg}(\%)$	$SC_{avg}(\%)$	CPU(s)
S-pr01	1,741.32	4	1,606.23	−7.76	3	1,606.23	−7.76	14.58	44.7	1,595.28	−8.39	3	1,596.66	−8.31	14.38	79.6
S-pr02	2,587.34	8	2,305.25	−10.90	6	2,315.72	−10.50	13.13	101.0	2,301.32	−11.05	6	2,305.21	−10.90	12.50	201.6
S-pr03	3,723.80	12	3,239.03	−13.02	8	3,286.14	−11.75	10.21	173.8	3,240.77	−12.97	8	3,251.62	−12.68	17.71	297.9
S-pr04	3,978.31	16	3,595.55	−9.62	13	3,629.45	−8.77	8.65	218.9	3,575.24	−10.13	12	3,595.98	−9.61	10.26	414.2
S-pr05	4,731.20	20	4,548.49	−3.86	18	4,572.15	−3.36	8.13	287.0	4,522.94	−4.40	18	4,554.05	−3.74	9.00	407.5
S-pr06	5,556.30	24	5,204.22	−6.34	20	5,268.48	−5.18	6.04	364.9	5,185.81	−6.67	20	5,216.69	−6.11	11.35	617.3
S-pr07	2,169.56	6	1,969.81	−9.21	5	1,969.81	−9.21	8.33	33.5	1,968.15	−9.28	5	1,968.15	−9.28	8.33	63.8
S-pr08	3,344.85	12	2,987.89	−10.67	9	3,031.42	−9.37	14.79	126.7	2,955.41	−11.64	8	2,967.18	−11.29	24.51	212.1
S-pr09	4,293.20	18	3,940.58	−8.21	14	3,963.50	−7.68	15.56	209.7	3,899.89	−9.16	13	3,927.11	−8.53	20.09	435.1
S−pr10	5,748.26	24	5,383.28	−6.35	19	5,435.59	−5.44	20.35	365.5	5,344.27	−7.03	18	5,390.83	−6.22	22.99	870.3
Average		17.5		−11.84	12.6		−11.41	30.33	227.3		−12.12	12.5		−11.71	31.57	405.2

Table 3 Results obtained on the original benchmark set for the PCVRPNL and the set with a linearized cost function

Instance	$Cost_{UB}$	k_{UB}	PCVRPNL-original							PCVRPNL-linearized						
			$Cost_{best}$	$\Delta_{best}(\%)$	k_{best}	$Cost_{avg}$	$\Delta_{avg}(\%)$	$SC_{avg}(\%)$	CPU(s)	$Cost_{best}$	$\Delta_{best}(\%)$	k_{best}	$Cost_{avg}$	$\Delta_{avg}(\%)$	$SC_{avg}(\%)$	CPU(s)
CEP-01	1,049.22	5	946.16	−9.42	4	946.1629	−9.42	30.00	136.5	990.37	−5.61	4	990.37	−5.61	30.00	211.5
CEP-02	1,670.52	11	1,482.8	−10.69	7	1,483.93	−10.62	36.27	136.3	1,486.50	−11.02	7	1,490.03	−10.80	36.67	294.8
CEP-03	1,652.28	8	1,455	−11.83	6	1,459.75	−11.54	35.50	321.8	1,480.13	−10.42	6	1,485.41	−10.10	31.80	431.3
CEP-04	2,056.84	12	1,770.7	−13.56	8	1,798.071	−12.22	34.53	582.5	1,783.30	−13.30	8	1,786.04	−13.17	37.53	1,135.6
CEP-05	2,590.82	17	2,324.6	−10.02	12	2,333.105	−9.69	31.61	439.2	2,367.08	−8.64	12	2,374.45	−8.35	35.43	1025.1
CEP-06	1,110.86	6	908.71	−17.94	4	908.7138	−17.94	30.00	138.2	955.58	−13.98	4	955.58	−13.98	30.00	202.8
CEP-07	1819.36	11	1514.4	−16.41	7	1,515.452	−16.35	35.73	134.9	1,519.03	−16.51	7	1,522.01	−16.34	36.67	267.5
CEP-08	1,731.88	9	1,432.2	−17.21	6	1,436.033	−16.99	35.90	285.9	1,460.87	−15.65	6	1,461.23	−15.63	29.40	491.2
CEP-09	2,325.10	14	1,826.9	−21.41	8	1,869.036	−19.60	34.13	497.8	1,841.23	−20.81	8	1,843.85	−20.70	37.13	1,046.6
CEP-10	2,805.62	18	2,335.2	−16.05	12	2,344.136	−15.73	31.86	509.9	2,388.57	−14.86	12	2,396.00	−14.60	34.22	995.5
CEP-11	2,084.22	7	1918.3	−7.69	5	1,922.233	−7.50	31.08	601.6	1,946.83	−6.59	5	1,947.56	−6.56	31.83	812.8
CEP-12	1,639.12	10	1,471.6	−9.69	7	1,474.877	−9.49	25.30	247.8	1,512.88	−7.70	7	1,516.38	−7.49	29.90	310.6
CEP-13	3,099.88	11	2126.8	−30.97	5	2,130.561	−30.85	30.83	794.1	2,125.99	−31.42	5	2,166.10	−30.12	31.25	1,188.1
CEP-14	1,732.74	11	1,470.4	−14.73	7	1,473.93	−14.52	24.70	266.2	1,517.90	−12.40	7	1,521.28	−12.20	29.20	288.9
Average CEP		10.7		−14.83	7.0		−14.46	31.96	363.7		−13.49	7.0		−13.26	32.93	621.6
GP-01	11,255.08	9	9,901.5	−12.03	7	9,971.30	−11.41	28.08	1,459.0	1,0322.18	−8.29	7	10,368.34	−7.88	32.08	2161.1
GP-02	16,895.84	10	14,157	−16.21	7	14,237.13	−15.74	36.31	2,503.3	14,623.34	−13.45	7	14,712.13	−12.92	42.25	2,502.3
GP-03	22,072.44	10	18,246	−17.34	7	18,449.12	−16.42	35.80	25,13.9	19,081.50	−13.55	7	19,307.47	−12.53	39.00	2,504.7
GP-04	27,249.04	10	23,416	−14.07	7	23,531.87	−13.64	41.58	2,504.9	24,008.95	−11.89	7	24,218.14	−11.12	43.04	2,505.8
GP-05	12,921.96	5	11,224	−13.14	4	11,375.99	−11.96	33.90	2,081.5	11,778.76	−8.85	4	11,804.36	−8.65	34.25	2,449.7
GP-06	16,825.76	7	13878	−17.52	5	13,920.43	−17.27	37.43	2,463.0	14,603.88	−13.21	5	14,677.29	−12.77	44.39	2,529.3
GP-07	20,391.12	9	16,825	−17.49	6	16,923.12	−17.01	40.00	2,503.9	17,299.10	−15.16	6	17,412.11	−14.61	45.17	2,502.8
GP-08	23,327.10	8	20,687	−11.32	8	20,761.16	−11.00	28.50	2,502.2	20,499.41	−12.12	7	21,677.00	−7.07	32.39	2,503.7
GP-09	1,166.78	14	1,023.8	−12.25	10	1,027.76	−11.91	45.22	2,083.9	1,027.17	−11.97	10	1,031.03	−11.63	47.80	2,524.5

Table 3 continued

Instance	$Cost_{UB}$	k_{UB}	PCVRPNL-original							PCVRPNL-linearized						
			$Cost_{best}$	$\Delta_{best}(\%)$	k_{best}	$Cost_{avg}$	$\Delta_{avg}(\%)$	$SC_{avg}(\%)$	CPU(s)	$Cost_{best}$	$\Delta_{best}(\%)$	k_{best}	$Cost_{avg}$	$\Delta_{avg}(\%)$	$SC_{avg}(\%)$	CPU(s)
GP-10	1,483.12	16	1,288.7	−13.11	11	1,299.12	−12.41	45.17	2,524.3	1,289.98	−13.02	11	1,297.25	−12.53	48.08	2,564.4
GP-11	1,836.90	18	1,611.7	−12.26	12	1,642.25	−10.60	45.29	2,502.0	1,643.03	−10.55	13	1,650.10	−10.17	47.44	2,502.6
GP-12	2,214.38	18	2,076.3	−6.24	14	2,092.78	−5.49	39.54	2,503.5	2,032.75	−8.20	14	2,054.12	−7.24	47.95	2,507.9
GP-13	1,718.22	26	1,551.7	−9.69	18	1,563.60	−9.00	40.79	1,024.3	1,546.71	−9.98	18	1,565.77	−8.87	45.24	1,374.7
GP-14	2,162.62	30	1,942.8	−10.16	21	1,960.02	−9.37	41.72	1,197.3	1,932.97	−10.62	21	1,944.59	−10.08	45.66	2,093.1
GP-15	2,690.46	33	2,414.3	−10.26	24	2,424.33	−9.89	41.87	1,908.8	2,420.44	−10.04	24	2,429.85	−9.69	46.24	2,509.4
GP-16	3,245.38	37	2,905	−10.49	27	2,919.19	−10.05	42.17	2,452.8	2,883.69	−11.14	26	2,914.68	−10.19	46.65	2,526.2
GP-17	1,415.58	22	1,292.5	−8.69	16	1,298.40	−8.28	27.17	497.6	1,306.92	−7.68	16	1,310.22	−7.44	29.92	836.7
GP-18	1,997.46	27	1,776.3	−11.07	19	1,811.41	−9.31	28.90	1,001.0	1,779.08	−10.93	19	1,815.24	−9.12	30.53	1543.1
GP-19	2,733.72	33	2,420.9	−11.44	23	2,451.35	−10.33	27.33	966.4	2,423.38	−11.35	23	2,440.80	−10.71	30.28	1,937.8
GP-20	3,640.18	38	3,191.9	−12.31	27	3,200.68	−12.07	28.69	1,767.6	3,202.29	−12.03	27	3,216.71	−11.63	30.40	2,410.8
Average GP		19.0		−12.36	13.7	7,643.05	−11.66	36.77	1,948.1	7,785.2764	−11.20	13.6	7,892.36	−10.34	40.44	2,249.5
Total average		15.9		−13.51	10.9		−12.95	34.79	1295.7		−12.15	10.9		−11.54	37.35	1,579.2

For both discount functions, the solutions improve the VRP-based upper bound by more than 12 % while requiring moderate computing times. In addition, the number of vehicles required is reduced by more than 30 %. This shows again that our algorithm is able to identify customers that can be profitably subcontracted and to construct cost-efficient vehicle routes. Comparing the solutions obtained with the two different cost functions, the deviation from the upper bound Cost_{UB} as well as the number of vehicles required are almost equal for both cost functions. However, almost 7 % less customers are subcontracted when the stepwise cost function is considered. This can be explained by the fact that the stepwise function reaches $e_{\max}$ earlier, i.e., with less subcontracted demand. In case of the linear function, increasing the subcontracted demand can always be profitable up to $\tilde{q} - T$ units as the discount factor continuously increases.

Compared to the multi depot case, the average improvement of the upper bound is higher. In the multi-depot problem different subcontractors with a limited delivery radius and capacity exist. Thus, the demand of outsourced customers is distributed among several subcontractors and maximal possible discounts are not reached. Our algorithm tries to maximize the discount by filling the vehicles of a subcontractor but the restricted delivery radiuses often counteracts this objective.

Varying the minimum demand to be delivered by the private fleet

One of the main characteristics of prize-collecting problems is the lower bound on the prize to be collected or, in our case, the minimum customer demand T that has to be served by the private fleet. Since the value of T strongly influences the outsourcing decision, we perform tests with different values of T to quantify the effect on the overall solution value. In detail, we used the generated PCMDVRPNL benchmark instances with the standard subcontracting price p_i and without any discount function. We varied the value of T between $0.5\tilde{q}$ and $0.9\tilde{q}$ in steps of $0.1\tilde{q}$ and solved each instance 10 times with this parameter setting. Figure 5 depicts the average gap of the best solutions found to the upper bound as well as the average number of subcontracted customers for each value of $\frac{T}{\tilde{q}}$ on the PCMDVRPNL instances.

With increasing value of $\frac{T}{\tilde{q}}$, the flexibility to outsource customers decreases and the solution quality clearly suffers. Similarly, the number of subcontracted customers decreases when the minimum demand to be served by the private fleet is increased. However, reducing T below $0.7\tilde{q}$ has only a very slight influence on the gap to the upper bound while still significantly increasing the number of subcontracted customers. This indicates that after the most unprofitable customers have been outsourced, a high number of solutions with similar solution quality exist. Their quality only slightly depends on further subcontracting customers.

Although the rough tendency of the outcome of this study appears expectable, the results show the strong influence of the important real-world constraint defining a lower bound on the demand served by the private fleet. In addition, the results prove again the suitability of our algorithm to handle the subcontracting decision while paying attention to the prize-collecting constraint.

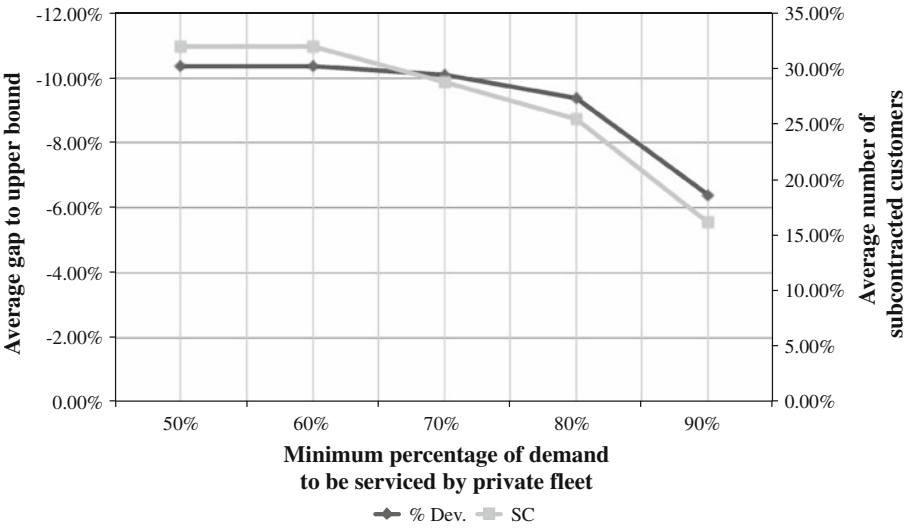

Fig. 5 Comparing results obtained with different values of the lower bound T on the multi-depot instances

Evaluating the algorithmic performance on benchmark instances of related problems

To prove the performance of the proposed AVNS-RN, we apply the algorithm to benchmark instances of the closely related MDVRPPC and the VRPPC. The MDVRPPC test instances are described in Stenger et al. (2012) and are available for download. We compare the results obtained with our AVNS-RN to those of the AVNS of Stenger et al. (2012) (abbreviated as SVES in the table). Table 4 reports for each instance the best known solution (BKS), either taken from Stenger et al. (2012) or found during the overall testing of our algorithm. For both algorithms, we report the best solution found in ten runs ($Cost_{best}$) and its percentage deviation from the BKS (Δ_{best}), the average solution cost over the ten runs and its deviation from the BKS ($Cost_{avg}$ and Δ_{avg}) and the average computing time in seconds (CPU(s)). If the BKS is found by one of the two algorithms during the ten runs this is indicated in bold.

Although our AVNS-RN is specifically adapted to the PCMDVRPNL and the parameter tuning is carried out on PCMDVRPNL instances, it obtains competitive results that even slightly improve on the average solution quality, requiring basically identical run-time. Furthermore, we found new best solutions for two benchmark instances during these 10 runs (BKS marked bold) and a total of 16 new best solutions during our overall testing (BKS marked in italics). These results further confirm the competitiveness of the proposed method.

For the VRPPC, Bolduc et al. (2008) designed 34 benchmark instances that can be downloaded at http://www.mcbolduc.com/VRPPC/tests.htm. The benchmark instances are based on the same VRP instances as those proposed for PCVRPNL.

Table 4 Results obtained on MDVRPPC benchmark instances compared to AVNS algorithm of Stenger et al. (2012)

Instance	BKS	SVES					AVNS-RN				
		$Cost_{best}$	$\Delta_{best}(\%)$	$Cost_{avg}$	$\Delta_{avg}(\%)$	CPU(s)	$Cost_{best}$	$\Delta_{best}(\%)$	$Cost_{avg}$	$\Delta_{avg}(\%)$	CPU(s)
s-p01	1,145.01	**1,145.01**	0.00	1,148.78	0.33	13.3	**1,145.01**	0.00	1,148.30	0.29	10.7
s-p02	937.21	**937.21**	0.00	940.52	0.35	20.5	**937.21**	0.00	938.32	0.12	16.0
s-p03	1,192.51	**1,192.51**	0.00	1,199.86	0.62	30.6	**1,192.51**	0.00	1,199.49	0.59	20.7
s-p04	1,827.27	1,828.74	0.08	1,834.95	0.42	34.7	1,831.94	0.26	1,838.41	0.61	27.7
s-p05	1,406.68	**1406.68**	0.00	1,411.72	0.36	88.7	**1,406.68**	0.00	1,413.04	0.45	82.1
s-p06	*1,686.32*	1,686.59	0.02	1,689.93	0.21	36.5	1,689.37	0.18	1,692.14	0.35	33.0
s-p07	*1,666.03*	1668.53	0.15	1,676.36	0.62	32.4	1,672.18	0.37	1,683.12	1.03	27.6
s-p08	*8,625.75*	8,661.27	0.41	8,702.42	0.89	233.2	8,673.50	0.55	8,706.75	0.94	156.7
s-p09	*7,251.74*	7,258.54	0.09	7,334.25	1.14	268.4	7,275.68	0.33	7,324.22	1.00	321.2
s-p10	*6,606.12*	6,655.90	0.75	6,679.96	1.12	179.0	6,639.91	0.51	6,688.54	1.25	249.9
s-p11	6,353.59	**6,353.59**	0.00	6,386.66	0.52	203.5	6,355.02	0.02	6,409.87	0.89	318.9
s-p12	2,303.31	**2,303.31**	0.00	2,307.33	0.17	53.2	**2,303.31**	0.00	2,304.99	0.07	32.6
s-p12	2,278.14	**2,278.14**	0.00	**2,278.14**	0.00	38.8	**2,278.14**	0.00	2,278.14	0.00	40.9
s-p14	2,404.72	**2,404.72**	0.00	2,407.28	0.11	41.0	**2,404.72**	0.00	2,404.72	0.00	35.8
s-p15	*4,456.02*	4,459.41	0.08	4,471.65	0.35	73.8	4,459.41	0.08	4,473.90	0.40	58.3
s-p16	4,449.41	**4,449.41**	0.00	4,465.02	0.35	67.3	**4,449.41**	0.00	4,461.24	0.27	67.8
s-p17	4,442.24	4,448.10	0.13	4,480.52	0.86	102.9	4,446.10	0.09	4,485.21	0.97	76.3
s-p18	*6,269.70*	6,272.70	0.05	6,313.71	0.70	122.3	6,270.70	0.02	6,284.63	0.24	94.4
s-p19	6,164.91	6,166.91	0.03	6,185.15	0.33	121.7	6,166.91	0.03	6,178.20	0.22	148.3
s-p20	**6,240.66**	6,245.90	0.08	6259.78	0.31	102.0	**6,240.66**	0.00	6,257.98	0.28	114.9
s-p21	8,758.25	8,759.25	0.01	8,790.24	0.37	178.9	8,767.11	0.10	8,818.00	0.68	187.0
s-p22	9,275.24	**9,275.24**	0.00	9,299.56	0.26	118.1	9,278.24	0.03	9,295.21	0.22	203.2
s-p23	*9,546.29*	9,581.64	0.37	9,609.59	0.66	162.1	9,557.54	0.12	9,598.85	0.55	188.4

Table 4 continued

Instance	BKS	SVES					AVNS-RN				
		$Cost_{best}$	$\Delta_{best}(\%)$	$Cost_{avg}$	$\Delta_{avg}(\%)$	CPU(s)	$Cost_{best}$	$\Delta_{best}(\%)$	$Cost_{avg}$	$\Delta_{avg}(\%)$	CPU(s)
s-pr01	1,606.23	**1,606.23**	0.00	1,627.51	1.32	34.7	**1,606.23**	0.00	1,613.14	0.43	21.8
s-pr02	*2,303.99*	2,306.76	0.12	2,330.14	1.14	74.8	2,304.99	0.04	2,323.08	0.83	62.5
s-pr03	*3,278.81*	3,283.52	0.14	3,319.86	1.25	129.7	3,282.01	0.10	3,303.94	0.77	113.9
s-pr04	*3,573.21*	3,598.16	0.70	3,666.68	2.62	213.5	3,603.69	0.85	3,652.40	2.22	158.6
s-pr05	4,519.67	**4,519.67**	0.00	4,569.21	1.10	170.4	4,525.87	0.14	4,563.84	0.98	168.7
s-pr06	*5,202.88*	5,232.78	0.57	5,271.81	1.32	329.2	5,239.48	0.70	5,268.96	1.27	197.1
s-pr07	1,969.81	**1,969.81**	0.00	**1,969.81**	0.00	29.1	**1,969.81**	0.00	1,969.81	0.00	22.5
s-pr08	*2,979.94*	2,994.37	0.48	3,038.74	1.97	99.8	2,996.38	0.55	3,043.59	2.14	84.3
s-pr09	**3,913.70**	3,914.22	0.01	3,967.92	1.39	166.9	**3,913.70**	0.00	3,957.01	1.11	150.9
s-pr10	*5,399.10*	5,457.46	1.08	5,499.17	1.85	263.8	5,417.12	0.33	5,497.26	1.82	222.9
Average			0.16		0.76	116.2		0.16		0.69	112.6

Values in bold indicate the best known solution

Values in italics indicate a new best known solution found during overall testing

Table 5 Results of AVNS-RN on VRPPC benchmark instances

Instance	BKS	TS+		TS		AVNS		AVNS-RN		
		$\Delta_{best}(\%)$	CPU(s)	$\Delta_{best}(\%)$	CPU(s)	$\Delta_{best}(\%)$	CPU(s)	$\Delta_{best}(\%)$	$\Delta_{avg}(\%)$	CPU(s)
CE-01	1,119.47	**0.00**	24.9	**0.00**	24.3	0.40	92.5	**0.00**	0.41	81.2
CE-02	1,814.52	**0.00**	33.9	**0.00**	33.0	**0.00**	48.6	**0.00**	0.13	63.4
CE-03	1,919.05	0.60	81.0	0.11	78.6	0.09	212.1	**0.00**	0.65	258.1
CE-04	2,507.44	0.71	200.6	0.71	193.2	0.18	279.7	0.07	0.74	179.6
CE-05	3,097.67	0.63	353.2	0.51	309.9	0.07	228.6	0.45	0.76	132.9
CE-06	1,207.47	**0.00**	25.2	**0.00**	25.5	0.03	75.9	**0.00**	0.11	79.8
CE-07	2,004.53	0.10	34.0	0.10	32.7	0.47	50.9	**0.00**	0.21	61.0
CE-08	2,052.05	0.22	81.6	0.40	85.1	**0.00**	253.1	**0.00**	0.50	251.1
CE-09	2,422.74	0.55	188.2	0.65	185.3	0.40	259.0	0.35	0.71	190.8
CE-10	3,383.00	0.56	345.7	0.70	311.1	0.25	201.0	0.18	0.71	162.2
CE-11	2,330.94	0.06	131.0	0.96	126.3	0.05	316.0	**0.00**	0.06	370.5
CE-12	1,952.86	**0.00**	59.5	**0.00**	60.4	0.04	92.9	0.04	0.04	107.5
CE-13	2,858.94	0.07	132.1	0.83	130.0	**0.00**	278.5	**0.00**	0.07	351.4
CE-14	2,213.02	0.18	64.2	0.18	65.0	0.11	93.2	0.11	0.11	148.8
Average CE		0.26	125.4	0.37	118.6	0.15	177.3	0.09	0.37	174.2
G-01	14,123.38	0.47	1,183.8	0.68	638.3	0.24	652.6	0.23	0.39	979.3
G-02	19,155.17	0.28	5,220.6	3.00	1,215.2	0.26	1,558.4	0.17	0.67	1,809.6
G-03	24,411.37	0.74	5,940.4	5.09	2,241.3	0.78	2,356.1	0.51	1.21	2,267.4
G-04	34,275.11	1.54	5,508.7	5.10	3,833.9	0.41	2,500.9	0.76	1.19	2,241.1
G-05	14,229.24	0.23	847.8	3.12	875.0	0.30	1,301.1	0.33	0.63	2263.5
G-06	**21,396.38**	0.48	1,591.1	4.13	14,45.3	0.21	1,783.5	**0.00**	0.62	2,049.0
G-07	23,375.60	0.59	5,514.0	3.49	2,052.8	**0.00**	2,262.8	0.25	0.99	2,107.7
G-08	29,712.97	1.21	5,729.0	3.08	3,059.9	0.28	2,339.7	0.36	0.70	1,907.3

Table 5 continued

Instance	BKS	TS+		TS		AVNS		AVNS-RN		
		$\Delta_{best}(\%)$	CPU(s)	$\Delta_{best}(\%)$	CPU(s)	$\Delta_{best}(\%)$	CPU(s)	$\Delta_{best}(\%)$	$\Delta_{avg}(\%)$	CPU(s)
G-09	1,321.73	0.29	819.1	0.48	611.0	1.04	602.0	0.78	1.49	704.1
G-10	1,583.82	0.44	1,762.3	0.44	938.8	1.31	978.4	1.30	1.97	864.9
G-11	2,163.34	0.48	3,284.3	0.41	1,492.7	1.19	1,534.3	1.17	1.95	1,311.0
G-12	2,479.62	0.62	8,587.6	0.53	2,309.7	1.64	2,043.9	1.98	2.52	1,975.8
G-13	2,268.32	0.26	504.5	0.47	360.8	1.04	116.5	0.68	1.15	170.8
G-14	2,692.46	0.40	976.9	0.47	610.4	0.59	183.5	0.94	1.34	215.6
G-15	3,153.55	0.24	1,952.0	0.17	924.8	1.31	357.3	0.76	1.44	221.0
G-16	3,631.47	0.19	4,675.1	0.21	1,313.7	1.10	561.2	1.19	1.61	370.0
G-17	1,632.83	0.03	472.0	0.20	402.6	3.04	110.3	2.40	3.78	182.3
G-18	2,691.61	0.69	892.0	0.53	622.0	1.86	156.4	2.23	2.58	211.5
G-19	3,442.16	1.61	1,418.4	1.61	1,012.5	1.91	194.1	2.17	2.58	234.3
G-20	4,265.11	0.98	2,476.3	1.08	1,368.9	1.58	290.3	1.94	2.20	241.3
Average G		0.59	2,967.8	1.71	1,366.5	1.00	1,094.2	1.01	1.55	1,116.4
Total average		0.45	1,797.4	1.16	852.6	0.65	716.6	0.63	1.07	728.4

However, the number of available vehicles is reduced such that the total vehicle capacity is not sufficient to serve the demand of all customers. As comparison methods, we use the tabu search with ejection chains described in Potvin and Naud (2011), the tabu search proposed by Côté and Potvin (2009) and the AVNS algorithm of Stenger et al. (2012). Table 5 presents for all algorithms the deviation of the best solution found in ten runs from the best known solution as well as the average computing time in seconds. For our AVNS-RN, we additionally report the deviation of the average solution found to the best known solution. Since Potvin and Naud (2011) mentions only the best solution found, a comparison of the average solution quality is not possible.

The results obtained on the VRPPC instances show again the high efficiency of the AVNS-RN algorithm. Compared to the currently best performing algorithm, the TS with ejection chains of Potvin and Naud (2011), the average gap is slightly worse, but we require significantly less computing time. Furthermore, we are able to improve the results of the AVNS presented in Stenger et al. (2012), while the computing time remains on an equal level. It is also worth mentioning that we are able to find a new overall best solution on instance G-06. Moreover, the average results obtained with AVNS-RN confirm the robustness of the algorithm.

Conclusion

In this paper, we proposed a single and multi-depot version of the PCVRPNL to model an important route planning problem arising in small package shipping. To solve the proposed NP-hard problems, we presented a powerful AVNS algorithm applying cyclic-exchange neighborhoods, which bases on the framework of Stenger et al. (2012). To tackle the requirements of the PCVRPNL, we designed specific route and customer selection methods. In addition, we implemented a random ordering of the neighborhoods used in shaking. Both extensions proved their positive impact on solution quality during testing.

For the computational studies, we designed a set of 34 benchmark instances for the single depot problem and 33 instances for the multi-depot problem. Numerical studies are performed on the newly designed benchmark instances to investigate the suitability of the algorithm for the proposed problems. The tests further demonstrate the strong influence of the value chosen for the minimum demand to be served by the private fleet. In addition, we solved benchmark instances of the closely related VRPPC and MDVRPPC. The results clearly prove the high performance of the proposed algorithm.

References

Balas E (1989) The prize collecting traveling salesman problem. Networks 19(6):621–636

Bolduc M-C, Renaud J, Boctor F (2007) A heuristic for the routing and carrier selection problem. Eur J Oper Res 183(2):926–932

Bolduc M-C, Renaud J, Boctor F, Laporte G (2008) A perturbation metaheuristic for the vehicle routing problem with private fleet and common carriers. J Oper Res Soc 59(6):776–787
Chao IM, Golden BL, Wasil E (1993) A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions. Am J Math Manag Sci 13(3–4):371–406
Chaves AA, Lorena LAN (2008) Hybrid metaheuristic for the prize collecting travelling salesman problem. In: van Hemert J, Cotta C (eds), EvoCOP 2008. LNCS, vol. 4972, Springer, Heidelberg
Christofides N, Eilon S (1969) An algorithm for one vehicle-dispatching problem. Oper Res Q 20(3): 309–318
Chu C-W (2005) A heuristic algorithm for the truckload and less-than-truckload problem. Eur J Oper Res 165(3):657–667
Clarke G, Wright JW (1964) Scheduling of vehicles from a central depot to a number of delivery points. Oper Res 12(4):568–581
Cordeau J-F, Gendreau M, Laporte G (1997) A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 30(2):105–119
Côté J-F, Potvin J-Y (2009) A tabu search heuristic for the vehicle routing problem with private fleet and common carrier. Eur J Oper Res 198(2):464–469
Dell'Amico M, Maffioli F, Sciomachen A (1998) A lagrangian heuristic for the prize collecting travelling salesman problem. Ann Oper Res 81(0):289–305
Dueck G (1993) New optimization heuristics: the great deluge algorithm and the record-to-record travel. J Comput Phys 104(1):86–92
Feillet D, Dejax P, Gendreau M (2005) Traveling salesman problems with profits. Transp Sci 39(2): 188–205
Golden BL, Wasil EA, Kelly JP, Chao I-M (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
Hemmelmayr VC, Doerner KF, Hartl RF (2009) A variable neighborhood search heuristic for periodic routing problems. Eur J Oper Res 195(3):791–802
Lin S (1965) Computer solutions to the traveling-salesman problem. Bell Syst Tech 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
Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24(11):1097–1100
Pirkwieser S, Raidl GR (2008) A variable neighborhood search for the periodic vehicle routing problem with time windows, Proceedings of the 9th EU/Meeting on Metaheuristics for Logistics and Vehicle Routing
Pisinger D, Ropke S (2007) A general heuristic for vehicle routing problems. Comput Oper Res 34(8): 2403–2435
Potvin J-Y, Naud M-A (2011) Tabu search with ejection chains for the vehicle routing problem with private fleet and common carrier. J Oper Res Soc 62(2):326–336
Prosser P, Shaw P (1996) Study of greedy search with multiple improvement heuristics for vehicle routing problems, Technical report, RR96/201, Department of Computer Science, University of Strathclyde, Glasgow, Scotland
Renaud J, Boctor FF, Laporte G (1996) An improved petal heuristic for the vehicle routing problem. J Oper Res Soc 47:329–336
Renaud J, Laporte G, Boctor FF (1996) A tabu search heuristic for the multi-depot vehicle routing problem. Comput Oper Res 23(3):229–235
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
Stenger A, Vigo D, Enz S, Schwind M (2012) An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. Transp Sci 47:64–80
Tang L, Wang X (2006) Iterated local search algorithm based on very large-scale neighborhood for prize-collecting vehicle routing problem. Int J Adv Manuf Technol 29(11–12):1246–1258
Thompson PM, Psaraftis HN (1993) Cyclic transfer algorithms for multivehicle routing and scheduling problems. Oper Res 41(5):935–946
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
Wren A, Holliday A (1972) Computer scheduling of vehicles from one or more depots to a number of delivery points. Oper Res Q 23(3):333–344