OR Spectrum
DOI 10.1007/s00291-010-0194-3

REGULAR ARTICLE

# Vehicle routing with compartments: applications, modelling and heuristics

**Ulrich Derigs · Jens Gottlieb · Jochen Kalkoff · Michael Piesche · Franz Rothlauf · Ulrich Vogel**



**Abstract** Despite the vast amount of literature about vehicle routing problems, only very little attention has been paid to vehicles with compartments that allow transportation of inhomogeneous products on the same vehicle, but in different compartments. We motivate a general vehicle routing problem with compartments that is essential for several industries, like the distribution of food or petrol. We introduce a formal model, an integer program formulation and a benchmark suite of 200 instances. A solver suite of heuristic components is presented, which covers a broad range of alternative approaches for construction, local search, large neighbourhood search and meta-heuristics. The empirical results for the benchmark instances identify effective algorithmic setups as well as essential components for achieving high solution quality. In a comparison on 23 specific and combinatorially less complex instances taken from literature, our algorithm showed to be competitive.

**Keywords** Transportation · Vehicle routing · Compartment · Heuristics

U. Derigs (✉) · M. Piesche · U. Vogel
Universität zu Köln, Pohligstr. 1, 50969 Köln, Germany
e-mail: ulrich.derigs@uni-koeln.de

M. Piesche
e-mail: michael.piesche@uni-koeln.de; michael.piesche@googlemail.com

U. Vogel
e-mail: ulrich.vogel@uni-koeln.de

J. Gottlieb
SAP AG, Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany
e-mail: jens.gottlieb@sap.com

J. Kalkoff
SAP Deutschland AG & Co. KG, Hasso-Plattner-Ring 7, 69190 Walldorf, Germany
e-mail: jochen.kalkoff@sap.com

F. Rothlauf
Universität Mainz, Jakob-Welder-Weg 9, 55128 Mainz, Germany
e-mail: rothlauf@uni-mainz.de

Published online: 11 February 2010

## 1 Introduction

The classical capacitated vehicle routing problem considers a set of customers, each having a certain demand, and a set of vehicles that have the same capacity and are based at a single central depot. A tour is a sequence of customers assigned to one vehicle. The objective is to determine a set of tours that respect all capacity constraints, serve all customers and minimize total distance travelled by the vehicles.

In several industries, all delivered goods are homogeneous in the sense that they can be transported together. However, in other industries the goods are inhomogeneous. To save transportation costs, those industries often employ vehicles with *compartments* in order to allow transporting inhomogeneous goods together on the same vehicle, but in different compartments. In addition to the vehicle capacity, each compartment in the vehicle has its own capacity restriction. The compartment setup of a vehicle may be configurable. In some applications, the separators between the compartments can be adjusted, such that a single compartment could cover any volume up to the total volume of the vehicle. In contrast to these *flexible compartments*, other scenarios consider *fixed compartments* that cannot be configured. We introduce the *vehicle routing problem with compartments* (VRPC), which covers these significantly different problem structures.

While many extensions of the classical *vehicle routing problem* (VRP) were studied extensively in the literature like time windows, distance constraints, backhauls or variants involving pickups and deliveries (cf. Toth and Vigo 2002) leading to the research area termed as rich vehicle routing problems, published research regarding compartments is quite rare.

Most of the published papers concern fuel distribution, i.e. the compartments are tanks which can receive different fuel types. Brown and Graves (1981) developed a real-time computer system for centralized control of distribution of light petroleum products for a major US Oil Corporation. Van der Bruggen et al. (1995) reported on a study for a large oil company where the proper assignment of products to truck compartments was only one subproblem in a hierarchy of redesign problems. A special aspect was here to analyse the effect of allowing switch loading, i.e. not to assign a fixed product to every compartment. Avella et al. (2004) discussed a multi-period routing problem where each compartment (tank) has to travel either completely full or empty. For solving the resulting packing and routing problem, they proposed a simple greedy-like heuristic and a set partitioning-based exact method. Cornillier et al. (2008) developed heuristic approaches for solving the multi-period petrol station replenishment problem, considering trucks with different compartments.

Fagerholt and Christiansen (2000) introduced a bulk ship scheduling problem where each ship in the fleet is equipped with a flexible cargo hold that can be partitioned into several smaller holds in a given number of ways. Jetlund and Karimi (2004) discussed ship scheduling problems for chemical tankers with compartments and proposed a solution approach based on integer programming for a simplified model in which the assignment of cargos to compartments is ignored. Recently, El Fallahi et al. (2008) studied the special problem type where every compartment is dedicated to one product, a scenario which arises in the distribution of cattle food to farms. They presented a memetic algorithm and a tabu search heuristic and evaluated the algorithms on classical

VRP instances from the literature that were modified by adding compartments. Muyldermans and Pang (2007) analysed the improvement of using multiple compartments over single compartments in waste collection, applying a guided local search algorithm for solving the specific routing problems, and Chajakis and Guignard (2003) discussed optimization models for cargo space layout when using multiple compartments.

Our research has been motivated by a demand for methods and software handling compartments from two industries: the distribution of petrol and food. The distribution of petrol typically involves different fuel types and vehicles with five compartments. Thus, a single petrol station demanding several different fuel types can be served by one vehicle with several compartments. Each compartment can carry any product, but different products must not be mixed in one compartment.

In food retail, transportation involves frozen and dry goods which need special equipment for fresh delivery. Typical vehicles have two compartments, one served by a refrigerator and one for dry goods. This allows serving stores that demand frozen and dry goods at the same time by one vehicle with two compartments. In this industry, certain product-compartment combinations are infeasible.

Many heuristic solution methods for vehicle routing problems have been proposed in the literature. Cordeau et al. (2002, 2005) presented a comprehensive computational study on state-of-the-art VRP heuristics. Most VRP methods are using some general (meta-)heuristic strategy to control some kind of a problem-specific neighbourhood search. While for (simple) vehicle routing problems classical local search (LS) based on small moves which alter the current solution to a small degree has shown to be effective, investigations on rich vehicle routing problems indicate that applying local search moves may not converge to near optimal solutions (cf. Derigs and Vogel 2009). The concept of large neighbourhood search (LNS) uses moves, which are destroying the current solution structure significantly, and was successfully applied by Ropke and Pisinger (2006). In this paper, we investigate these options for the VRPC, i.e. we want to identify effective combinations of meta-heuristic control and move strategies. Furthermore, we evaluate a specific and promising algorithmic setup on 23 instances considered by El Fallahi et al. (2008) and Muyldermans and Pang (2007).

The paper is organized as follows: The vehicle routing problem with compartments is introduced in Sect. 2 and a new benchmark suite is provided in Sect. 3. Section 4 presents heuristics for the VRPC, which are studied empirically on the benchmark instances in Sect. 5. Conclusions are given in Sect. 6.

## 2 The vehicle routing problem with compartments

In this section, we describe the VRPC and formulate it as an integer linear program. Furthermore, we discuss special aspects and possible extensions.

### 2.1 Problem formulation

The *vehicle routing problem with compartments* (VRPC) is an abstract problem covering different applications that can occur e.g. in retail and petrol industries. It extends the classical capacitated VRP in several aspects: (i) it considers demands for multiple

inhomogeneous products rather than a homogeneous product, (ii) a vehicle consists of multiple compartments rather than a single one, (iii) all goods delivered on a tour must be assigned to compartments of a vehicle, (iv) certain product pairs must not be loaded together into the same compartment, and (v) certain products must not be loaded into certain compartments.

Formally, the VRPC involves a fixed set of vehicles $V$, a set of compartments $C$, a set of products $P$, a set of locations $L = \{0, \ldots, n\}$, and a set of orders $O$. All vehicles have the same capacity $vehCapa$ ($vehCapa > 0$) and the same set of compartments $C$ with the capacity $compCapa(c)$ of compartment $c \in C$ ($compCapa(c) > 0$). Cost of travelling from location $i$ to $j$ is assumed to be symmetric and denoted as $cost_{ij} \geq 0$. Location 0 is the depot and $L_c = L \setminus \{0\}$ is the set of customer locations. A customer may place several orders, each referring to one single product. For each order $o \in O$, $customer(o) \in L_c$ denotes the customer that placed the order, $product(o) \in P$ the product demanded by the customer, and $quantity(o) > 0$ the demanded quantity. Thus, a customer may receive several deliveries of different orders. We assume without loss of generality that each customer $l \in L_c$ places at least one order, and there is at least one order for each product $p \in P$. Furthermore, the delivery of one order must not be split.

The relation $IncProd \subseteq P \times P$ defines incompatibilities between products, i.e. $(p, q) \in IncProd$ means that products $p$ and $q$ must not be transported together in the same compartment. Incompatibilities between products and compartments are expressed by the relation $IncProdComp \subseteq P \times C$, where $(p, c) \in IncProdComp$ indicates that product $p$ must not be transported in compartment $c$.

Each order has to be assigned to the compartment of a vehicle and all orders assigned to one vehicle have to be sequenced. The resulting order sequence determines the tour $t = (t_1, \ldots, t_r)$, $t_i \in L$, of a vehicle. Each tour $t$ starts and ends at the depot ($t_1 = t_r = 0$), and the cost of the tour is $\sum_{i=1}^{r-1} cost_{t_i t_{i+1}}$.

The VRPC seeks an assignment of orders to vehicles, an assignment of orders to compartments, and a sequence of the assigned orders per vehicle such that all vehicle capacities and compartment capacities are met, the incompatibility relations are satisfied, and the total cost of the determined tours is minimized.

## 2.2 Integer program formulation

The VRPC can be formulated as integer program:

$$\min \sum_{v \in V} \sum_{i \in L} \sum_{j \in L} cost_{ij} \cdot b_{ijv} \tag{1}$$

$$\text{subject to} \quad \sum_{j \in L_c} b_{0jv} \leq 1 \qquad v \in V \tag{2}$$

$$\sum_{i \in L} b_{ikv} = \sum_{j \in L} b_{kjv} \qquad v \in V, k \in L \tag{3}$$

$$u_{iv} - u_{jv} + |L| \cdot b_{ijv} \leq |L_c| \qquad v \in V, i \in L, j \in L_c \tag{4}$$

$$u_{l_0 v} = 1 \qquad v \in V \tag{5}$$

$$\sum_{o \in O} quantity(o) \cdot x_{ovc} \leq compCapa(c) \qquad v \in V, c \in C \tag{6}$$

$$\sum_{o \in O} \sum_{c \in C} quantity(o) \cdot x_{ovc} \leq vehCapa \qquad v \in V \tag{7}$$

$$\sum_{v \in V} \sum_{c \in C} x_{ovc} = 1 \qquad o \in O \tag{8}$$

$$\sum_{o \in ordCust(j)} \sum_{c \in C} x_{ovc} \leq |O| \cdot \sum_{i \in L} b_{ijv} \qquad v \in V, j \in L_c \tag{9}$$

$$\sum_{o \in ordProd(p)} x_{ovc} \leq |O| \cdot y_{pvc} \qquad p \in P, v \in V, c \in C \tag{10}$$

$$y_{pvc} = 0 \qquad (p, c) \in IncProdComp, v \in V \tag{11}$$

$$y_{pvc} + y_{qvc} \leq 1 \qquad (p, q) \in IncProd, v \in V, c \in C \tag{12}$$

$$b_{ijv} \in \{0, 1\} \qquad i, j \in L, v \in V \tag{13}$$

$$u_{iv} \in \{1, \ldots, |L|\} \qquad i \in L, v \in V \tag{14}$$

$$x_{ovc} \in \{0, 1\} \qquad o \in O, v \in V, c \in C \tag{15}$$

$$y_{pvc} \in \{0, 1\} \qquad p \in P, v \in V, c \in C \tag{16}$$

Objective (1) minimizes total transportation cost, where $b_{ijv}$ is a binary variable indicating whether vehicle $v$ travels from location $i$ to $j$.

Constraint (2) ensures that each vehicle $v$ departs at most once from depot 0. (3) imposes that each arrival of vehicle $v$ at location $k$ corresponds to a departure of $v$ from $k$. Together with (2), a tour must always start and end at the depot 0 (if $v$ is used).

Subtour elimination is based on the integer variable $u_{iv}$ that specifies the position of location $i$ in the tour of vehicle $v$. (4) ensures that all tours depart from the depot 0, by imposing that the position of customer $j$ is higher than the position of location $i$, if the vehicle $v$ travels from $i$ to $j$. As there are many potential numberings for the same tour, (5) eliminates duplicates by enforcing the depot 0 to be at position 1.

The binary variable $x_{ovc}$ indicates whether order $o$ is delivered by vehicle $v$ in compartment $c$. (6) states that the goods loaded into compartment $c$ on vehicle $v$ do not exceed the compartment capacity $compCapa(c)$. Analogously, (7) ensures that the vehicle capacity $vehCapa$ is not exceeded by the goods loaded into all compartments of vehicle $v$. (8) ensures that each order is assigned to exactly one compartment on a vehicle. (9) imposes that a vehicle must visit customer $j$ if loads for $j$ are assigned to the vehicle. Here, $ordCust(j) = \{o \in O \mid customer(o) = j\}$ denotes the orders of customer $j$. In (10), the binary variable $y_{pvc}$ indicates whether product $p$ is assigned to compartment $c$ on vehicle $v$. Here, $ordProd(p) = \{o \in O \mid product(o) = p\}$ represents the orders for product $p$.

Using the incompatibility relations $IncProdComp$ and $IncProd$, (11) and (12) model incompatibilities between products and compartments as well as between two products. (13)–(16) define the decision variables, which are all binary except for the integer variable $u_{iv}$.

### 2.3 Discussion

The capacitated VRP is a special case of the VRPC with one product only, one compartment per vehicle that has identical capacity as the vehicle, and no incompatibility constraints. Therefore, the decision version of the VRPC is NP-complete.

Without loss of generality, we assume $vehCapa \leq \sum_{c \in C} compCapa(c)$. The relation between the vehicle capacity $vehCapa$ and the compartments' capacities $compCapa(c)$ indicates whether the compartment setup is configurable. If $vehCapa < \sum_{c \in C} compCapa(c)$, the compartments are flexible, and goods loaded in one compartment may affect the available remaining capacity of other compartments. If $vehCapa = \sum_{c \in C} compCapa(c)$, the compartments are fixed as the capacity of one compartment is independent from the goods loaded in other compartments. In case of fixed compartments, (7) is redundant because of (6).

The problem formulation (1)–(16) is idealized from several perspectives and could easily be extended. If there are weight and volume constraints, demands and capacities could be measured using different dimensions and not only one as in the current formulation. Such an extension would add more equations of type (6) and (7), but no new decision variables would be necessary.

The model assumes that vehicles are homogeneous, i.e. that they have the same total capacity. Also, all compartments are either flexible or fixed and the compartments' capacities are identical for all vehicles. (6) and (7) could be adapted to model vehicles with different capacities and vehicle-dependent compartment capacities.

Currently, we assume enough capacity for all orders. However, in reality, we may be faced with capacity shortage where not all orders can be delivered. In our current formulation, capacity shortage leads to an infeasible integer program. Introducing non-delivery costs into the objective function (1) could model such kind of applications. Then, additional decision variables are necessary to specify which goods are delivered.

Our model assumes that initially the vehicles are empty. In some scenarios, vehicles may have a preload assigned to some compartments, e.g. because a vehicle could not fully perform a previously planned tour. This case can be modelled by preload orders, each being pre-assigned to a compartment of a vehicle.

We only consider tours where each order is fully assigned to one compartment. In some applications, splitting an order and distributing it among different compartments or vehicles could be feasible. The potential of splitting orders was first observed by Dror and Trudeau (1989). State-of-the-art heuristic approaches for solving the split delivery vehicle routing problem were presented by Archetti et al. (2006) as well as Derigs et al. (2009). These extensions would require to define how an order can be split, e.g. continuously into arbitrary smaller parts or according to rules based on lot sizes or other predefined bounds. All these variants would need more decision variables and additional constraints.

In the remaining paper, we consider the problem formulation from Sect. 2.1. We regard this as the core for VRPs involving compartments.

## 3 The benchmark suite for VRPC

Kalkoff (2006) developed an instance generator capable to create VRPC instances according to predefined problem characteristics. Using this generator we created a set of test instances which are available on http://www.ccdss.org/vrp/, intended to serve as a VRPC benchmark suite for the research community. The instances were generated by systematically varying relevant design parameters. Therefore, the benchmark suite is grouped into families of similar problem instances. This allows studying the effects of special problem characteristics on the performance of algorithms as well as the development of problem-specific algorithms. In addition, we can evaluate and compare different optimization algorithms on the complete set of instances.

### 3.1 General problem structure

Table 1 lists the general characteristics of the test instances. The number of customers varies between 10 and 200. The depot is always located at the centre of a two-dimensional square. There are two possibilities on how the customers are placed. In the non-clustered version, the customers are placed randomly on the square. In the clustered version, we randomly place a pre-defined number of smaller squares on the square which resemble clusters. Then, customers are evenly distributed over the clusters and each customer is randomly placed in its square. The total demand of each customer is 1,000. The demand for products follows two alternative product distributions: one with two products and similar demand for the two products and one with three products and significantly different demands. The vehicle capacity is varied from 600 to 9,000 and the number of vehicles is chosen sufficiently large such that all orders can be delivered. In total, there are 100 different parameter combinations represented by Table 1.

### 3.2 Specific problem structure for petrol and food scenarios

Table 2 lists industry-specific characteristics of test instances. In the petrol scenario, each vehicle has five fixed compartments of the same size. Each product is incompatible

**Table 1** General characteristics of test instances

| Parameter | Variations | Values |
|---|---|---|
| # Customers | 5 | 10, 25, 50, 100, 200 |
| Customer distribution | 2 | Clustered, not clustered |
| Customer demand | 1 | 1,000 |
| Product distribution | 2 | (0.53, 0.47), (0.68, 0.24, 0.08) |
| Vehicle capacity | 5 | 600, 800, 1,000, 3,000, 9,000 |
| # Vehicles | 1 | Sufficiently large |

**Table 2** Specific characteristics of petrol and food instances

| Parameter | Petrol | | Food | |
|---|---|---|---|---|
| | Variations | Values | Variations | Values |
| # Compartments | 1 | 5 | 1 | 2 or 3 |
| Compartment capacity/vehicle capacity | 1 | 0.2 | 1 | 1 |
| Max. order demand/compartment capacity | 2 | 1, 0.5 | 2 | 0.5, 0.25 |
| Product incompatibility | 1 | Yes | 1 | No |
| Product-compartment incompatibility | 1 | No | 1 | Yes |

with each other and there are no product-compartment incompatibilities. Thus, each product can be loaded into each compartment, but a compartment can contain only orders of the same product.

In the food scenario, product-compartment incompatibilities exist since each product can be loaded only in one specific compartment. Therefore, the number of compartments equals the number of products. The compartments are flexible, i.e. a compartment's capacity equals the vehicle capacity. Although there are no product incompatibilities, the product-compartment incompatibilities imply that only one product can be loaded into one compartment.

An order's demand may exceed the compartment capacity. For this case, we define a maximum order demand and split the original order into several (smaller) orders. The split is performed by cutting the original order into several orders with maximum order demand and one order for the remaining demand of the original order. For example, assuming compartment capacity of 200 and maximum order demand of 100, an order with quantity 530 would be cut into five orders with quantity 100 and one order with quantity 30. We consider two alternatives for the maximum order demand. For petrol, the maximum order demand is either the same or half of the compartment capacity. For food, it is either one half or a quarter of the compartment capacity.

Given the 100 different general parameter combinations in Table 1, the two petrol scenarios and the two food scenarios listed in Table 2 yield a total of 400 instances, which are the basis for the 200 representative instances which we used to calibrate our solver suite. Some of the generated instances were found to be redundant, due to the fact that in some cases the splitting procedure did not split any order because the compartment capacity was larger than all customers' orders.

## 4 The heuristic solver suite for VRPC

Vehicle routing problems are NP-complete combinatorial optimization problems, and thus only small instances can be solved to optimality and for instances of practical size arising in real-world scenarios, one has to resort to heuristic methods.

Most heuristics for the VRP are search procedures which start from one or several feasible solutions, obtained by so-called construction heuristics, and iteratively improve the solutions by some kind of neighbourhood search where the improvement is controlled by a so-called meta-heuristic. Here, the essence of a good search is the

intelligent combination of intensification and diversification. Diversification allows scanning large parts of the search space, i.e. the set of feasible tours, and intensification is used to scan promising regions more carefully.

Neighbourhood search for the VRP is based on so-called move operators which exchange objects either within a tour (intra-tour moves) or between tours (inter-tour moves). In the classical VRP where every customer receives exactly one delivery, these objects refer to single customers or sequences/sets of customers. In our case, the basic planning entities are the orders and thus we move orders or sequences/sets of orders. Many variants of move operators have been proposed for the different VRP variants; they are all combining rules for the selection of how many and which objects to delete from a tour and where to insert these objects again. When inserting objects, it is necessary to guarantee feasibility of the resulting tours. Checking feasibility of an insertion is the computational bottleneck, since for rich vehicle routing problems quite a number of aspects have to be tested, like vehicle capacity, time windows or load compatibility. Note that for the classical VRP with homogeneous products and without compartments, customers with their entire delivery are moved. The VRPC considers orders of different products for one customer and thus we may move single orders. Instead of "removing/inserting a customer from/into a tour" we have to decide on "removing/inserting an order from/to a compartment of a vehicle" which requires checking load and compatibility constraints and then to decide on the order sequence, i.e. the route that the vehicle should take and that determines the cost.

The algorithmic tree presented in Fig. 1 is adopted from Piesche (2007) and displays in a structured form a survey on the various heuristic approaches/algorithms which we have investigated in our computational study. Although Cordeau et al. (2005) concluded that methods working with one solution at a time are superseded by population-based methods, we did not include population-based methods in our study. Our choice was motivated by our earlier work on other VRP variants (cf. Derigs and Kaiser 2007; Derigs and Reuter 2009; Derigs and Vogel 2009), and we wanted to investigate how the methods and implementations can be extended and how they perform on this richer class of VRP.

The remaining part of this section gives an overview on our algorithmic test-bed. We assume that the reader is familiar with the basic concepts and thus we only give a short description of the various methods with some details which are specific for our problem at hand and our study. The most important references to the underlying literature are given, too. The tree motivates to structure the exposition into subsections corresponding to the components of the tree: local search, large neighbourhood search, meta-heuristics, adaption and construction.

### 4.1 Intensification: local search

Traditionally, for the simple capacitated VRP neighbourhood search based on moves swapping only a very small number of customers has been applied. Such local search (LS) methods for VRP can be partitioned into intra- and inter-tour exchange methods. Among the intra-tour exchange methods, originating from the work on the travelling salesman problem (TSP), the so-called class of $k$-opt methods (cf. Lin and Kernighan

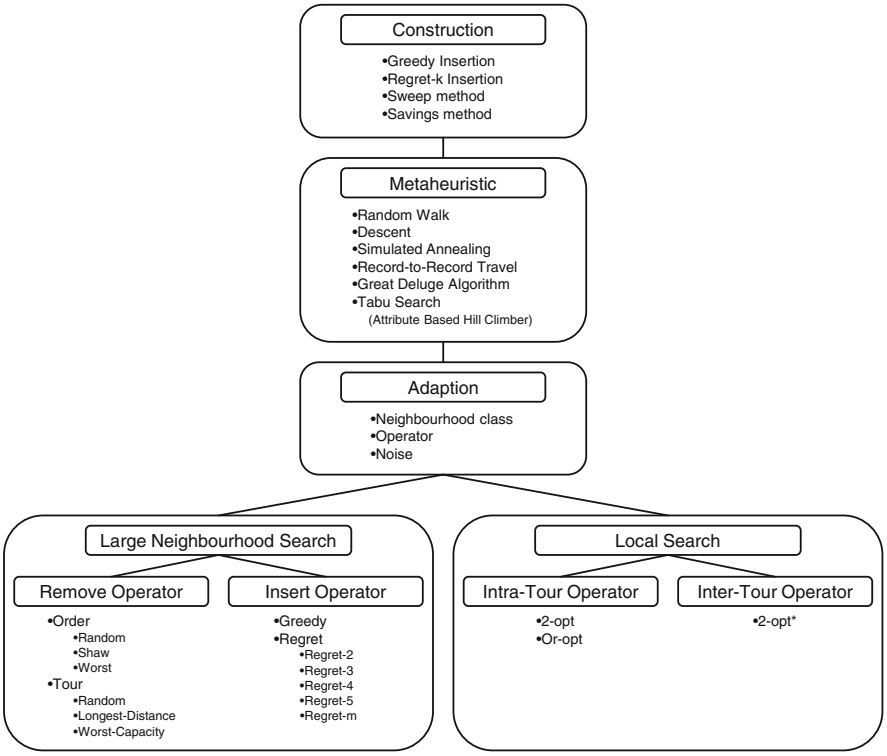


**Fig. 1** Algorithmic tree

1973) is most widely used. Here, for a move a subset of $k$ connections between customers is deleted from the tour, and the resulting segments are tentatively reconnected in all possible ways, and the most profitable reconnection is chosen. Note that only small values like $k \in \{2, 3, 4\}$ are computationally feasible. Or (1976) proposed a restricted variant of 3-opt where strings of sequences of 1, 2 or 3 consecutive customers are replaced reducing the computational effort compared to 3-opt without losing too much quality. For the VRPC, routings are described as sequences of orders and thus connections between orders are exchanged, respecting of course the fact that for intra-tour exchanges deleting a connection between orders of the same customer will never result in an improvement since the customer would have to be visited more than once.

Most inter-tour exchanges are applied to two tours and are based on the relocate/exchange principle. Relocate inserts an order of the first tour at an appropriate position in the second tour. Exchange swaps two orders between the tours. We have implemented 2-opt*, a special inter-tour move introduced by Potvin and Rousseau (1995) where two connections, one from each tour, are removed. This produces four sub-tours which are then recombined. Note that these moves lead to relatively small neighbourhoods and are thus computationally very efficient. Also, for rich vehicle routing problems, the exchange of a larger set of deliveries will most probably violate some constraints.

In our solver suite we have used these neighbourhoods to implement three intensifying heuristics—SD2, SDOr and SD2*—which start from a feasible solution as input and use the steepest descent strategy to construct an associated local optimum.

## 4.2 Diversification: large neighbourhood search

Large neighbourhood search (LNS) is also referred to as ruin-and-recreate strategy (cf. Schrimpf et al. 2000). Here, in order to obtain a "neighbour" the structure of a solution is altered significantly. In the case of VRPC, a significant number $q$ of the orders is first deleted from their respective tours and then re-inserted. An LNS move shows a significantly higher complexity, leads to a larger neighbourhood than local search and re-insertion is more time-consuming and thus efficient implementation is mandatory. Also the brute force testing of entire neighbourhoods for potential improvements—as is done, for instance, in steepest descent search—becomes infeasible. Intelligent criteria for selecting the orders to be removed/inserted are important.

The LNS concept was first proposed by Shaw (1998a,b) for complex VRP. He introduced the "similarity" concept for selecting customers to be deleted and based on this he developed a well-defined VRP procedure, while Pisinger and Ropke (2007) proposed LNS as a generic heuristic based on domain-independent principles for deletion/insertion which have to be specialized for the specific problem type. Quite recently adaptive LNS (ALNS), an extension of the basic LNS concept, was applied rather successfully by Ropke and Pisinger (2006) for solving rich VRP, e.g. the pickup and delivery vehicle routing problem with time windows. In ALNS, different strategies/heuristics for deletion and re-insertion are concurrently used with a learning mechanism guiding the selection of which strategy to apply next. In Bartodziej et al. (2009) we successfully applied ALNS to a combined vehicle and crew scheduling problem with rest constraints.

As mentioned earlier, all remove (ruin) and insert (recreate) operators are highly problem-specific. The basic procedure is always as follows: after performing a remove operator, we obtain an incomplete solution with a set of partial tours which are feasible in the sense that no tour-related constraints like vehicle capacity or product compatibility are violated and a set of orders which are not assigned to a tour/vehicle. These orders are then submitted to an insertion routine. In the following, we describe the set of operators which we have implemented for VRPC.

### *4.2.1 Removal heuristics for VRPC*

We have implemented six different removal heuristics. The first three—Random Removal (R-O), Worst Removal (W-O) and Shaw Removal (S-O)—are adoptions from Ropke and Pisinger (2006), which remove single orders instead of customers as in the case of the VRP with homogeneous goods. Let $q$ be the number of orders to be removed. Then in R-O we simply remove $q$ randomly selected orders from the solution while in W-O we remove $q$ orders which seem to be "misplaced" at their current position. Here, an order is regarded as misplaced if the cost decrease is significantly high when removing it from the solution. S-O is based on the Shaw-concept of

"similarity" and the procedure is simple: one order is selected randomly and removed, and then $q-1$ more orders are removed, with each being similar to at least one of the orders which have been removed before. Here, similarity is defined based on a problem-specific distance measure. For VRPC, we have modified the distance measure of Shaw (1998a) using three components: the distance of customer locations and the difference in quantity and product, respectively.

For the sake of simplicity, let $cost_{ij}$ denote in the following the distance/cost between (the customers of) two orders $i, j \in O$ and $c_{max}$ the maximal distance between any two customers as well as $q_{max}$ the maximal difference of quantities between any two orders. We define

$$p(i,j) := \begin{cases} 1 & \text{if } product(i) \neq product(j) \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

indicating whether two orders $i, j \in O$ have the same product type or not. Using weights $\varphi, \psi, \omega$ the distance measure is given by

$$R(i,j) := \varphi \cdot \frac{cost_{ij}}{c_{max}} + \psi \cdot \frac{|quantity(i) - quantity(j)|}{q_{max}} + \omega \cdot p(i,j) \tag{18}$$

In addition to these variants of removal heuristics for VRP, we have designed and implemented three new heuristics—R-T, L-T and W-T—which alter the solution structure more purposefully and support a different diversification. In these removal heuristics, we select entire tours and then remove their complete set of orders until a preset number of $q$ orders has been removed. In R-T, we select the tours randomly, while in L-T (W-T) we select the tours having longest tour length (smallest load) and thus, those tours which are extreme in a sense. Note that for the last tour selected, only the remaining number of orders to be removed is chosen and thus not all orders of this tour are removed.

#### *4.2.2 Insertion heuristics for VRPC*

We have implemented two insertion strategies, GREEDY and REGRET, which were already proposed by Ropke and Pisinger (2006). In the following, let $I$ be the set of orders to be re-inserted and for $o \in I$ and $v \in V$ let $cost\_increase_o^v$ be the increase in total cost if order $o$ is inserted into the tour of vehicle $v$ at the cheapest position. GREEDY sequentially searches for the pair $(o', v')$, $o' \in I$, $v' \in V$ leading to a minimal cost increase, $cost\_increase_{o'}^{v'} = \min\left\{cost\_increase_o^v \mid o \in I, v \in V\right\}$, and inserts order $o'$ into tour $v'$.

The myopic GREEDY insertion heuristic tends to postpone the insertion of those orders which can only be inserted at a relatively high cost increase. Yet, since in the course of the procedure the number of tours into which an order can be inserted without violating a constraint diminishes, whereas the cost for insertion increases. Thus, a more foresighted criterion should be used to prevent this situation. For $o \in I$, let $v_{ok} \in V, k \in \{1, \ldots, |V|\}$ be the vehicle/tour to which

order $o$ can be assigned at the $k$-th lowest cost increase, i.e. $cost\_increase_o^{v_{ok}} \leq cost\_increase_o^{v_{ok'}} \forall k < k'$. For $k \geq 2$, we define REGRET-$k$ based on an indicator $regret_o^k := \sum_{i=2}^{k} \left(cost\_increase_o^{v_{oi}} - cost\_increase_o^{v_{o1}}\right)$ which measures the disadvantage (regret) of not inserting $o$ in the currently best suited tour but to a less suitable one. Here, large values of $k$ yield a high degree of foresight. Now, in each iteration, we insert the order having the highest regret first. In our implementation we have used REGRET-$k$ for $k \in \{2, 3, 4, 5, m\}$, where $m$ denotes the number of tours.

### 4.3 Adaptive search

To obtain a robust heuristic, Ropke and Pisinger (2006) proposed to alternate between neighbourhoods, using all pairs of removal and insertion operators, but in an adaptive way. Weights are assigned to the different operators and a kind of roulette wheel selection is used, i.e. if we have $k$ neighbourhoods with weights $w_i$ , $i \in \{1, \ldots, k\}$, then we select the neighbourhood $j$ with probability

$$p_j := \frac{w_j}{\sum_{i=1}^{k} w_i} \tag{19}$$

The weights are automatically adjusted based on a score-value for each neighbourhood which is calculated using the information on their performance in earlier iterations. To enhance diversification, Ropke and Pisinger (2006) proposed to disturb the exact insertion cost by adding a randomly generated noise term and to let the adaptive mechanism decide whether to use the real cost or the disturbed cost for determining the next insertion. We have modified their approach in only one aspect that also the local search moves are used concurrently with the LNS operators and here the weights decide whether a local search operator or an LNS operator is used. For details on the exact procedure of score-updating and generating noise we refer to Ropke and Pisinger (2006).

The pseudocode for our search heuristic is shown in Algorithm 1. It abstracts from the initial solution and the meta-heuristic control used. Also, note that for sake of simplicity we write $cost(S)$ for the cost of a feasible solution $S$.

### 4.4 Construction heuristics

LS and LNS are improvement methods which can only be started from a given initial feasible solution. The purpose of a construction heuristic is to generate a feasible solution of acceptable quality in rather short computation time. A widely used principle is the greedy principle where based on some problem-specific myopic decision rule objects are sequentially inserted into partial solutions. Two old and prominent greedy methods which have been proposed for the VRP and which we have used in our study too are the Savings method (Clarke and Wright 1964) and the Sweep method (Gillett and Miller 1974).

In the Savings method, deadhead tours are built for every order and the savings are calculated which may result when combining two deadhead tours to a larger tour.

**Algorithm 1** Neighbourhood search schema

```
1: function NeighbourhoodSearch(initial solution S, meta-heuristic control H)
2:     S_best := S
3:     neighbourhood classes 𝒩 := {LS, LNS}
4:     steepest descent moves 𝒮𝒟 := {SD2, SDOr, SD2*}
5:     removal heuristics ℛ := {R-O,W-O,S-O,R-T,L-T,W-T}
6:     insertion heuristics ℐ := {GREEDY,REGRET-k for k ∈ {2,3,4,5,m}}
7:     repeat
8:         S' := S
9:         select neighbourhood class N ∈ 𝒩
10:        if N = LS then
11:            select operator SD ∈ 𝒮𝒟
12:            apply SD to S' until S' is a local optimum
13:        else
14:            select subheuristics R ∈ ℛ and I ∈ ℐ
15:            choose q, the number of orders to be removed
16:            remove q orders from S' using R
17:            re-insert removed orders into S' using I
18:        end if
19:        if cost(S') < cost(S_best) then
20:            S_best := S'
21:        end if
22:        if AcceptNeighbour(S', S, H) then
23:            S := S'
24:        end if
25:    until StopCriterion
26:    return S_best
27: end function
```

Then starting with the highest saving and working down the ordered savings list, tours are combined by connecting last and first orders if possible. And here feasibility constraints have to be checked. Except for this eventually complex and time-consuming check this greedy method is rather fast.

The Sweep method is applicable for planar problems only where the locations for customers are given by polar coordinates. In the classical VRP a customer location is assigned 0-angle, and the angles of the other customer locations are computed from this. Then customers are ranked in increasing order of their polar coordinate and starting from the unrouted customer having the smallest angle deliveries are sequentially assigned to the first vehicle as long as constraints on capacity are fulfilled. Otherwise, the current vehicle is closed and the process is repeated with a new vehicle and so on. Finally, the customer deliveries in each vehicle cluster are routed by solving a corresponding TSP. For VRPC, we have modified this procedure. All orders are ordered based on their customer's polar angle and sweeps are started from the first unassigned order. Now, in the case that the next order cannot be assigned to the current vehicle due to a violation of a constraint, we do not close the vehicle but we go on trying the following orders until we have checked all unassigned orders or the remaining vehicle capacity becomes smaller than the smallest order. In that case, we close the vehicle and repeat from the first unassigned order. Obviously, the quality of the solution

depends on the customer defining the 0-angle and thus the procedure should be repeated from different especially selected customers. In our approach for solving VRPC, we have implemented and tested several variations of the Sweep method (cf. Sect. 5). An alternative to using the Savings or Sweep method is to use one of the LNS insertion heuristics, GREEDY or REGRET, to insert all orders into the "empty solution".

### 4.5 Meta-heuristics

LS and LNS are heuristic principles to solve hard (combinatorial) optimization problems based on the specification of problem-specific neighbourhood structures. Starting from an initial solution $S_0$, a sequence $S_0, S_1, S_2, \ldots$ of solutions is created, where $S_i$ is contained in the neighbourhood of $S_{i-1}$, $i = 1, 2, \ldots$, and the selection of the solution $S_i \in N(S_{i-1})$ is guided by a meta-heuristic, i.e. some (heuristic) strategy which governs the search process.

The conceptually most simple approach for guiding LS is called Steepest Descent (SD) where one always selects and accepts the most improving neighbour until no further improvement is possible, and a solution is obtained which is locally optimal with respect to the neighbourhood. This simple heuristic has the tendency to get trapped in a "bad" local optimum. In our computational study, we have investigated classical meta-heuristics which overcome this problem. For the first three heuristics neighbours are generated randomly. In Simulated Annealing (SA) (Kirkpatrick et al. 1983) a solution $S_i \in N(S_{i-1})$ is always accepted if $cost(S_i) \leq cost(S_{i-1})$ and accepted with probability $e^{-\frac{cost(S_i)-cost(S_{i-1})}{T}}$ if $cost(S_i) > cost(S_{i-1})$. Here, $T$ is the so-called *temperature* which is initialized at $T > 0$ and is decreased every $L$ iterations by performing the so-called *geometric cooling update* $T := \alpha \cdot T$ with $0 < \alpha < 1$ such that finally the procedure converges to a simple iterative Descent Algorithm. Record-to-Record Travel (RRT) (Dueck 1993) accepts a solution if it does not exceed the best solution found so far during the search by a specified $deviation$ value, and the Great Deluge Algorithm (GDA) (Dueck 1993) accepts a solution if its objective function value is below a so-called $waterlevel$ value which is decreased by a so-called $dryspeed$ value each time a solution is accepted.

On the other hand, Tabu Search (TS) (Glover 1989, 1990) always moves to the best neighbour even if this is not leading to an improvement, and, to avoid cycling, solutions which have already been visited, are forbidden. For this purpose, solutions possessing some attributes of recently explored solutions are temporarily declared tabu, unless their cost is less than a so-called aspiration level.

With respect to implementation, all of these meta-heuristics leave the user with several degrees of freedom, i.e. the control of the algorithmic process is specified through several parameter settings. In SA, we have to choose the initial probability and the schedule of lowering this probability over time, the so-called cooling schedule, and in TS we have to specify the length of the tabu-list, which controls the number of iterations that a solution stays tabu. Whittley and Smith (2004) proposed the Attribute Based Hill Climber (ABHC) heuristic as a parameter-free variant of TS which was successfully applied to the simple VRP by Derigs and Kaiser (2007) and showed to be compatible in quality and performance with the state-of-the-art VRP heuristics.

ABHC uses a special concept for specifying non-tabu neighbours which is generic and which needs to be specialized for every problem domain. Here, certain characteristic potential properties of solutions, the so-called attributes, have to be specified and a solution is acceptable if it is the best solution found so far for at least one attribute. Hence, ABHC is parameter-free and except for some tie-breaking completely deterministic.

## 5 Empirical results

In this section, we present the essence of a comprehensive computational study extending the work described by Piesche (2007) in which we have tested numerous variants of combinations of the subheuristics within the different components from the solver suite introduced in Sect. 4 on the instances from the petrol/food benchmark suite. Then we have applied one specific parameterization which has shown to be rather effective on a set of benchmark problems which were published by El Fallahi et al. (2008) and Muyldermans and Pang (2007). All our programs have been written in C# and the computational results were obtained on 3 GHz PCs with 2GB memory running under Windows XP.

### 5.1 Computational tests on the VRPC benchmark suite

It is evident that due to the large number of heuristic modules which have been implemented an enormous number of potential combinations exists, and it is beyond any realistic expectation to test all of these. Therefore, we have designed a test procedure on our set of 200 benchmark problems which concentrates on identifying appropriate and recommendable choices within each component—construction, search neighbourhoods/moves and meta-heuristic control—by applying the ceteris paribus principle, i.e. we compared different combinations of neighbourhoods using the same meta-heuristic control which had shown to be best when applying the entire set of neighbourhood moves etc. Our findings can be summarized as follows:

- no significant difference in computational behaviour between the two classes of instances—petrol industry and food distribution—could be observed which may be due to the fact that our implementation is oriented towards solving the general VRPC and did not consider the specifics of the two domains explicitly,
- yet, when neglecting the compartment constraints, i.e. solving the instances as standard VRP, it turned out that the difference between the heuristically determined VRP and VRPC solution values is rather small for the examples with flexible compartments but specific compartments per product type (food scenario), while this difference is significant for the examples with fixed compartment sizes but no incompatibilities between products and compartments (petrol scenario),
- the entire set of diversification and intensification concepts/heuristics is essential to obtain the best performance with respect to quality,

- the application of the adaption strategy (as well as noise) is not of significant importance on the instance level, i.e. we could identify parameter configurations which worked well on the entire set of instances,
- an intelligent combination of Sweep and Savings has shown to give the best start solutions, yet, the choice of a proper combination of construction/start heuristics is more crucial with respect to running time than solution quality,
- the choice of the meta-heuristic control is of minor influence with Record-to-Record Travel being the best choice if also complexity of implementation is considered.

We will now describe the experimental design and present several tables which support these statements. We only give aggregated information on relative deviation to the best solutions which we could find during our study. The best solutions found are published together with the benchmark data.

### *5.1.1 Comparison of operators*

In the following, we first summarize our findings on the use of neighbourhoods/move operators. For that purpose, we have compared different combinations using the same meta-heuristic control which has shown to be rather effective: RRT with a $deviation$ of 0.009; thus we accept a neighbour solution if it does not exceed the best-known solution found so far by more than 0.9%. For the LNS-removal operators we choose $q$, the number of orders to be removed randomly between four and one-third of the number of orders. For the distance measure we have set $\varphi = \psi = 100$ and $\omega = 50$.

The quality is measured systematically by giving for each combination the average deviation from the best solutions found over all 200 instances. The implementation with the entire set of operators is used as reference or basic configuration. In a table, every column corresponds to an algorithmic concept, and every row represents a specific parameterization where a dot indicates that the associated concept has been applied.

Table 3 shows how starting from the reference implementation the solution quality changes if exactly one algorithmic concept is deleted from the basic configuration. Except for the version where SD2* has been eliminated, all other combinations perform similar. This result shows that except for SD2*, basically every concept can be compensated. The next test was performed to identify the impact of a single operator within its specific component class. For that purpose, all other operators within the class were eliminated. From Table 4 we can see that especially for the LNS removal class, different operators representing different heuristic principles are necessary, and for LNS insertion at least one REGRET-operator is necessary. Within the class of intensification operators, SD2* turns out to be essential, which could be expected from the previous result.

Table 5 shows the results for some characteristic configurations. First, comparing configurations C1 and C2 it is remarkable that when using all move operators the use of the adaptive strategy and introducing noise does improve the solution quality only marginal. Yet, due to the fact that these instruments are computationally inexpensive we recommend their use. Comparing configurations C3 and C4, one can see that pure local search is ineffective for VRPC while with pure LNS still acceptable results are

**Table 3** Sensitivity analysis for individual heuristics

| Configuration | S-O | W-O | R-O | R-T | L-T | W-T | Greedy | Reg-2 | Reg-3 | Reg-4 | Reg-5 | Reg-m | SD2 | SDOr | SD2* | Noise | Adaptivity | Deviation (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S1 | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | 1.5 |
| S2 | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | 1.7 |
| S3 | • | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | 1.8 |
| S4 | • | • | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | 1.6 |
| S5 | • | • | • | | • | • | • | • | • | • | • | • | • | • | • | • | • | 1.5 |
| S6 | • | • | • | • | | • | • | • | • | • | • | • | • | • | • | • | • | 1.5 |
| S7 | • | • | • | • | • | | • | • | • | • | • | • | • | • | • | • | • | 1.7 |
| S8 | • | • | • | • | • | • | | • | • | • | • | • | • | • | • | • | • | 1.5 |
| S9 | • | • | • | • | • | • | • | | • | • | • | • | • | • | • | • | • | 1.6 |
| S10 | • | • | • | • | • | • | • | • | | • | • | • | • | • | • | • | • | 1.5 |
| S11 | • | • | • | • | • | • | • | • | • | | • | • | • | • | • | • | • | **1.4** |
| S12 | • | • | • | • | • | • | • | • | • | • | | • | • | • | • | • | • | 1.5 |
| S13 | • | • | • | • | • | • | • | • | • | • | • | | • | • | • | • | • | 1.5 |
| S14 | • | • | • | • | • | • | • | • | • | • | • | • | | • | • | • | • | 1.5 |
| S15 | • | • | • | • | • | • | • | • | • | • | • | • | • | | • | • | • | 1.5 |
| S16 | • | • | • | • | • | • | • | • | • | • | • | • | • | • | | • | • | 2.1 |
| S17 | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | | • | 1.7 |
| S18 | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | | 1.6 |

**Table 4** Impact analysis for individual heuristics

| Configuration | S-O | W-O | R-O | R-T | L-T | W-T | Greedy | Reg-2 | Reg-3 | Reg-4 | Reg-5 | Reg-m | SD2 | SDOr | SD2* | Noise | Adaptivity | Deviation (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I1 | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | **1.5** |
| I2 | • | | | | | | • | • | • | • | • | • | • | • | • | • | • | 1.9 |
| I3 | | • | | | | | • | • | • | • | • | • | • | • | • | • | • | 2.2 |
| I4 | | | • | | | | • | • | • | • | • | • | • | • | • | • | • | 2.2 |
| I5 | | | | • | | | • | • | • | • | • | • | • | • | • | • | • | 3.3 |
| I6 | | | | | • | | • | • | • | • | • | • | • | • | • | • | • | 5.4 |
| I7 | | | | | | • | • | • | • | • | • | • | • | • | • | • | • | 4.1 |
| I8 | • | • | • | • | • | • | • | | | | | | • | • | • | • | • | 2.8 |
| I9 | • | • | • | • | • | • | | • | | | | | • | • | • | • | • | 1.8 |
| I10 | • | • | • | • | • | • | | | • | | | | • | • | • | • | • | 1.7 |
| I11 | • | • | • | • | • | • | | | | • | | | • | • | • | • | • | 1.6 |
| I12 | • | • | • | • | • | • | | | | | • | | • | • | • | • | • | 1.7 |
| I13 | • | • | • | • | • | • | | | | | | • | • | • | • | • | • | 1.8 |
| I14 | • | • | • | • | • | • | • | • | • | • | • | • | • | | | • | • | 2.2 |
| I15 | • | • | • | • | • | • | • | • | • | • | • | • | | • | | • | • | 2.2 |
| I16 | • | • | • | • | • | • | • | • | • | • | • | • | | | • | • | • | 1.6 |

**Table 5** Analysis of characteristic configurations

| Configuration | S-O | W-O | R-O | R-T | L-T | W-T | Greedy | Reg-2 | Reg-3 | Reg-4 | Reg-5 | Reg-m | SD2 | SDOr | SD2* | Noise | Adaptivity | Deviation (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C1 | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | 1.5 |
| C2 | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | | | 1.7 |
| C3 | | | | | | | | | | | | | • | • | • | | | 7.3 |
| C4 | • | • | • | • | • | • | • | • | • | • | • | • | | | | • | • | 2.2 |
| C5 | | | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | 3.0 |
| C6 | • | • | • | | | | • | • | • | • | • | • | • | • | • | • | • | 1.7 |
| CR | • | • | • | | | • | | • | • | | • | • | • | | • | • | • | **1.4** |

produced. Comparing configurations C5 and C6, we can conclude that among the LNS removal operators those removing single orders are essential while the operators which remove complete tours should only be regarded as add-ons. Based on these findings we could identify several configurations which are almost equally good for all instances and which are obtained from C1 by omitting a small number of operators only. Configuration CR seems to be the best choice and is recommended. Here, we retain only one vehicle-based removal operator (W-T) and we omit GREEDY insertion and REGRET-4 as well as SDOr. Thus, the number of operators has been reduced achieving the solution quality of the reference implementation C1.

#### *5.1.2 Comparison of meta-heuristics*

This comparison was performed using the reference configuration, i.e. applying all move operators as well as the adaptive strategy and noise. First, we have performed individual tests to determine good parameterizations for every single meta-heuristic. We just give the result of these tests:

For Simulated Annealing the best cooling factor was $\alpha = 0.9999$ and $L$ was set to 1. For Record-to-Record Travel, a $deviation$ of 0.009 has shown to give good results while for Great Deluge the $dryspeed$ was set to 0.009. For ABHC, we used the connections $(i, j)$ between two customers as attributes, a choice which has shown to be effective for the simple VRP; thus, for VRPC a solution has attribute $(i, j)$ if a tour exists with two consecutive deliveries of orders with $i$ being the customer of the first and $j$ being the customer of the second delivery. In combination with LNS, ABHC turned out to be computationally infeasible since it requires scanning entire neighbourhoods. Therefore, we have implemented a variant ABHC-100 which works as follows: for the current solution, we test 100 remove/insert—moves and then we move to the best acceptable neighbour.

We have run the meta-heuristics on every instance for a fixed time of 10 CPU minutes. As reference implementation for the meta-heuristic competition, we have also implemented the simple random walk control (RW) that accepts every solution and the simple descent control (DC), which accepts every improving solution. In Fig. 2, we display the results giving for each meta-heuristic the average and the maximal deviation. As one can see, the differences among the meta-heuristics are not large with RRT performing best.

#### *5.1.3 Comparison of construction heuristics*

Our computational tests have shown that Sweep as well as Savings outperform each other significantly on certain sets of instances and that robust results can only be obtained by using the best solution from Savings and a specific Sweep variant. Obviously, the solution obtained by Sweep is dependent on the choice of the 0-angle customer. Better results can be obtained by Sweep-All which is starting Sweep from every order and selecting the best solution obtained, but this turned out to be too costly for our benchmark instances. Therefore, we have implemented a rule called "big Gap" for selecting the first customer: Choose the order from the sorted list which has the largest angle with its predecessor in the list. Note that we use Opt2 and OrOpt in a

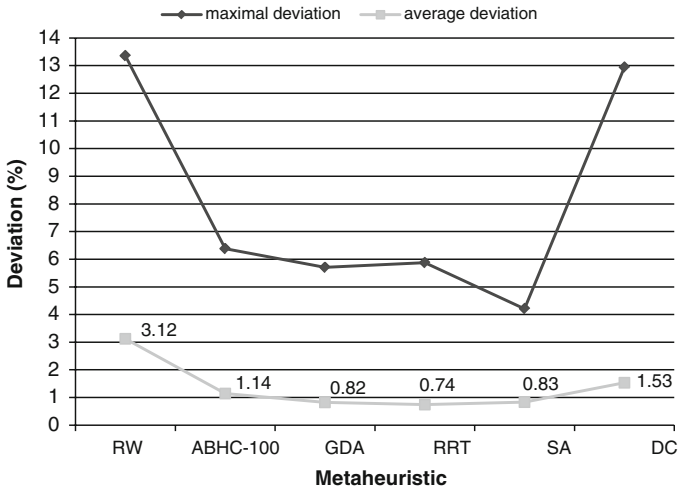


**Fig. 2** Comparison of meta-heuristics

**Table 6** Comparison of construction heuristics

| | Savings | Sweep-All | Sweep big Gap | Savings + | |
|---|---|---|---|---|---|
| | | | | Sweep-All | Sweep big Gap |
| Average deviation (%) | 11.9 | 10.9 | 12.9 | 7.7 | 8.6 |
| Maximal deviation (%) | 31.4 | 36.2 | 55.8 | 27.3 | 29.5 |
| Minimal deviation (%) | 0 | 0 | 0 | 0 | 0 |
| Average CPU-time (min:sec) | 00:00.80 | 00:15.90 | 00:00.20 | 00:16.60 | 00:01.00 |
| Maximal CPU-time (min:sec) | 00:12.80 | 03:00.70 | 00:03.20 | 03:20.80 | 00:15.20 |

postprocessing to optimize the routings in the resulting clusters. The results are shown in Table 6. Applying LNS insertion for constructing initial solutions did not pay off since the running time was comparable to the running time of Sweep-All, yet the quality was inferior to Savings + big Gap Sweep.

### *5.1.4 Impact of compartment constraints*

In a further analysis, we have studied to which extent the compartment constraints affect the tour length of the solutions. For each instance of the test set, we have created the associated standard VRP instance by considering every order as an individual customer and keeping the original vehicle capacity but neglecting different product types, compartments and compatibility constraints. We have solved these VRP instances using the ABHC heuristic from Derigs and Kaiser (2007)—with a large number of iterations per instance (400,000) to obtain high-quality VRP solutions—and we have then compared the results with the best solutions we have found for the original VRPC instances. Table 7 shows the average and maximum percentage by which the VRPC

**Table 7** Deviations of VRPC solutions from corresponding VRP solutions

| Scenario | $r$ | Average deviation (%) | Maximum deviation (%) |
|---|---|---|---|
| Food | 0.6 | 0.47 | 1.82 |
| Food | 0.8 | 0.39 | 1.71 |
| Food | 3 | 0.01 | 0.07 |
| Food | 9 | −0.44 | −1.98 |
| Petrol | 0.6 | 2.72 | 8.32 |
| Petrol | 0.8 | 4.42 | 10.54 |
| Petrol | 1 | 20.19 | 44.40 |
| Petrol | 3 | 16.74 | 24.61 |
| Petrol | 9 | 20.64 | 30.86 |

tour lengths exceed the VRP tour lengths. Here, we have analysed the influence of two problem characteristics: we distinguish the food and the petrol scenario, and the ratio $r$ between vehicle capacity and single customer demand (cf. Table 1). Obviously, the deviations for the food instances are close to 0% since in the case of flexible compartments the VRPC is equivalent to the standard VRP unless certain constellations of compatibility constraints apply. Note, that for the food case all instances with $r = 1$ turned out to be trivially solvable and have been discarded from our sample. For the petrol scenario, we observed notably high deviations with a peak for $r = 1$. Here, for the VRP each customer can be completely served by a deadhead tour, while for the VRPC it must be served by multiple tours which results in an increase in total tour length.

### 5.2 Results on the instances from the literature

Recently, two studies on a special case of the VRPC were published, which state computational results on reproducible instances. Here, the well-known instances for the VRP from Christofides et al. (1979) and Eilon et al. (1971) were transformed into sets of VRPC instances with two compartments with fixed capacities and two products by splitting the request of each customer into two equal amounts, one per product. Each product is compatible with only one compartment. Thus, these instances belong to a specific and combinatorially less complex class of our food scenario. Due to this construction mechanism, a solution for the original VRP is feasible for the associated VRPC. El Fallahi et al. (2008) transformed 20 instances with seven of these instances including a route length constraint which could not be solved by our present implementations. Muyldermans and Pang (2007) transformed a set of 19 instances, from which nine were also considered by El Fallahi et al. (2008). For comparison, we run our implementation using the configuration CR where we have tuned the $deviation$-parameter of RRT only. For this class of instances $deviation = 0.03$ showed to be a recommendable choice. The results are summarized in Fig. 3 and Table 8. We adopt the identifiers from Toth and Vigo (2002): the first three-digit integer gives the number of clients including the depot and the second two-digit integer gives the number of

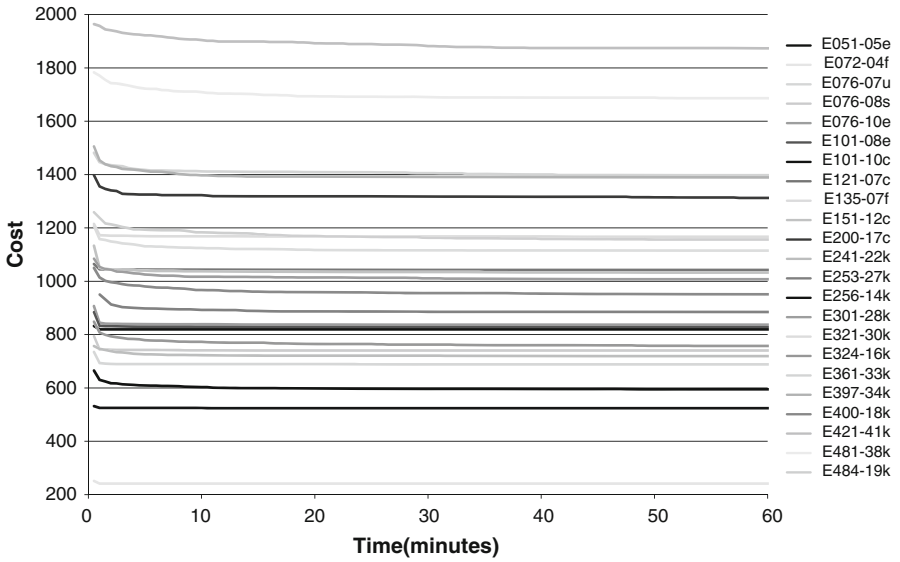


**Fig. 3** Convergence with increasing running time

vehicles needed. We use the (best known) solution values for the original VRP as reported by Derigs and Kaiser (2007) as reference values.

Figure 3 shows the convergence of our implementation with increasing running time. As can be seen, after about 20 min of CPU-time the solution quality does not improve significantly.

Table 8 states the results reported by El Fallahi et al. (2008) for their memetic algorithm and their Tabu Search, the results reported by Muyldermans and Pang (2007) for different numbers of iterations of their guided local search implementation as well as the results of our implementation for four different running times (5, 10, 20, and 60 min). Note that the running times reported by El Fallahi et al. (2008) and Muyldermans and Pang (2007) are in seconds based on a PC Pentium 4 with 2.4 GHz and a PC Pentium M740 with 1.73 GHz, respectively. After 5 min, we reach a better solution quality than El Fallahi et al. (2008) and worse quality than the 1200k-version of Muyldermans and Pang (2007) on 16 out of 19 instances. After 10 min of CPU-time, our implementation yields comparable solution quality as Muyldermans and Pang (2007). After 20 min of CPU-time, we achieve better solution quality on 11 out of 19 instances considered by Muyldermans and Pang (2007). After 60 min of CPU-time, our implementation yields better or equal solution values on all but one instance.

Thus our algorithm has shown to be competitive and is able to produce better solutions for longer running times on the instances considered by El Fallahi et al. (2008) and Muyldermans and Pang (2007). Besides speed and accuracy Cordeau et al. (2002) identified flexibility as another essential attribute for a heuristic. Here, flexibility refers to the potential of a heuristic to accommodate the various side constraints encountered in real-life applications. Note that our algorithm has been designed for a more general

**Table 8** Comparison with El Fallahi et al. (2008) and Muyldermans and Pang (2007)

| Instance | VRP | El Fallahi et al. (2008) | | | | Muyldermans and Pang (2007) | | | | | | Derigs et al. | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MA | | TS | | GLS (300k) | | GLS (600k) | | GLS (1200k) | | 5 min | 10 min | 20 min | 60 min |
| | Cost | Cost | Time | Cost | Time | Cost | Time | Cost | Time | Cost | Time | Cost | Cost | Cost | Cost |
| E051-05e | 524.61 | 524.60 | 23.7 | 524.60 | 15.8 | 524.61 | 123.6 | 524.61 | 247.2 | 524.61 | 538.2 | 525.83 | 525.83 | 524.61 | 524.61 |
| E076-10e | 835.26 | 842.70 | 55.1 | 851.80 | 21.5 | 842.93 | 127.8 | 837.40 | 253.8 | 837.40 | 516.6 | 839.70 | 838.84 | 837.85 | 837.07 |
| E101-08e | 826.14 | 853.20 | 62.3 | 835.20 | 95.3 | 829.84 | 139.8 | 829.84 | 279.0 | 829.84 | 564.6 | 830.18 | 829.38 | 829.02 | 827.49 |
| E151-12c | 1,028.42 | 1,070.90 | 207.1 | 1,055.10 | 411.2 | 1,040.18 | 142.8 | 1,040.18 | 295.8 | 1,040.18 | 606.6 | 1,040.40 | 1,037.50 | 1,035.55 | 1,032.65 |
| E200-17c | 1,291.29 | 1,330.30 | 410.3 | 1,348.80 | 809.2 | 1,330.06 | 150.6 | 1,318.38 | 297.6 | 1,313.96 | 612.0 | 1,325.55 | 1,323.10 | 1,318.71 | 1,312.82 |
| E121-07c | 1,042.11 | 1,044.65 | 126.5 | 1,043.80 | 64.9 | 1,048.37 | 125.4 | 1,048.37 | 242.4 | 1,048.67 | 499.8 | 1,043.57 | 1,043.30 | 1,042.85 | 1,042.34 |
| E101-10c | 819.56 | 819.60 | 76.7 | 822.00 | 34.9 | 819.56 | 129.6 | 819.56 | 272.4 | 819.56 | 558.6 | 819.56 | 819.56 | 819.56 | 819.56 |
| E241-22k | 707.79 | 743.80 | 306.4 | 741.20 | 805.8 | 723.54 | 159.6 | 720.09 | 318.6 | 719.71 | 658.2 | 726.52 | 723.26 | 721.54 | 719.86 |
| E484-19k | 1,107.19 | 1,204.10 | 2,055.9 | 1,148.90 | 6,459.7 | 1,202.67 | 274.8 | 1,181.66 | 538.2 | 1,173.37 | 1,087.8 | 1,193.67 | 1,183.96 | 1,170.14 | 1,155.93 |
| Avg (dev %) | | 2.82 | | 2.12 | | 1.88 | | 1.45 | | 1.32 | | 1.74 | 1.51 | 1.24 | 0.95 |
| Avg (s) | | | 369.33 | | 968.70 | | 152.67 | | 305.00 | | 626.93 | 300 | 600 | 1,200 | 3,600 |
| E072-04f | 237.00 | 244.80 | 11.4 | 241.90 | 28.0 | | | | | | | 241.97 | 241.97 | 241.97 | 241.97 |
| E076-08s | 735.00 | 749.50 | 32.6 | 747.90 | 22.6 | | | | | | | 742.76 | 741.46 | 740.66 | 740.66 |
| E076-07u | 682.00 | 690.80 | 27.9 | 690.20 | 29.5 | | | | | | | 689.34 | 689.34 | 689.34 | 688.47 |
| E135-07f | 1,162.00 | 1,165.60 | 160.2 | 1,179.21 | 140.7 | | | | | | | 1,168.98 | 1,168.82 | 1,167.88 | 1,166.90 |
| Avg (dev %) | | 1.72 | | 1.63 | | | | | | | | 1.21 | 1.16 | 1.11 | 1.06 |
| Avg (s) | | | 58.03 | | 55.20 | | | | | | | 300 | 600 | 1,200 | 3,600 |
| E253-27k | 859.11 | | | | | 894.49 | 146.4 | 893.61 | 295.2 | 885.03 | 605.4 | 893.07 | 887.20 | 885.59 | 881.50 |
| E256-14k | 583.39 | | | | | 614.28 | 199.8 | 609.85 | 386.4 | 605.65 | 783.6 | 609.58 | 603.53 | 598.19 | 594.83 |
| E301-28k | 998.73 | | | | | 1,044.25 | 175.2 | 1,041.95 | 337.8 | 1,036.22 | 702.0 | 1,025.84 | 1,017.32 | 1,014.78 | 1,007.91 |

**Table 8** continued

| Instance | VRP | El Fallahi et al. (2008) | | | | Muyldermans and Pang (2007) | | | | | | Derigs et al. | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | MA | | TS | | GLS (300k) | | GLS (600k) | | GLS (1200k) | | 5 min | 10 min | 20 min | 60 min |
| | Cost | Cost | Time | Cost | Time | Cost | Time | Cost | Time | Cost | Time | Cost | Cost | Cost | Cost |
| E321-30k | 1,081.31 | | | | | 1,135.24 | 152.4 | 1,134.19 | 303.6 | 1,125.75 | 619.2 | 1,131.31 | 1,124.88 | 1,118.21 | 1,115.21 |
| E324-16k | 742.03 | | | | | 778.48 | 214.8 | 770.98 | 414.6 | 770.98 | 848.4 | 779.42 | 773.19 | 765.32 | 757.41 |
| E361-33k | 1,366.86 | | | | | 1,410.56 | 181.2 | 1,410.13 | 361.8 | 1,404.58 | 738.0 | 1,417.41 | 1,411.92 | 1,409.38 | 1,397.19 |
| E397-34k | 1,345.23 | | | | | 1,405.56 | 162.6 | 1,405.56 | 317.4 | 1,405.56 | 646.8 | 1,413.82 | 1,397.53 | 1,392.35 | 1,389.30 |
| E400-18k | 918.45 | | | | | 967.55 | 237.0 | 965.57 | 459.6 | 965.56 | 940.2 | 980.91 | 967.43 | 959.51 | 950.99 |
| E421-41k | 1,821.15 | | | | | 1,900.11 | 199.8 | 1,897.76 | 397.2 | 1,897.76 | 808.8 | 1,922.49 | 1,904.67 | 1,894.42 | 1,872.02 |
| E481-38k | 1,622.69 | | | | | 1,700.18 | 172.2 | 1,695.27 | 339.6 | 1,691.45 | 681.0 | 1,722.43 | 1,709.22 | 1,693.94 | 1,685.70 |
| Avg (dev %) | | | | | | 4.60 | | 4.31 | | 3.94 | | 4.81 | 3.92 | 3.33 | 2.64 |
| Avg (s) | | | | | | | 184.14 | | 361.32 | | 737.34 | 300 | 600 | 1,200 | 3,600 |

**Table 9** Results for original VRP instances

| Instance | VRP | Derigs et al. | | |
|---|---|---|---|---|
| | | 10 min | 20 min | 60 min |
| | Cost | Cost | Cost | Cost |
| E051-05e | 524.61 | 531.90 | 531.90 | 524.61 |
| E076-10e | 835.26 | 838.17 | 838.17 | 835.26 |
| E101-08e | 826.14 | 830.94 | 829.25 | 826.14 |
| E151-12c | 1,028.42 | 1,041.99 | 1,041.99 | 1,033.54 |
| E200-17c | 1,291.29 | 1,331.38 | 1,320.24 | 1,309.42 |
| E121-07c | 1,042.11 | 1,042.81 | 1,042.12 | 1,042.12 |
| E101-10c | 819.56 | 819.56 | 819.56 | 819.56 |
| E241-22k | 707.79 | 730.28 | 729.31 | 728.15 |
| E484-19k | 1,107.19 | 1,182.41 | 1,176.64 | 1,173.32 |
| E072-04f | 237.00 | 241.97 | 241.97 | 241.97 |
| E076-08s | 735.00 | 740.66 | 740.66 | 740.66 |
| E076-07u | 682.00 | 688.47 | 687.60 | 687.60 |
| E135-07f | 1,162.00 | 1,167.31 | 1,166.84 | 1,166.56 |
| Avg (dev %) | | 1.62 | 1.47 | 1.14 |

class of VRPC-instances, i.e. the number of compartments and products is not limited to two; we allow variable and fixed compartment capacities as well as complex incompatibility constraints.

Finally, we have analysed the behaviour of our VRPC implementation for solving standard VRP instances, i.e. we have interpreted VRP data sets as VRPC instances with one product type and one compartment which is a valid input for our implementation. Table 9 shows that we are able to produce solutions with a deviation from the best-known VRP solutions around 1% after only 10 min of running time. This result does not only document that our implementation performs well on VRP instances, but it also supports the hypothesis that adopting and specializing methods like (A)LNS with move-operators which have been shown to be successful on VRP to richer VRP classes by proper implementation of feasibility checks for the additional constraints is a promising approach.

## 6 Conclusions

We have motivated the vehicle routing problem with compartments (VRPC) and presented a problem statement covering scenarios from different industries, namely distribution of petrol and food. We have provided an integer program formulation and discussed its extensibility towards further scenario enrichments.

A benchmark suite of 200 VRPC instances has been created and is available on http://www.ccdss.org/vrp. The benchmark suite contains instances with varying problem characteristics, like the number of customers, the distribution of customers, the

product distribution and the vehicle capacity. The instances are specifically designed to resemble scenarios taken from the distribution of food and petrol, which is reflected by additional problem characteristics like the number of compartments, the compartments' capacities and the incompatibilities between products and between products and compartments. The generated instances shall serve the vehicle routing community as benchmark suite for the VRPC.

We have implemented a portfolio of different heuristic components for solving the VRPC. Our solver suite covers a broad range of alternative approaches for construction (e.g. greedy insertion, Sweep, Savings), local search (2-opt, Or-opt, 2-opt*) and large neighbourhood search (order- and vehicle-based removal operators, greedy and regret-based insertion) as well as diverse meta-heuristics like Simulated Annealing, Record-to-Record Travel and Tabu Search. Many of the considered move operators are popular in vehicle routing, yet they had to be adapted to the specifics of the VRPC, in particular the solution representation which is order-based rather than customer-based and has to store the compartment assignments. Furthermore, we also considered some new operators like the vehicle-based removals.

Since our solver suite can be instantiated to a huge number of solution algorithms for the VRPC, we designed a series of experiments in order to identify a good algorithm for the VRPC as well as algorithmic components that are essential for good solution quality. The empirical results revealed that the considered algorithms are not sensitive to the structural difference of instances from petrol industry and food distribution. All diversification and intensification concepts are essential for obtaining the best solution quality. If the set of operators is chosen appropriately, then the adaption strategy (including noise) that chooses the operators based on their past performance is not necessary. Regarding construction algorithms, the best strategy is to generate different solutions and select the best for the subsequent optimization process. The choice of meta-heuristic control has a limited impact on the solution quality.

The comparison with the algorithms by El Fallahi et al. (2008) and Muyldermans and Pang (2007) showed that our algorithm which is designed for a more general class of VRPC instances is competitive on their specific problem instances. It is slightly better and faster than El Fallahi et al. (2008) but somewhat slower than Muyldermans and Pang (2007) for the smaller instances. After all, we were able to produce better solutions on most of the 23 considered instances within 20 min and on all but one instance within 60 min.

There are several opportunities for further research regarding the vehicle routing problem with compartments. The integer program formulation was not the primary focus of our work and can therefore certainly be improved. Our algorithms were compared empirically by average solution quality on 200 instances. It would be interesting to study the impact of instance characteristics on solution quality in more detail. Moreover, a search space analysis of the instances regarding specific neighbourhoods could provide further insight into obtained solution quality, cf. (Hoos and Stützle 2004). While the motivation of our algorithmic development was to study the extension of our earlier VRP-implementations to this specific richer class, the application of other strategies like variable neighbourhood search (cf. Mladenović and Hansen 1997) or iterated local search (cf. Lourenço et al. 2002) for combining the different neighbourhoods could be of interest as well as the study of population-based

approaches like evolutionary algorithms (cf. Michalewicz 1996), memetic algorithms (cf. Moscato 1999) or scatter search (cf. Glover et al. 2000).

The presented VRPC model is a basis for further extensions towards real-world scenarios involving compartments. Therefore, besides studying our model in more detail, the following extensions present interesting avenues for further studies: optimized splitting of an order to allow delivering one order in several compartments and considering vehicles with different compartment setups (number of compartments and capacities of compartments) in one model.

## References


Archetti C, Speranza M, Hertz A (2006) A tabu search algorithm for the split delivery vehicle routing problem. Transp Sci 40(1):64–73

Avella P, Boccia M, Sforza A (2004) Solving a fuel delivery problem by heuristic and exact approaches. Eur J Oper Res 152(1):170–179

Bartodziej P, Derigs U, Malcherek D, Vogel U (2009) Models and algorithms for solving combined vehicle and crew scheduling problems with rest constraints: an application to road feeder service planning in air cargo transportation. OR Spectr 31(2):405–429

Brown G, Graves G (1981) Real-time dispatch of petroleum tank trucks. Manag Sci 27(1):19–32

Chajakis E, Guignard M (2003) Scheduling deliveries in vehicles with multiple compartments. J Glob Optim 26(1):43–78

Christofides N, Mingozzi A, Toth P (1979) The vehicle routing problem. In: Christofides N, Mingozzi A, Toth P, Sandi C (eds) Combinatorial optimization. Wiley, Chichester, pp 315–338

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

Cordeau J-F, Gendreau M, Laporte G, Potvin J-Y, Semet F (2002) A guide to vehicle routing heuristics. J Oper Res Soc 53:512–522

Cordeau J-F, Gendreau M, Hertz A, Laporte G, Sormany J (2005) New heuristics for the vehicle routing problem. In: Langevin A, Riopel D (eds) Logistic systems: design and optimization. Wiley, Chichester, pp 279–298

Cornillier F, Boctor F, Laporte G, Renaud J (2008) A heuristic for the multi-period petrol station replenishment problem. Eur J Oper Res 191(2):295–305

Derigs U, Kaiser R (2007) Applying the attribute based hill climber heuristic to the vehicle routing problem. Eur J Oper Res 177(2):719–732

Derigs U, Li B, Vogel U (2009) Local search-based metaheuristics for the split delivery vehicle routing problem. J Oper Res Soc. doi:10.1057/jors.2009.100

Derigs U, Reuter K (2009) A simple and efficient tabu search heuristic for solving the open vehicle routing problem. J Oper Res Soc 60:1658–1669

Derigs U, Vogel U (2009) A computational study on neighborhood search heuristics for the open vehicle routing problem with time windows. In: MIC 2009: the VIII metaheuristics international conference

Dror M, Trudeau P (1989) Savings by split delivery routing. Transp Sci 23(2):141–145

Dueck G (1993) New optimization heuristics. J Comput Phys 104(1):86–92

Eilon S, Watson-Gandy C, Christofides N (1971) Distribution management: mathematical modelling and practical analysis. Griffin, London

El Fallahi A, Prins C, Wolfler Calvo R (2008) A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. Comput Oper Res 35(5):1725–1741

Fagerholt K, Christiansen M (2000) A combined ship scheduling and allocation problem. J Oper Res Soc 51(7):834–842

Gillett BE, Miller LR (1974) A heuristic algorithm for the vehicle-dispatch problem. Oper Res 22(2):340–349

Glover F (1989) Tabu search—part I. ORSA J Comput 1(3):190–206

Glover F (1990) Tabu search—part II. ORSA J Comput 2(1):4–32

Glover F, Laguna M, Marti R (2000) Fundamentals of scatter search and path relinking. Control Cybern 39(3):653–684

Hoos HH, Stützle T (2004) Stochastic local search. Foundations and applications. Elsevier/Morgan Kaufmann, San Francisco

Jetlund AS, Karimi IA (2004) Improving the logistics of multi-compartment chemical tankers. Comput Chem Eng 28:1267–1283

Kalkoff J (2006) Generierung von Benchmarks und empirische Analyse von Metaheuristiken für Tourenplanungsprobleme mit teilbaren Frachträumen. Diplomarbeit, Lehrstuhl für Wirtschaftsinformatik I, Universität Mannheim

Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680

Lin S, Kernighan B (1973) An effective heuristic algorithm for the traveling-salesman problem. Oper Res 21(2):498–516

Lourenço H, Martin O, Stützle T (2002) Iterated local search. In: Glover F, Kochenberger G (eds) Handbook of metaheuristics. Kluwer, Norwell, pp 321–353

Michalewicz Z (1996) Genetic algorithms + data structures = evolution programs, 3rd edn. Springer, Berlin

Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24(11):1097–1100

Moscato P (1999) Memetic algorithms: a short introduction. In: Corne D, Dorigo M, Glover F (eds) New ideas in optimization. McGraw-Hill, London, pp 219–234

Muyldermans L, Pang G (2007) On the benefits of co-collection: experiments with a multi-compartment vehicle routing algorithm (submitted)

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

Piesche M (2007) Entwicklung und Praxistest leistungsfähiger Meta-Heuristiken für das Vehicle Routing Problem mit teilbaren Frachträumen. Diplomarbeit, Seminar für Wirtschaftsinformatik und Operations Research, Universität zu Köln

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

Potvin J-Y, Rousseau J-M (1995) An exchange heuristic for routing problems with time windows. J Oper Res Soc 46(12):1433–1446

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

Schrimpf G, Schneider J, Stamm-Wilbrandt H, Dueck G (2000) Record breaking optimization results using the ruin and recreate principle. J Comput Phys 159:139–171

Shaw P (1998a) A new local search algorithm providing high quality solutions to vehicle routing problems. Technical report, APES group

Shaw P (1998b) Using constraint programming and local search methods to solve vehicle routing problems. In: Proceedings CP-98 fourth international conference on principles and practice of constraint programming

Toth P, Vigo D (2002) The vehicle routing problem. SIAM

Van der Bruggen L, Gruson R, Salomon M (1995) Reconsidering the distribution structure of gasoline products for a large oil company. Eur J Oper Res 81(3):460–473

Whittley I, Smith G (2004) The attribute based hill climber. J Math Model Algorithm 3(2):167–178