WORKING PAPER SERIES

A Genetic Algorithm for the Multi-Compartment Vehicle Routing Problem with Flexible Compartment Sizes

Henriette Koch/Tino Henke/Gerhard Wäscher

Working Paper No. 4/2016

OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

FACULTY OF ECONOMICS
AND MANAGEMENT

Impressum (§ 5 TMG)

Herausgeber:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Wirtschaftswissenschaft
Der Dekan

Verantwortlich für diese Ausgabe:
Henriette Koch/Tino Henke/Gerhard Wäscher
Otto-von-Guericke-Universität Magdeburg
Fakultät für Wirtschaftswissenschaft
Postfach 4120
39016 Magdeburg
Germany

http://www.fww.ovgu.de/femm

Bezug über den Herausgeber
ISSN 1615-4274

A Genetic Algorithm for the Multi-Compartment Vehicle Routing Problem with Flexible Compartment Sizes

Henriette Koch[a], Tino Henke[a], Gerhard Wäscher[a,b]

[a]Department of Management Science, Otto-von-Guericke-University Magdeburg, 39106 Magdeburg, Germany

[b]School of Mechanical, Electronic and Control Engineering, Beijing Jiaotong University, 100044 Beijing, China

Abstract

In this paper, a genetic algorithm for the multi-compartment vehicle routing problem with continuously flexible compartment sizes is proposed. In this problem, supplies of several product types have to be collected from customer locations and transported to a depot at minimal cost. In order to avoid mixing of different product types which are transported in the same vehicle, the vehicle's capacity can be separated into a limited number of compartments. The size of each compartment can be selected arbitrarily within the limits of the vehicle's capacity, and in each compartment one or several supplies of the same product type can be transported.

For solving this problem, a genetic algorithm is presented. The performance of the proposed algorithm is evaluated by means of extensive numerical experiments. Furthermore, the economic benefits of using continuously flexible compartments are investigated.

Keywords: vehicle routing, multiple compartments, genetic algorithm, heuristics

1 Introduction

In the classic capacitated vehicle routing problem (CVRP), goods have to be transported from a central depot to several customers. Each customer demands a certain quantity of goods and the available vehicles have limited capacities (see e.g. Dantzig and Ramser, 1959, or Toth and Vigo, 2014). In this paper, a variant of the CVRP is considered in which different product types have to be collected from customer locations and kept separated during transportation. For this purpose, the loading space of the collection vehicle can be divided into different compartments, where in each compartment a single product type can be transported.

In general, this problem can be classified as a multi-compartment vehicle routing problem (MCVRP). Problems of this kind arise when liquid or bulk products, or products which require different transportation conditions, e.g. different temperatures, are considered. Specific examples mentioned in the literature include the distribution of petroleum products, the transport of food of different levels of refrigeration, or the collection of glass waste of different colours. Previous research results suggest that substantial cost savings can be gained when vehicles with multiple compartments are used in such contexts compared to vehicles without compartments (Henke et al., 2015).

In most previously discussed MCVRP variants, compartment sizes are fixed and unchangeable. In contrast to this, a MCVRP with flexible compartment sizes is considered in this paper, i.e. the compartment sizes for each product type can be adjusted arbitrarily. Furthermore, the maximal number of compartments that can be used in one vehicle can be equal to the number of product types, but it can also be smaller. Hence, the problem is similar to the one presented by Henke et al. (2015) with the exception of one aspect. While compartment sizes can only be selected from a set of potential compartment sizes (discrete flexibility) in their variant, we consider compartment sizes which can be selected arbitrarily (continuous flexibility). In the following, the regarded problem will be referred to as the multi-compartment vehicle routing problem with continuously flexible compartment sizes (MCVRP-CFCS). Apart from the typical CVRP decisions about the composition of the tours, i.e. which customers are visited by each vehicle and in which sequence, it needs to be decided for each vehicle which product types are transported and which compartment sizes are chosen. A practical application for the MCVRP-CFCS is the distribution of food (Derigs et al., 2011), the shipment of bulk products (Fagerholt and Christiansen, 2000), or the collection of glass waste (Henke et al., 2015).

Only small problem instances of the MCVRP-CFCS can be solved to optimality by exact solution approaches. In order to solve larger problem instances, too, a genetic algorithm was developed and will be presented in this paper. Moreover, the benefits of using compartments with continuously flexible compartment sizes instead of discretely flexible ones will be analysed.

The remainder of this paper is organized as follows: In Section 2, a formal definition and a mathematical model for the MCVRP-CFCS are presented. Section 3 gives an overview of the relevant literature about the MCVRP. In Section 4, the proposed genetic algorithm is introduced and explained. The algorithm was tested by means of extensive numerical experiments. The experimental design and the corresponding results will be presented in Section 5. Finally, the main findings are summarized in Section 6.

2 Problem Description and Formulation

The MCVRP-CFCS can be stated as follows (Henke et al., 2015): Let $G = (V, E)$ be an undirected, weighted graph where $V = \{0, 1, \ldots, n\}$ represents the vertex set consisting of the depot ($\{0\}$) and n customer locations, and $E = \{(i,j): i, j \in V, i < j\}$ represents the set of edges which can be travelled between the locations. A non-negative cost c_{ij}, $(i, j) \in E$ is assigned to each edge. Furthermore, let P be a set of product types and $s_{ip} \geq 0$ the supply at location i of product type p ($i \in V \setminus \{0\}$, $p \in P$). These supplies have to be transported from the customer locations to the depot in separate product type-specific compartments. For this purpose, a set of homogenous vehicles K is available. Each of these vehicles has the same capacity Q which can be divided into a (limited) number of compartments $\hat{m} \leq |P|$. The size of a compartment can be selected arbitrarily between 0 and Q provided that the sum of the sizes of all compartments in a vehicle must not exceed its capacity. Since a vehicle might not be able to transport all product types at the same time, a customer location can be visited by more than one vehicle. However, an individual supply of a product type at a certain location must not be split onto several vehicles.

In order to solve this problem several partial decisions have to be made simultaneously: It has to be determined which product types are to be delivered by each vehicle. Moreover, each supply has to be assigned to one tour which also contains the decision about the assignment of customer locations to tours and about the compartment sizes, i.e. the compartment size for product type p in vehicle k results from the sum of all supplies of product type p that are assigned to vehicle k. Finally, for each tour the visiting sequence of the locations assigned to the tour has to be determined.

In order to formulate the mathematical model for the MCVRP-CFCS, the following decision variables are introduced:

$$u_{ipk} = \begin{cases} 1, \text{ if supply of product type p at location i is collected by vehicle k,} \\ 0, \text{ otherwise,} \end{cases} \qquad i \in V \setminus \{0\},\ p \in P,\ k \in K;$$

$$x_{ijk} = \begin{cases} 2, \text{ if } i = 0 \text{ and edge } (i,j) \text{ is used twice by vehicle k,} \\ 1, \text{ if edge } (i,j) \text{ is used once by vehicle k,} \\ 0, \text{ otherwise,} \end{cases} \qquad i,j \in V: i < j,\ k \in K;$$

$$y_{pk} = \begin{cases} 1, \text{ if a compartment for product type p is used in vehicle k,} \\ 0, \text{ otherwise,} \end{cases} \qquad p \in P,\ k \in K;$$

$$z_{ik} = \begin{cases} 1, \text{ if location i is visited by vehicle k,} \\ 0, \text{ otherwise,} \end{cases} \qquad i \in V, k \in K.$$

The model can be stated as follows:

$$\min\ z(S) = \sum_{(i,j)\in E} \sum_{k\in K} c_{ij} \cdot x_{ijk} \tag{1}$$

$$\sum_{k\in K} u_{ipk} = 1 \qquad \forall\, i \in V\backslash\{0\},\ p \in P:\ s_{ip}>0 \tag{2}$$

$$u_{ipk} \leq z_{ik} \qquad \forall\, i \in V\backslash\{0\},\ p \in P,\ k \in K \tag{3}$$

$$u_{ipk} \leq y_{pk} \qquad \forall\, i \in V\backslash\{0\},\ p \in P,\ k \in K \tag{4}$$

$$z_{ik} \leq z_{0k} \qquad \forall\, i \in V\backslash\{0\},\ k \in K \tag{5}$$

$$\sum_{\substack{j\in V:\\ j>0}} \sum_{k\in K} x_{0jk} \leq 2 \cdot |K| \tag{6}$$

$$\sum_{\substack{j\in V:\\ i<j}} x_{ijk} + \sum_{\substack{j\in V:\\ j<i}} x_{jik} = 2 \cdot z_{ik} \qquad \forall\, i \in V,\ k \in K \tag{7}$$

$$\sum_{p\in P} y_{pk} \leq \hat{m} \qquad \forall\, k \in K \tag{8}$$

$$\sum_{p\in P} \sum_{i\in V\backslash\{0\}} s_{ip} \cdot u_{ipk} \leq Q \qquad \forall\, k \in K \tag{9}$$

$$\sum_{i\in S} \sum_{\substack{j\in S:\\ i<j}} x_{ijk} \leq |S|-1 \qquad \forall\, k \in K,\ S \subseteq V\backslash\{0\}:|S|>2 \tag{10}$$

$$u_{ipk} \in \{0,1\} \qquad \forall\, i \in V\backslash\{0\},\ p \in P,\ k \in K \tag{11}$$

$$x_{0jk} \in \{0,1,2\} \qquad \forall\, j \in V\backslash\{0\},\ k \in K \tag{12}$$

$$x_{ijk} \in \{0,1\} \qquad \forall\, i \in V\backslash\{0\}, j \in V\backslash\{0\}:\ i < j,\ k \in K \tag{13}$$

$$y_{pk} \in \{0,1\} \qquad \forall\, p \in P,\ k \in K \tag{14}$$

$$z_{ik} \in \{0,1\} \qquad \forall\, i \in V,\ k \in K \tag{15}$$

(1) represents the objective function value z(S) of a solution Z, i.e. the total cost (travelled distance) of all tours which has to be minimized. Constraints (2) ensure that every positive supply is assigned to exactly one tour. The following constraints ensure that a supply can only be assigned to a tour which visits the respective location (3) and to which the respective product type has been assigned (4). Constraints (5)

guarantee that the depot is included in every tour. Furthermore, constraints (6) ensure that the maximum number of available vehicles is not exceeded. Constraints (7) represent the node degree constraints, guaranteeing that every location that is visited by a vehicle is also left by that vehicle again. Constraints (8) and (9) ensure that the maximum number of compartments per vehicle and the vehicle capacities, respectively, are not exceeded. Constraints (10) are the subtour elimination constraints and constraints (11) – (15) represent the variable domains.

The MCVRP-CFCS extends the CVRP by regarding multiple product types and multiple compartments with flexible sizes. Because the CVRP is already NP-hard, the MCVRP-CFCS is also NP-hard (see e.g. Toth and Vigo, 2014).

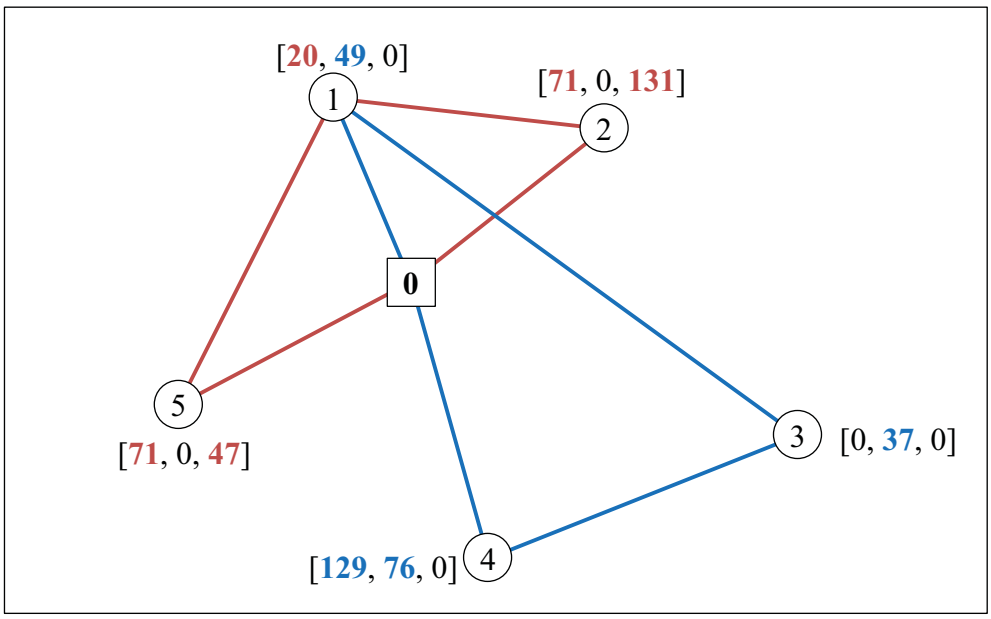


Figure 1: Solution to an example of the MCVRP-CFCS

Figure 1 shows an example for the MCVRP-CFCS with five customer locations, three product types and two available vehicles. Vertex 0 represents the depot; the other vertices represent the customer locations. The supplies for the different product types 1, 2 and 3 are given in square brackets. Each vehicle has a capacity of 350 units and can transport at most two different product types. The figure shows a feasible solution for the problem in which vehicle 1 (indicated by a red line) has compartments for product types 1 and 3 and which collects all supplies at locations 2 and 5 as well as the supply at location 1 for product type 1. Vehicle 2 (indicated by a blue line) has compartments for product types 1 and 2 and collects all supplies at locations 3 and 4 and the supply at location 1 for product type 2. The example further shows that some locations are only visited once, e.g. location 2, whereas other locations are visited multiple times, e.g. location 1.

3 Literature Review

One of the first studies about the MCVRP was published by Brown and Graves (1981). They present the case of an American oil company that aims at reducing its transportation costs for the distribution of different fuel products nationwide across the USA. In contrast to the MCVRP-CFCS, in the problem presented by Brown and Graves the compartment sizes are fixed, the vehicles are not loaded at a central depot but at more than 80 terminals, and (for technical reasons) they cannot deliver the same product type to more than one customer. The problem is formulated as a linear optimization problem and solved as such. However, due to nonsatisfying computing times of the optimal method, a heuristic was developed in order to achieve good solutions within fractions of a second.

Further applications of vehicle routing problems with multiple compartments were presented in the literature, later. Chajakis and Guignard (2003) describe a MCVRP for the transportation of food to convenience stores. The problem is formulated both as a binary and as a mixed-integer linear programming model and solved with a heuristic approach based on Lagrangian relaxations.

Avella et al. (2004) present another MCVRP in the context of petrol replenishment. In this problem, the compartment sizes are fixed and the compartments must be either completely empty or completely filled. The problem is formulated as a set partitioning problem and solved with a heuristic and a branch-and-price-algorithm.

Coelho and Laporte (2015) introduce a classification scheme for MCVRPs in fuel distribution with regard to two aspects. They distinguish between the cases in which the load of one compartment can or cannot be delivered to more than one customer, and the cases in which the demand for a single tank at a customer location can or cannot be delivered by multiple vehicles. They also propose a branch and cut algorithm which solves all four problem variants. According to this classification, the MCVRP-CFCS considered in this paper is a compartment split, tank unsplit problem.

Repoussis et al. (2006) introduce a MCVRP with a heterogeneous fleet and time windows for the depot. The problem is solved with a hybrid metaheuristic consisting of a GRASP algorithm and a variable neighbourhood search. El Fallahi et al. (2008) and Muyldermans and Pang (2010) consider MCVRPs with fixed compartment sizes and where each compartment is assigned a priori to a product type. El Fallahi et al. (2008) solve the problem with a tabu search procedure and a memetic algorithm. Muyldermans and Pang (2010) use a guided local search algorithm. A similar problem is discussed by Mendoza et al. (2010) and (2011). However, in contrast to the problem variants studied earlier, demands are considered to be stochastic. The problem is solved by use of construction heuristics and a memetic algorithm. Archetti et al. (2013) compare multi- and single-commodity vehicle routing problems, and provide insights on relationships between the split delivery vehicle routing problem and the MCVRP. Lahyani et al. (2015) present a case study of the collection of different types of olive oil in Tunesia, which represents an

extended MCVRP with multiple periods and the possibility of cleaning activities. For this problem, they propose a branch and cut algorithm.

As mentioned above, MCVRPs with flexible compartment sizes are rarely considered yet. One of the few exceptions is the study published by Derigs et al. (2011). They introduce a model which covers both fixed and flexible compartment sizes. Unlike other problems, a customer may further have multiple demands for the same product type. Thus, the same product type can be delivered to a customer by more than one vehicle.

Henke et al. (2015) introduce a MCVRP with discretely flexible compartment sizes. They developed a variable neighbourhood search procedure to solve this problem.

In addition, MCVRPs also occur in a number of maritime transportation problems where liquid or bulk products have to be transported (e.g. Fagerholt and Christiansen, 2000, Jetlund and Karimi, 2004, Al-Khayyal and Hwang, 2007).

4 Genetic Algorithm

4.1 Overview

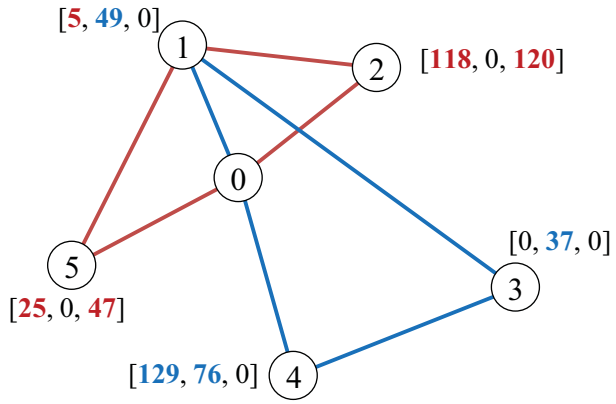
In order to solve the MCVRP-CFCS, a genetic algorithm (GA) has been developed. The concept of GAs was first introduced by John H. Holland (1975) and later adapted and applied to various optimization problems. GAs are inspired by natural processes where organisms evolve and reproduce over the course of several generations to be better adapted to environmental conditions. In the GA presented here, a population consisting of several individuals (solutions of the underlying optimization problem) is considered. Starting from an initial population, which is generated by specific construction methods, in each generation, i.e. iteration, two individuals of the population are selected as parents by means of a selection rule. These parents generate one offspring by application of a crossover operator, a mutation operator or both. In this way, the inheritance of properties of "good" parents should lead to the generation of a "good" offspring. If the offspring fulfils certain criteria with respect to its solution structure and its objective function value, it replaces one individual in the population and the process is repeated by selecting two new parent individuals. This procedure is continued until a pre-defined termination criterion is fulfilled. In the following subsections, the components of the GA are explained in detail.

4.2 Representation and Evaluation of a Solution

In a genetic algorithm, a solution is represented as a chromosome consisting of several genes. Within a chromosome, all properties of the solution are encoded. For routing problems, a permutation is often chosen as representation where each gene represents a customer location and where the order of the genes indicates the sequence in which the locations are visited (see e.g. El Fallahi et al., 2008, Talbi, 2009). In

the MCVRP-CFCS, however, not only the visiting sequence but also the allocation of supplies to vehicles and the respective compartment sizes for the different product types need to be derivable from a chromosome. For this purpose, a representation, which is based on the works of El Fallahi et al. (2008) and Pereira et al. (2002), is used. Each gene represents a positive supply of a specific product type at a specific customer location. The supplies are numbered lexicographically according to location index first and product type index second. Moreover, a chromosome does not consist of a single permutation, but of several strings containing ordered subsets of genes. Each of these subsets represents a tour. Let ρ be the number of positive supplies ($s_{ip} > 0$, $\rho \leq n \cdot |P|$), then a chromosome consists of ρ genes and up to $|K|$ strings.

Figure 2 shows the chromosome for the solution depicted in Figure 1. It consists of the two subsets of genes that are shaded in grey; the information for customer locations and product types depicted below the actual genes is only given for a better understanding. The sequence in which the supplies are collected and, thereby, the sequence in which the customer locations are visited, can be derived from the sequence in which the genes are arranged in the chromosome.



Tour 1	**8**	**9**	**1**	**3**	**4**
Location	*5*	*5*	*1*	*2*	*2*
Product	*1*	*3*	*1*	*1*	*3*

Tour 2	**2**	**5**	**6**	**7**
Location	*1*	*3*	*4*	*4*
Product	*2*	*2*	*1*	*2*

Compartment sizes:

		Product Type		
		1	**2**	**3**
Tour	**1**	148	0	167
	2	129	162	0

Figure 2: An example of a chromosome

The quality of a solution, i.e. its objective function value, is determined by the total cost of all tours in this solution: The lower the cost, the better is the solution. In a GA, however, an individual is traditionally evaluated by a fitness function where higher values indicate better solutions (see e.g. Talbi, 2009). Thus, the objective function value z(S) of a solution S is to be transformed into a fitness value f(S) by application of the following function, in which c_{max} represents the highest cost among all edges:

$$f(S) = n \cdot |K| \cdot c_{max} - z(S) \quad (1)$$

Apart from guaranteeing that a solution with a lower cost is associated with a higher fitness value, the fitness function also needs to ensure that the fitness value is never negative. This is achieved by including the first term $n \cdot |K| \cdot c_{max}$, which is a generous approximation of the total cost in the worst case.

4.3 Initial Population

An important design decision for a GA is the generation of the initial population. On the one hand, the initial population should contain sufficiently diverse solutions in order to avoid premature convergence towards a local optimum. On the other hand, the quality of the solutions in the initial population should be as high as possible in order to quickly generate better solutions. However, this might also lead to a loss of diversity of the population. To incorporate both ideas into the algorithm, one part of the initial population is generated by a completely randomized procedure whereas the other part is generated by a mainly deterministic construction procedure, namely the *savings heuristic* of Clarke and Wright (1964).

In both procedures all (positive) supplies are initially assigned randomly to vehicles. More precisely, in each step a supply that has not been assigned yet is selected at random and assigned to the vehicle with the lowest possible index. It can be assigned to a vehicle if a compartment for the respective product type has already been opened or if it can still be opened, and if the capacity of the vehicle is not exceeded by adding this supply. If a supply cannot be assigned to any of the $|K|$ vehicles, an additional artificial vehicle with capacity Q is considered. In order to account for the resulting infeasibility of the solution, every additional vehicle is penalized by adding a high value to the total cost of the solution resulting in a high reduction of the individual's fitness value.

For the first part of the population, the routes and, consequently, the chromosomes are mainly derived from the sequence in which the supplies were assigned to the tours. That is, if a supply of customer location i is assigned to a tour before a supply of customer location j, then location i is visited before location j and the chromosome is built accordingly. If a supply of a location is assigned to a vehicle that has already a supply from that location assigned to it, the customer location is not visited multiple times. Hence, in the chromosome, the gene for the new supply is arranged after the gene that was already created before. An example of how a solution is generated randomly is illustrated in Table 1. (The different colors represent the different customer locations.)

Step	Supply				$\hat{k}$	Tour # 1			Tour # 2		
	Gene	Cust.	Prod.	Qty.		String	Comp.	$\hat{Q}$	String	Comp.	$\hat{Q}$
0								350			350
1	1	1	1	20	1	1	1	330			350
2	4	2	3	131	1	1 4	1, 3	199			350
3	7	4	2	76	2*	1 4	1, 3	199	7	2	274
4	8	5	1	71	1	1 4 8	1, 3	128	7	2	274
5	2	1	2	49	2*	1 4 8	1, 3	128	7 2	2	225
6	5	3	2	37	2*	1 4 8	1, 3	128	7 2 5	2	188
7	6	4	1	129	2**	1 4 8	1, 3	128	7 6 2 5	1, 2	59
8	3	2	1	71	1	1 4 3 8	1, 3	57	7 6 2 5	1, 2	59
9	9	5	3	47	1	1 4 3 8 9	1, 3	10	7 6 2 5	1, 2	59

Table 1: Example for the generation of a solution for the initial population

(Cust.: customer, Prod.: product type, Qty.: quantity, Comp.: compartment, $\hat{k}$: assigned tour, $\hat{Q}$: remaining capacity;

*Assignment to tour # 1 not possible because of the restriction of the maximum number of compartments, **Assignment to tour # 1 not possible because of capacity restriction)

For the second part of the population, the routes (and thus also the chromosomes) are not constructed according to the chronological assignments of the supplies. Instead, the routes are constructed by application of the well-known savings heuristic of Clarke and Wright (1964). This can easily be done, since the supplies were already assigned to tours randomly in the previous step.

Finally, a solution is evaluated and included in the initial population if it is not a duplicate of another solution already existing in the initial population. For the identification of duplicates, a simple approach is used here: Two solutions are regarded as duplicates if they have the same fitness value. Although solutions can be wrongly identified as duplicates if they consist of the same routes with a different allocation of compartments or individual supplies, or if they consist of different routes with similar costs, tests have shown that this approach leads to better solutions than an exact search for duplicates. Presumably, this is the case because it is possible that two solutions with the same fitness value differ only slightly, e.g. one supply is assigned to another tour. If all these very similar solutions were included in the population, there would be a risk of a very low diversity within the population and, thus, of premature convergence.

4.4 Selection Operator

The selection operator determines how parent individuals are chosen for a crossover or a mutation. For the GA described in this paper, this is done by means of the so-called tournament selection (see e.g. Talbi, 2009): Two individuals are randomly selected from the population and the one with the higher fitness is chosen as a parent. The second parent is selected in the same way. It has to be ensured, though, that the first parent is not allowed to be selected as the second parent, too. Hence, the probability for an individual to be selected depends on its fitness value. Individuals with higher fitness values have a higher probability to be selected as a parent since the probability to have a "weaker opponent" is higher.

4.5 Crossover Operator

In the literature, crossover operators are frequently used, in which genes are exchanged between the two parents. In contrast, we use an approach in which genetic material is donated from one parent P_1 to the other one P_2, and which is based on the crossover operator proposed by Pereira et al. (2002) for the CVRP. Let K_1 and K_2 be the sets of actually used vehicles in the solutions of P_1 and P_2, i.e. $|K_1| \leq |K|$ and $|K_2| \leq |K|$. Firstly, a tour $t_1 \in K$ from P_1 is chosen randomly. The donated part of P_2 will be inserted (if possible) into this tour. If not all $|K|$ available vehicles are used in the solution represented by P_1 ($|K_1| < |K|$), a tour which is not used yet is selected with a probability of $1 / (2 \cdot |K_1|)$. Hence, the probability of opening a new tour is inversely proportional to the number of already used tours (Pereira et al., 2002). In this way, the possibility to open a new tour is given while it is simultaneously avoided to use an unnecessarily high number of vehicles.

Secondly, a tour $t_2 \in K_2$ from the donating parent P_2 is selected randomly. Note that this tour must be chosen from the set of actually used tours in order to guarantee that it contains genetic material. Then, from the string representing t_2, a sub-string is selected randomly which is to be donated to P_1. Since the inclusion of all genes of the selected sub-string into tour t_1 might lead to violations of certain constraints, it needs to be determined whether the whole sub-string or only some parts of it can be inserted into the genetic material of P_1 by regarding three constraints more closely. (1) The genes of a product type have to be excluded, for which no compartment is available in t_1 and for which no compartment can be opened anymore. (2) From the first to the last gene of the sub-string it needs to be checked whether the supplies represented by these genes can be added to tour t_1 without exceeding the respective vehicle's capacity. If adding a certain supply would exceed the capacity, the sub-string is only selected up to this supply, regardless of subsequent supplies which might have lower supplies. For example, let $\{g_1, \ldots, g_j, \ldots, g_m\}$ be a sub-string of length m. If the supply represented by gene g_j was the first to exceed the vehicle's capacity

(i.e. $\sum_{i=1}^{j-1} s_{customer(g_i),product(g_i)} \leq Q$ and $\sum_{i=1}^{j} s_{customer(g_i),product(g_i)} > Q$), the finally donated sub-string would be $\{g_1,\ldots,g_{j-1}\}$.

(3) In order to avoid identical genes within a chromosome, all genes included in the sub-string have to be removed from the genetic material of P_1. Finally, the remaining sub-string is inserted in t_1 behind the last gene of the location in t_1 that is closest to the first location in the sub-string.

Figure 3 gives an example to illustrate the crossover operator ($n = 5$, $p = 3$, $\hat{m} = 2$, $\rho = 15$). As before, different customer locations are indicated by different colors. As can be seen, $t_1 = 2$ and $t_2 = 1$ were determined and the sub-string to be donated is initially $\{3,4,10,9\}$. These genes represent supplies of the product types 1 and 3. For both product types there are compartments available in t_1. Hence, only the capacity constraint needs to be checked. Since the gene with value 4 is already contained in t_1, checking the capacity constraint is not necessary at this stage. Furthermore, assuming that the supplies with the gene values 3 and 10 can be added to the tour without exceeding the vehicle capacity, and that the supply represented by gene 9 would exceed the capacity, the final sub-string that will be donated is $\{3,4,10\}$. The respective genes are removed from the present genetic material of P_1 and the sub-string is inserted behind a gene belonging to the location closest to location 1. In the example location 1 is already contained in t_1. Thus, the sub-string can be inserted after the gene of this location. The resulting solution is the current offspring C. As can be seen, the operator allows to add (by selecting a tour t_1) and to remove tours (by removing duplicate genes) from a solution.

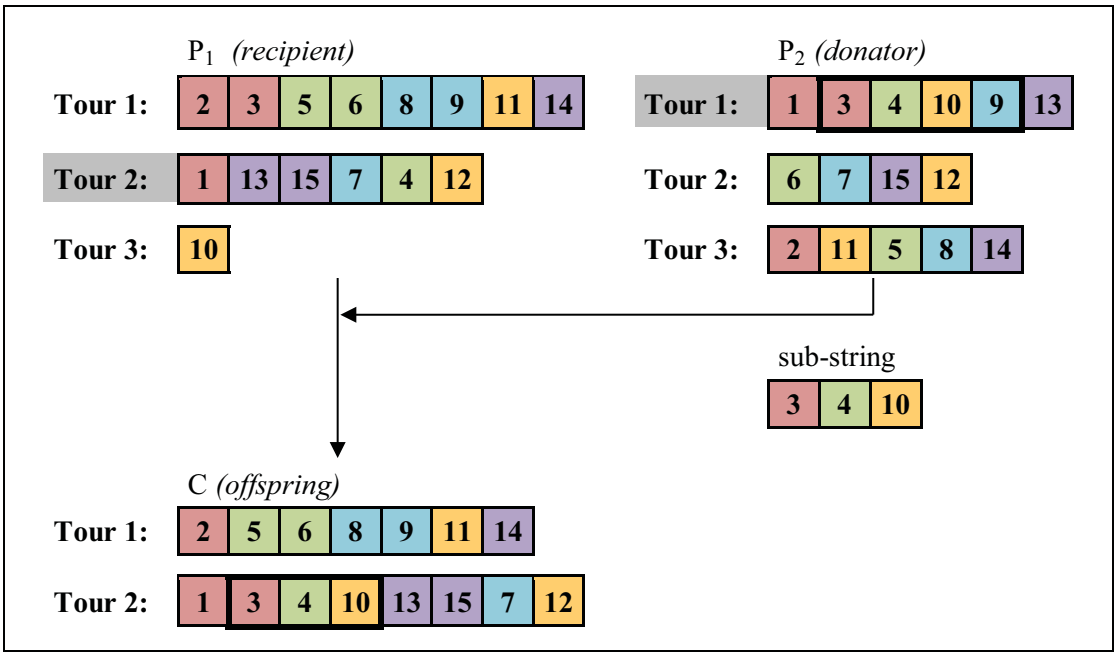


Figure 3: Example of a crossover operation

After applying the crossover operator, the use of an additional repair operator might be necessary. This is the case if an offspring resulted from the crossover in which one or more customer locations are visited more than once within one tour. In the example presented in Figure 3, customer location 4 is visited twice by vehicle 2. On the one hand, this violates constraint (7). On the other hand, this solution is always dominated by solutions in which location 4 is visited once – either between locations 2 and 5 or between location 3 and the depot – if the triangle inequality is fulfilled. The repair operator corrects such multiple visits by shifting the respective genes. With respect to the associated costs, it is checked whether it is better to visit customer location 4 between 2 and 5 or between 3 and the depot. Assuming that the first option is chosen, the resulting offspring would be:

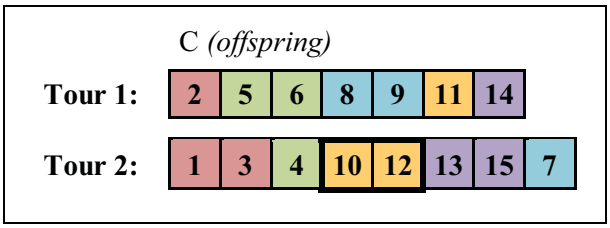


Figure 4: Offspring after the application of the repair operator

A crossover is executed with a certain probability p_{co} that is usually chosen in the interval between 0.5 and 1 (Talbi, 2009, Srinivas and Patnaik, 1994). Therefore, the operator is not used in some iterations. In this case, a mutation operator must be applied to create an offspring.

4.6 Mutation Operators

Mutations cause small, random changes in chromosomes in order to help to preserve the diversity within a population. In the presented GA, two different mutation operators are considered: swap and insertion.

In the swap operator, two genes are randomly chosen and – if possible – swapped. The swap is possible if capacity and compartment constraints are not violated. It might be necessary to apply the repair operator afterwards.

In the inversion operator, a tour and a sub-string within this tour are randomly chosen. Then, the sequence of the genes within this string is reversed. In order to avoid using the repair operator afterwards, the string is expanded if the selected sub-string would separate two genes representing the same customer location.

If a crossover was performed before, both operators will be applied to the offspring with low probabilities p_{swap} and p_{inv}. Mutation probabilities are typically in the range of 0.001-0.05 (Talbi, 2009, Srinivas and Patnaik, 1994). If the crossover operator was not used, one of the mutation operators is used with certainty on solution P_1. In this case, the probability for each operator to be chosen depends on the ratio of p_{swap} and p_{inv}.

4.7 Replacement Strategy and Termination Criteria

As mentioned in Section 4.1, the current offspring should replace an individual from the current population, i.e. the population is changing continuously throughout the procedure. This procedure corresponds to the so-called *steady state model* (see e.g. Talbi, 2009). An offspring is only included in the population if it is not a duplicate of an already existing solution (a duplicate is here defined in the same way as in Section 4.3) and if its fitness value is higher than the currently lowest fitness value in the population. It replaces a randomly chosen individual with a fitness value below the average fitness value in the population.

The algorithm stops if a certain number of iterations were performed or if no improvement of the currently best solution was achieved for a certain number of iterations.

5 Numerical Experiments

5.1 Overview

The presented GA was implemented in Microsoft Visual Studio C++ 2013 and the experiments were performed on a 3.07 GHz and 4 GB RAM computer.

For the numerical experiments, the small and the large instances introduced by Henke et al. (2015) were used. The experiments can be divided into three parts. In the first part, the economic advantage of using continuously flexible compartments is investigated, whereas in the remaining two parts the performance of the GA with respect to solution quality and computing time is analysed by solving the small instances on the one hand and the large instances on the other hand. The size of the small instances allowed for computing the optimal solutions within acceptable computing time by implementing the model presented in Section 2 in standard software for linear programming (CPLEX). In this way, it was possible to evaluate the solution quality of the GA by comparing the heuristic solutions to the optimal solutions. It was not possible to compare the GA with existing algorithms because problem variants solved by other algorithms are different from the MCVRP-CFCS. Therefore, further experiments on the large instances have been conducted where the performance of the GA was analysed over extensive computing times.

5.2 Problem Instances

In the following, the randomly generated instances of Henke et al. (2015) are described. In each instance, either n = 10 (small instances) or n = 50 (large instances) customer locations are considered. They further differ with respect to the number of product types ($|P| \in \{3,6,9\}$), the maximum number of compartments ($\hat{m} \in \{2,3\}$ for $|P| = 3$, $\hat{m} \in \{2,4,6\}$ for $|P| = 6$, $\hat{m} \in \{2,4,7,9\}$ for $|P| = 9$) and a parameter indicating the number of supplies $\bar{s}$ at the customer locations. For this parameter, three options are considered: small

($\bar{s} = 1$), medium ($\bar{s} = 2$), and large ($\bar{s} = 3$). For a small number of supplies, a supply for at least one but at most one third of the product types occurs at each customer location. For a medium level, each customer location has a supply for more than one third and at most two thirds of the product types. Finally, for a large number of supplies, each customer location provides more than two thirds of the product types. Thus, the instances can be divided into 27 problem classes with each class containing 50 different small instances and one large instance. The vehicle capacity has been set to 1,000 for each problem instance. By solving a corresponding bin-packing problem, the number of vehicles needed to guarantee feasible solutions were determined. Data sets and results are available for download at: http://www.mansci.ovgu.de/mansci/en/Research/Materials.html.

5.3 Comparison between discretely and continuously flexible compartments

While in the MCVRP-CFCS the compartment sizes can be varied continuously, Henke et al. (2015) considered a problem variant with discretely flexible compartment sizes. That is, the compartment sizes can be varied in equal steps based on a compartment unit size. This problem variant is motivated by a real-world application in the context of waste glass collection. In this problem, the vehicle capacity is divided by separation walls which can only be inserted at specific positions. Hence, the solution space is further restricted compared to the problem with continuously flexible compartment sizes. Consequently, objective function values of the MCVRP-CFCS cannot be higher than those obtained in the MCVRP with discretely flexible compartment sizes. In order to investigate how large the difference in the objective function values can be and under which problem characteristics continuously flexible compartments are especially beneficial, an analysis, which compares the optimal objective function values for both problem variants, has been performed. For this, the 1,350 small instances have been solved to optimality by implementing the model for the MCVRP-CFCS in CPLEX. The optimal solutions for the MCVRP with discretely flexible compartment sizes for the same instances were provided by Henke et al. (2015).

Table 2 compares the optimal objective values of the 1,350 problem instances for these two variants. The optimal objective function values differ by 2.83 % on average. 856 problem instances (63.4 %) have the same optimal objective function value in both problem variants; in the remaining 494 instances (36.6 %) a – in some cases quite considerably – better optimal objective function value has been obtained in the problem variant with continuously flexible compartment sizes. In some instances, the travelled distances could be reduced by up to about 31 %. In addition, it can be observed that the differences increase with the number of product types and the maximum number of compartments. An explanation for this finding can be given as follows: Due to the discreteness of the compartments, a part of a compartment's capacity might remain unused because the sum of the assigned supplies does not equal a feasible compartment size. The larger the number of compartments is, into which a vehicle's capacity is split, the larger this unused capacity can be. Consequently, more vehicles are necessary in order to collect all supplies. In contrast to

this observation, no conclusion can be drawn about the influence of the number of supplies. While the differences increase with an increasing number of supplies in some problem classes (e.g. $|P| = 9$, $\hat{m} = 9$), they decrease with an increasing number of supplies in others (e.g. $|P| = 3$, $\hat{m} = 2$). Based on these results, it can be concluded that continuously flexible compartments are especially beneficial in problems with large numbers of product types and compartments, whereas no significant difference compared to discretely flexible compartments occurs when the number of product types or compartments is relatively low.

Parameters			**z.exact**		**Difference**	
$\|P\|$	$\hat{m}$	$\bar{s}$	**Discrete**	**Continuous**	**Average**	**Max.**
3	2	1 (small)	366.90	363.80	0.86%	5.54%
		2 (medium)	465.35	464.68	0.13%	1.60%
		3 (large)	596.16	596.16	0.00%	0.00%
	3	1 (small)	349.08	343.32	1.76%	14.22%
		2 (medium)	337.15	332.33	1.65%	28.82%
		3 (large)	336.64	332.94	1.11%	7.44%
6	2	1 (small)	549.98	549.98	0.00%	0.00%
		2 (medium)	767.48	767.48	0.00%	0.00%
		3 (large)	919.78	919.78	0.00%	0.00%
	4	1 (small)	366.98	361.67	1.49%	8.81%
		2 (medium)	492.11	491.39	0.15%	2.60%
		3 (large)	584.46	583.18	0.18%	5.71%
	6	1 (small)	351.98	334.50	5.31%	24.26%
		2 (medium)	357.98	334.96	6.92%	21.30%
		3 (large)	357.42	335.21	6.67%	20.53%
9	2	1 (small)	724.17	724.17	0.00%	0.00%
		2 (medium)	1,114.25	1,114.25	0.00%	0.00%
		3 (large)	1,419.09	1,419.09	0.00%	0.00%
	4	1 (small)	463.05	461.78	0.29%	3.90%
		2 (medium)	647.57	646.41	0.18%	3.86%
		3 (large)	815.27	815.27	0.00%	0.00%
	7	1 (small)	360.16	341.17	5.58%	15.60%
		2 (medium)	456.97	449.27	1.71%	9.48%
		3 (large)	564.50	556.35	1.52%	7.85%
	9	1 (small)	354.98	321.28	10.39%	21.24%
		2 (medium)	392.48	345.78	13.63%	30.49%
		3 (large)	383.36	328.28	16.95%	31.43%
Minimum			336.64	321.28	0.00%	0.00%
Average			551.68	542.02	2.83%	9.80%
Maximum			1,419.09	1,419.09	16.95%	31.43%

Table 2: Comparison of the MCVRP with continuously and discretely flexible compartment sizes

5.4 Experiments with small, randomly-generated instances

In pre-tests, 50 out of the 1,350 small instances were selected randomly in order to determine the best parameters for the GA. For this purpose, the algorithm was applied 1,000 times to each of the 50 instances and for different parameter settings. The following parameters for the GA were determined:

- Population size $\sigma = 100$;
- Share of solutions in the initial population for which the savings heuristic is applied $\pi_{sav} = 0.5$;
- Crossover probability $p_{co} = 0.9$;
- Mutation probabilities $p_{swap} = 0.05$ and $p_{inv} = 0.05$;
- Termination criteria: Maximum number of iterations *iter_max* = 10,000 and maximum number of iterations without improvement *iter_max_impr* = 5,000.

In the main experiments, the GA was applied to all 1,350 instances. Each instance was solved five times with the GA to obtain reliable results. For benchmarking, these instances were also solved to optimality with CPLEX. The results of these experiments are shown in Table 3. For each problem class, it is shown how many instances could be solved to optimality (#opt) by the exact approach and by the GA. An instance is counted as solved to optimality by the GA if the optimal solution was found in at least one of the five runs. In addition, the column aver.opt indicates in how many of the five runs the optimal solutions were found on average. Furthermore, the average objective function values per problem class (z) obtained by the exact approach and by the GA, the average computing times in seconds per problem class for the exact approach and for one run of the GA (CPU), and the maximum computing time of a single instance in each problem class (CPU max) are presented. Finally, the last two columns show the average (aver.error) and the maximum (max.error) deviation of the average objective function values obtained by the GA from the average optimal values per problem class.

With an average deviation from the optimum of 0.48 % over all instances, the GA obtained good results. In 16 of 27 problem classes the GA was able to solve all 50 instances at least once to optimality, and in total 1,232 out of 1,350 instances (91.3 %) were solved optimally in at least one of the five runs. Furthermore, 834 instances (61.8 %) of all instances were solved to optimality in all five runs. While the average deviation from the optimal objective function value was lower than 0.5 % in 20 of the 27 problem classes, it was relatively high in the problem class with $|P| = 9$, $\hat{m} = 7$, $\bar{s} = 2$ with an average deviation of 3.63 %.

Parameters			CPLEX		GA				Error	
$\|P\|$	$\hat{m}$	$\bar{s}$	#opt	z	#opt	aver.opt	z	CPU	CPU max	aver.error	max.error
3	2	1 (small)	50	363.80	50	4.92	363.83	3.45	7.24	0.01%	0.54%
		2 (medium)	50	464.68	43	3.76	465.72	3.63	9.61	0.21%	6.54%
		3 (large)	50	596.16	49	4.66	596.42	4.53	16.30	0.07%	3.29%
	3	1 (small)	50	343.32	50	4.96	343.32	2.88	4.46	0.00%	0.00%
		2 (medium)	50	332.33	50	4.92	332.35	2.72	4.12	0.03%	7.52%
		3 (large)	50	332.94	50	4.78	333.40	3.61	5.04	0.05%	3.07%
6	2	1 (small)	50	549.98	50	5.00	549.98	4.10	5.54	0.00%	0.00%
		2 (medium)	50	767.49	50	4.92	767.67	6.00	8.39	0.01%	1.07%
		3 (large)	50	919.78	50	4.96	920.22	8.28	11.58	0.01%	0.92%
	4	1 (small)	50	361.67	50	4.64	361.98	3.25	8.42	0.15%	6.95%
		2 (medium)	50	491.39	39	2.84	495.82	5.63	15.73	0.96%	10.02%
		3 (large)	50	583.18	50	4.82	583.41	9.00	36.55	0.02%	1.09%
	6	1 (small)	50	334.50	50	4.94	334.59	2.57	4.04	0.01%	0.96%
		2 (medium)	50	334.96	50	4.42	335.64	4.49	6.79	0.37%	12.04%
		3 (large)	50	335.21	50	4.54	336.55	8.20	11.36	0.28%	18.33%
9	2	1 (small)	50	724.17	39	2.62	731.48	4.75	7.01	0.93%	12.00%
		2 (medium)	50	1,114.25	28	0.90	1,134.59	6.88	9.03	1.68%	10.23%
		3 (large)	50	1,419.09	39	2.70	1,421.28	14.99	22.76	0.14%	1.73%
	4	1 (small)	50	461.78	49	3.20	466.79	3.55	4.70	1.04%	12.39%
		2 (medium)	50	646.41	38	2.44	653.11	10.06	22.15	0.92%	6.62%
		3 (large)	50	815.27	42	2.74	819.09	14.61	29.39	0.42%	5.24%
	7	1 (small)	50	341.17	50	4.62	342.03	3.17	4.65	0.23%	8.77%
		2 (medium)	50	449.27	31	1.40	465.17	7.52	10.59	3.63%	27.42%
		3 (large)	50	556.35	35	2.12	563.33	15.97	45.01	1.24%	13.66%
	9	1 (small)	50	321.28	50	4.84	321.47	3.11	5.21	0.10%	7.44%
		2 (medium)	50	345.78	50	4.40	346.57	7.29	9.44	0.28%	11.67%
		3 (large)	50	328.28	50	4.20	329.91	22.73	50.81	0.28%	15.24%
Minimum			50	321.28	28	0.90	321.61	2.57	4.04	0.00%	0.00%
Average			50	542.02	45.63	3.90	544.86	6.92	13.92	0.48%	7.58%
Maximum			50	1,419.09	50	5.00	1421.19	22.73	50.81	3.63%	27.42%

Table 3: Results for the problem instances of Henke et al. (2015)

In addition, further tests were run with extended termination criteria. Table 4 shows the results of tests with *iter_max* = 100,000 and *iter_max_impr* = 50,000 in comparison to the previous results. As can be seen, improvements (shaded in green) were obtained in almost all problem classes with the total average deviation from the optimal solutions decreasing from 0.48 % to 0.33 %. The only (small) deterioration (shaded in red) could be observed in the problem class with $|P| = 9$, $\hat{m} = 9$, $\bar{s} = 2$ where one instance less than before could be solved to optimality. Moreover, 94.7 % of all instances could be solved to optimality at least once (compared to 91.3 % previously) and the optimal solutions were found in 84 % of all runs (compared to 78 % previously). Not unexpectedly, the computing time increased with the extension of the termination criteria. On average the computing times increased fivefold compared to the earlier tests.

Parameters			GA_orig				GA_ext			
$\|P\|$	$\hat{m}$	$\bar{s}$	#opt	z	CPU	aver.error	#opt	z	CPU	aver.error
3	2	1 (small)	50	363.83	3.45	0.01%	50	363.80	15.31	0.00%
		2 (medium)	43	465.70	3.63	0.21%	47	465.57	17.49	0.19%
		3 (large)	49	596.59	4.53	0.07%	50	596.17	22.45	0.00%
	3	1 (small)	50	343.32	2.88	0.00%	50	343.32	14.99	0.00%
		2 (medium)	50	332.46	2.72	0.03%	50	332.35	17.01	0.00%
		3 (large)	50	333.12	3.61	0.05%	50	333.12	21.17	0.05%
6	2	1 (small)	50	549.98	4.10	0.00%	50	549.98	21.88	0.00%
		2 (medium)	50	767.56	6.00	0.01%	50	767.48	28.29	0.00%
		3 (large)	50	919.84	8.28	0.01%	50	919.78	38.69	0.00%
	4	1 (small)	50	362.27	3.25	0.15%	50	361.94	17.25	0.07%
		2 (medium)	39	495.86	5.63	0.96%	41	494.79	25.35	0.73%
		3 (large)	50	583.28	9.00	0.02%	50	583.19	39.88	0.00%
	6	1 (small)	50	334.53	2.57	0.01%	50	334.53	16.37	0.01%
		2 (medium)	50	336.16	4.49	0.37%	50	335.44	25.21	0.14%
		3 (large)	50	336.16	8.20	0.28%	50	335.72	41.87	0.13%
9	2	1 (small)	39	730.81	4.75	0.93%	42	730.71	29.51	0.89%
		2 (medium)	28	1,132.89	6.88	1.68%	32	1,132.29	38.75	1.60%
		3 (large)	39	1,421.19	14.99	0.14%	44	1,419.77	57.74	0.05%
	4	1 (small)	49	466.56	3.55	1.04%	50	464.93	19.74	0.66%
		2 (medium)	38	652.29	10.06	0.92%	41	651.86	36.98	0.84%
		3 (large)	42	818.82	14.61	0.42%	47	818.02	59.23	0.33%
	7	1 (small)	50	342.05	3.17	0.23%	50	341.19	18.97	0.01%
		2 (medium)	31	465.34	7.52	3.63%	41	458.90	40.67	2.21%
		3 (large)	35	562.90	15.97	1.24%	44	560.76	88.92	0.82%
	9	1 (small)	50	321.61	3.11	0.10%	50	321.38	19.01	0.03%
		2 (medium)	50	346.76	7.29	0.28%	49	346.06	38.19	0.09%
		3 (large)	50	329.21	22.73	0.28%	50	328.41	88.68	0.03%
Minimum			28	321.61	2.57	0.00%	32	321.38	14.99	0.00%
Average			45.63	544.86	6.92	0.48%	47.33	544.13	33.32	0.33%
Maximum			50	1,421.19	22.73	3.63%	50	1,419.77	88.92	2.21%

Table 4: Results for the problem instances of Henke et al. (2015) with extended termination criteria

5.5 Experiments with large, randomly-generated instances

The last part of the experiments has been conducted with the large, randomly generated instances of Henke et al. (2015). In these instances, 50 customer locations are considered, while the remaining parameters are the same as in the small instances. For each problem class, one instance was randomly generated, i.e. there are 27 instances.

Since it was not possible to obtain optimal results for these instances within reasonable computing time, we adopted the same approach as Henke et al. (2015) to evaluate the GA. Each instance was solved for 360 minutes and the best solutions obtained after 10, 20, 30, 40, 50, 60, 120, 180, 240, 300, and 360 minutes were recorded. Finally, these solutions were compared to the best solutions found after 360 minutes.

Table 5 shows the development of the deviations from the best known solutions during the computation for each instance. The results indicate that the solutions found by the algorithm converge relatively quickly to the best known solution. After ten minutes of computing time the average deviation from the best known solutions amounts to 0.4 % only, where the largest gaps with respect to the best known solutions occur for the instances with nine product types and a large number of supplies. These instances consider up to 450 demands which results in very large chromosomes, longer computing times per iteration, and, thus, fewer iterations overall. Furthermore, the largest changes can be observed between the first and the second 10-minute interval in which the average deviation from the best known solution decreases by 0.18 %. As illustrated in Figure 5, these differences decrease as the search progresses. Therefore, it can be cautiously concluded that the objective function values converge.

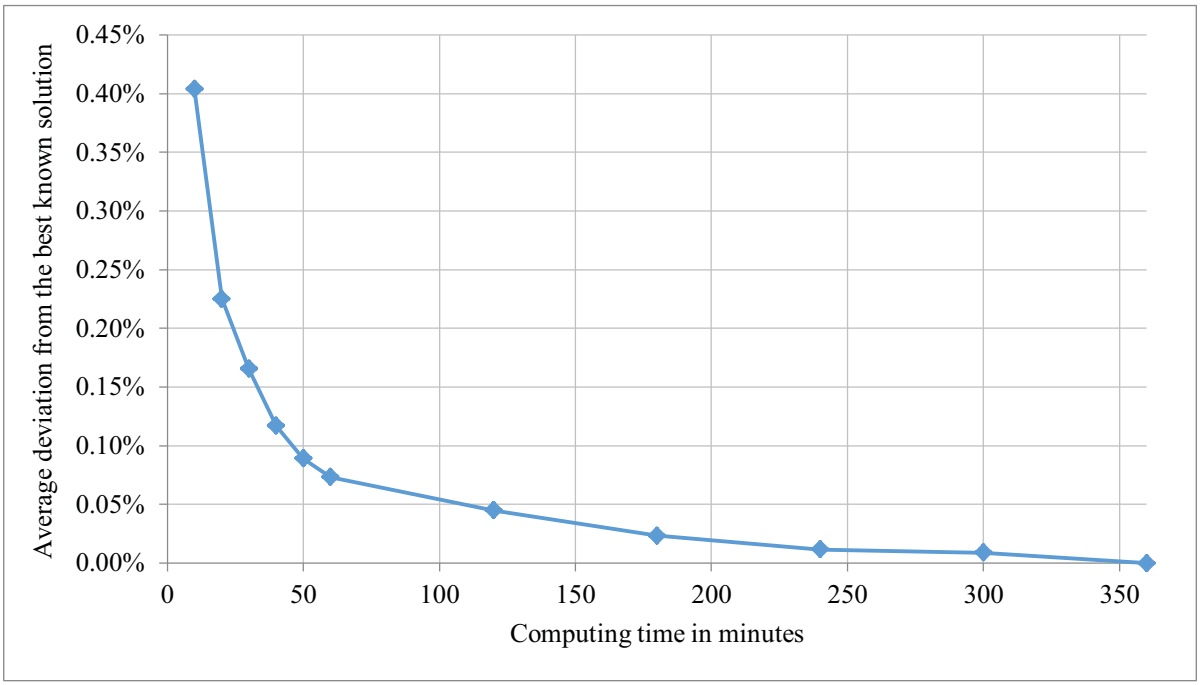


Figure 5: Average deviation from the best known solution plotted against computing time.

Parameters			Gap between the best solution and the best solution found after										Cost of the best solution after 360 minutes
$\|P\|$	$\hat{m}$	$\bar{s}$	10 Minutes	20 Minutes	30 Minutes	40 Minutes	50 Minutes	60 Minutes	120 Minutes	180 Minutes	240 Minutes	300 Minutes	
3	2	1 (small)	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1,021.80
		2 (medium)	0.08%	0.03%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1,131.76
		3 (large)	0.06%	0.05%	0.05%	0.05%	0.05%	0.05%	0.05%	0.05%	0.02%	0.00%	1,557.13
	3	1 (small)	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1,150.61
		2 (medium)	0.02%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1,154.71
		3 (large)	0.06%	0.03%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1,091.80
6	2	1 (small)	0.13%	0.03%	0.03%	0.03%	0.03%	0.00%	0.00%	0.00%	0.00%	0.00%	1,393.70
		2 (medium)	0.21%	0.15%	0.14%	0.14%	0.14%	0.14%	0.14%	0.00%	0.00%	0.00%	2,051.36
		3 (large)	0.51%	0.40%	0.39%	0.36%	0.21%	0.19%	0.15%	0.15%	0.15%	0.11%	2,391.80
	4	1 (small)	0.13%	0.13%	0.13%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	957.04
		2 (medium)	0.33%	0.04%	0.04%	0.03%	0.03%	0.03%	0.03%	0.02%	0.00%	0.00%	1,622.87
		3 (large)	0.64%	0.25%	0.11%	0.10%	0.06%	0.01%	0.01%	0.01%	0.00%	0.00%	1,790.39
	6	1 (small)	0.02%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1,053.79
		2 (medium)	0.18%	0.01%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1,033.70
		3 (large)	0.51%	0.14%	0.04%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1,081.50
9	2	1 (small)	0.11%	0.10%	0.10%	0.10%	0.10%	0.10%	0.10%	0.00%	0.00%	0.00%	1,671.82
		2 (medium)	0.33%	0.18%	0.15%	0.15%	0.15%	0.15%	0.15%	0.14%	0.00%	0.00%	3,014.97
		3 (large)	1.29%	0.74%	0.52%	0.41%	0.33%	0.29%	0.05%	0.03%	0.02%	0.02%	3,841.84
	4	1 (small)	0.07%	0.06%	0.06%	0.06%	0.06%	0.06%	0.06%	0.00%	0.00%	0.00%	1,320.10
		2 (medium)	0.45%	0.29%	0.16%	0.15%	0.15%	0.15%	0.06%	0.06%	0.00%	0.00%	1,749.34
		3 (large)	1.40%	1.00%	0.77%	0.57%	0.41%	0.33%	0.15%	0.02%	0.00%	0.00%	2,232.78
	7	1 (small)	0.11%	0.10%	0.10%	0.10%	0.10%	0.10%	0.10%	0.10%	0.10%	0.10%	1,290.30
		2 (medium)	0.47%	0.30%	0.18%	0.07%	0.07%	0.06%	0.06%	0.00%	0.00%	0.00%	1,251.43
		3 (large)	1.69%	1.12%	0.88%	0.51%	0.29%	0.17%	0.03%	0.03%	0.00%	0.00%	1,730.29
	9	1 (small)	0.11%	0.02%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1,068.93
		2 (medium)	0.44%	0.15%	0.08%	0.04%	0.04%	0.04%	0.04%	0.00%	0.00%	0.00%	1,151.05
		3 (large)	1.54%	0.78%	0.50%	0.27%	0.18%	0.09%	0.01%	0.01%	0.01%	0.01%	1,199.80
Minimum			0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	
Average			0.40%	0.23%	0.17%	0.12%	0.09%	0.07%	0.04%	0.02%	0.01%	0.01%	
Maximum			1.69%	1.12%	0.88%	0.57%	0.41%	0.33%	0.15%	0.15%	0.15%	0.11%	

Table 5: Results for large instances

6 Conclusions and outlook on future research

In this paper, a variant of the multi-compartment vehicle routing problem has been presented. Unlike in most of the previously studied problem variants, the compartment sizes can be varied completely freely. Moreover, the number of compartments that can be opened in a vehicle might be limited, i.e. it is possible that a vehicle cannot transport all available product types at the same time. Hence, additional decisions – compared to the classical CVRP – have to be made, namely which compartments are used on the individual tours and which sizes are chosen for these compartments. Due to the complexity of the problem, it can be solved to optimality only for smaller instances within reasonable computing times. Therefore, a genetic algorithm was developed which is based on different genetic algorithms for the CVRP from the literature. The algorithm was tested on 1,350 instances and obtained good results with an average deviation from the optimal solutions of 0.48 %. Moreover, it was shown that the possibility to vary the compartment sizes continuously can lead to significant cost savings compared to discretely flexible compartment sizes, especially in planning situations with a large number of product types and compartments. Although there is not much research dealing with the MCVRP-CFCS yet, these results might serve as an incentive to intensify future research about this problem and to develop efficient solution approaches. For the MCVRP-CFCS the development of a more sophisticated exact approach, which is able to solve larger instances to optimality, might provide the possibility to use more realistic benchmark solutions for further experiments. In addition, the analysis of problem modifications, e.g. stochastic supplies, and problem extensions, e.g. a multi-periodic context, could provide interesting research opportunities.

References

Al-Khayyal, F.; Hwang, S.-J. (2007): Inventory Constrained Maritime Routing and Scheduling for Multi-Commodity Liquid Bulk, Part I: Applications and Model. *European Journal of Operational Research* 176, 106-130.

Archetti, C.; Campbell, A.; Speranza, M.G. (2014): Multi-Commodity vs. Single-Commodity Routing. *Transportation Science*, in press. doi:10.1287/trsc.2014.0528.

Avella, P.; Boccia, M.; Sforza, A. (2004): Solving a Fuel Delivery Problem by Heuristic and Exact Approaches. *European Journal of Operational Research* 152, 170-179.

Brown, G.G.; Graves, G.W. (1981): Real-Time Dispatch of Petroleum Tank Trucks. *Management Science* 27, 19-32.

Chajakis, E.D.; Guignard, M. (2003): Scheduling Deliveries in Vehicles with Multiple Compartments. *Journal of Global Optimization* 26, 43-78.

Clarke, G.; Wright, J.W. (1964): Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research* 12, 568-581.

Coelho, L.C.; Laporte, G. (2015) Classification, Models and Exact Algorithms for Multi-Compartment Delivery Problems. *European Journal of Operational Research* 242, 854-864.

Dantzig, G.B.; Ramser, J.H. (1959): The Truck Dispatching Problem. *Management Science* 6, 80-91.

Derigs, U.; Gottlieb, J.; Kalkoff, J.; Piesche, M.; Rothlauf, F.; Vogel, U. (2011): Vehicle Routing with Compartments: Applications, Modelling and Heuristics. *OR Spectrum* 33, 885-914.

El Fallahi, A.; Prins, C.; Calvo, R.W. (2008): A Memetic Algorithm and a Tabu Search for the Multi-Compartment Vehicle Routing Problem. *Computers & Operations Research* 35, 1725-1741.

Fagerholt, K.; Christiansen, M. (2000): A Combined Ship Scheduling and Allocation Problem. *Journal of the Operational Research Society* 51, 834–842.

Henke, T.; Speranza, M.G.; Wäscher, G. (2015): The Multi-Compartment Vehicle Routing Problem with Flexible Compartment Sizes. *European Journal of Operational Research* 246, 730-743.

Holland, J.H. (1975): Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. Ann Arbor: University of Michigan Press.

Jetlund, A.S.; Karimi, I.A. (2004): Improving the Logistics of Multi-Compartment Chemical Tankers. *Computers and Chemical Engineering* 28, 1267-1283.

Lahyani, R.; Coelho, L.C.; Khemakhem, M.; Laporte, G.; Semet, F. (2015): A Multi-Compartment Vehicle Routing Problem Arising in the Collection of Olive Oil in Tunesia. *Omega* 51, 1-10.

Mendoza, J.E.; Castanier, B.; Guéret, C.; Medaglia, A.L.; Velasco, N. (2010): A Memetic Algorithm for the Multi-Compartment Vehicle Routing Problem with Stochastic Demands. *Computers & Operations Research* 37, 1886-1898.

Mendoza, J.E.; Castanier, B.; Guéret, C.; Medaglia, A.L.; Velasco, N. (2011): Constructive Heuristics for the Multicompartment Vehicle Routing Problem with Stochastic Demands. *Transportation Science* 45, 346-363.

Muyldermans, L.; Pang, G. (2010): On the Benefits of Co-collection: Experiments with a Multi-Compartment Vehicle Routing Algorithm. *European Journal of Operational Research 206*, 93-103.

Pereira, F.B.; Tavares, J.; Machado, P.; Costa, E. (2002): GVR: A New Genetic Representation for the Vehicle Routing Problem. *Artificial Intelligence and Cognitive Science* (Ed.: O'Neill, M. et al.). Berlin: Springer, 95-102.

Repoussis, P.P.; Tarantilis, C.D.; Ioannou, G. (2007): A Hybrid Metaheuristic for a Real Life Vehicle Routing Problem. *Numerical Methods and Applications. Lecture Notes in Computer Science 4310* (Ed.: Boyanov, T. et al.). Berlin, Heidelberg: Springer, 247-254.

Srinivas, M.; Patnaik, L.M. (1994): Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms. *IEEE Transactions on Systems* 24, 656-667.

Talbi, E. (2009): Metaheuristics: From Design to Implementation. Hoboken: Wiley.

Toth, P., Vigo, D. (2014): Vehicle Routing: Problems, Methods, and Applications (2nd ed.). Philadelphia: Society for Industrial and Applied Mathematics.

Otto von Guericke University Magdeburg
Faculty of Economics and Management
P.O. Box 4120 | 39016 Magdeburg | Germany

Tel.: +49 (0) 3 91 / 67-1 85 84
Fax: +49 (0) 3 91 / 67-1 21 20

www.fww.ovgu.de/femm

ISSN 1615-4274