Improved Packing and Routing of Vehicles with Compartments

Sandro Pirkwieser[1], Günther R. Raidl[1], and Jens Gottlieb[2]

[1] Institute of Computer Graphics and Algorithms
Vienna University of Technology, Vienna, Austria
{pirkwieser,raidl}@ads.tuwien.ac.at
[2] SAP AG, Walldorf, Germany
jens.gottlieb@sap.com

Abstract. We present a variable neighborhood search for the vehicle routing problem with compartments where we incorporate some features specifically aiming at the packing aspect. Among them we use a measure to distinguish packings and favor solutions with a denser packing, propose new neighborhood structures for shaking, and employ best-fit and best-fit-decreasing methods for inserting orders. Our approach yields encouraging results on a large set of test instances, obtaining new best known solutions for almost two third of them.

1 Introduction

We investigate the *vehicle routing problem with compartments* (VRPC) which has been tackled in the literature only very recently. We adhere to the definition of a rather general variant given by Derigs et al. [1]; see this original work for more details. In addition to the classical *vehicle routing problem* (VRP) a vehicle has several (at least two) compartments in which the customers' orders have to be placed. As in [1] we will consider the cases of having compartments which are flexible in size/capacity (but bounded by the total vehicle capacity), together with products that are only compatible with specific compartments, as well as fixed compartment capacities and product groups that might not be placed together in the same compartment. The first setting occurs in practice for food retail when delivering frozen and dry goods, whereas the second—and from a computational point of view more challenging and thus interesting—setting occurs when distributing petrol involving different fuel types. In fact, in case of the latter setting, the packing subproblem is NP-hard. El Fallahi et al. [2] and Muyldermans and Pang [3] considered a simpler scenario, comprising two compartments with fixed capacities and two product groups, each being compatible with only one compartment. Mendoza et al. [4, 5] tackled the VRPC (which they called the multi-compartment vehicle routing problem (MC-VRP)) with stochastic demands via several construction heuristics and a memetic algorithm.

Contrary to previous work we concentrate primarily on the packing aspect and introduce additional suitable neighborhood structures which contribute substantially to the overall success. Our proposed variable neighborhood search is described in the next section, experimental results are given in Section 3, and conclusions are drawn in Section 4.

2 Variable Neighborhood Search for the VRPC

Our heuristic solution approach is mainly based on *variable neighborhood search* (VNS) [6] and includes some of the problem-specific techniques from [1] which were reported to yield good performance. VNS is a metaheuristic that applies random steps in neighborhoods with growing size for diversification, referred to as shaking, and uses an embedded local search component for intensification. In the following we give an overview on our VNS for the VRPC.

Besides trying to minimize the total routing costs, which still is the ultimate goal, we also aim at increasing the *density*, i.e. the efficiency, of the packing. This measure is the average squared loading ratio (load divided by capacity) on a per compartment basis for fixed capacities and on a per vehicle basis otherwise. The basic idea is adopted from a concept introduced by Falkenauer and Delchambre for the one-dimensional bin packing and line balancing problem [7]. For now we only consider feasible solutions during search, hence no sort of repair operations or penalty terms for violations are necessary.

2.1 Initial Solution

As initial solution for the classical single-trajectory VNS we select the best solution out of several generated with variants of best insertion, the savings algorithm, and the sweep algorithm. We implemented two sweep-like algorithms: the first is similar to that of [1], whereas we did not search for the largest (radial) gap among all customers to obtain the beginning of the order sequence for insertion, but we select a customer at random, consider the successive one tenth of the customers and select the largest gap among them instead (sweep 1). In the second variant we do not only insert the orders in the current single open route but insert them in a greedy fashion considering all potential routes (sweep 2). In the results section we will state how often the individual methods could obtain the best initial solution for the instances considered.

2.2 Shaking Neighborhoods

In the shaking phase we utilize several move operations, i.e. we remove and reinsert a certain number of orders selected according to different criteria. On the one hand, we choose the orders either at random, based on the induced costs (or detour), or on their similarity to a randomly chosen seed order taking into account the product type, the demand and the customer location as in [1]. Regarding the removal based on the induced costs, unlike in [1] we consider orders belonging to the same customer as a set, otherwise only the first and last order of such a route subsequence would be “misplaced” (since the distance between orders of the same customer is zero). Whole sets of orders are also selected either

Table 1. Shaking neighborhoods and their order as applied by the VNS, newly proposed ones are marked bold.

k	$\mathcal{N}_k$
1	randomly remove orders
2	**remove random customers' orders**
3	remove orders of random route
4	remove orders of longest route
5	**remove orders of least density route**
6	**remove random compartments' orders**
7	**remove orders of non-empty least loaded compartment (having most orders)**
8	remove most costly orders (**as sets**) based on detour
9	remove orders based on similarity
10–15	exchange segments with lengths up to $k - 7$

via considering orders belonging to a certain customer, or being contained in a route which is itself selected at random, having the highest routing costs, or the least density. Similarly, such sets of orders might belong to a randomly selected compartment or the compartment with the least load. In the latter case we use the number of orders as a tie-breaking criterion and prefer a higher number, since several smaller orders are easier to reinsert. On the other hand, we also try to exchange route segments of limited size between two different routes as is often done in the context of VRPs. An overview of the shaking neighborhoods as well as their order is shown in Table 1.

2.3 Insertion of Orders

The insertion in a route's sequence is either done in a purely greedy and thus myopic way or using a regret-k heuristic [8, 1] which acts more foresighted. The latter takes the k cheapest routes' insertion costs into account; we randomly select k to be between two and five or equal to the number of all routes.

To improve upon the travel costs after the insertion we apply the well-known 3-opt as well as 2-opt* neighborhood structures. In 2-opt* all routes' end segments of all route pairs are tried to be exchanged, hence contrary to 3-opt also the packing needs to be checked and solved.

2.4 Solving the Packing Problem

Whenever assigning orders to a given route is permitted w.r.t. the capacity constraint of the vehicle a feasible packing needs to be determined, at least in case of having compartments of fixed capacity and more than one compartment available for one of the orders' product types. Only applying the capacity check is sufficient otherwise. A simple continuous lower bound is calculated beforehand per product type and summed up to exclude some cases for which no feasible

Table 2. Average results of both VNS variants on instances of type food compared to so far best solutions obtained by Derigs et al. [1]

n	p	VNS-FF			VNS-BFD		
		%-gap$_{\text{min.cost}}$	%-gap$_{\text{avg.cost}}$	%-gap$_{\text{avg.dens}}$	%-gap$_{\text{min.cost}}$	%-gap$_{\text{avg.cost}}$	%-gap$_{\text{avg.dens}}$
10	2	-0.16	0.35	5.85	-0.16	-0.14	6.86
	3	0.00	0.02	-0.05	0.00	0.00	0.35
25	2	-0.18	0.09	0.57	-0.23	-0.04	1.11
	3	-0.10	0.16	0.18	-0.26	-0.08	0.83
50	2	-0.35	-0.13	0.35	-0.36	-0.12	0.83
	3	-0.12	0.17	0.65	-0.37	-0.11	1.56
100	2	-0.30	-0.04	0.17	-0.24	0.07	0.23
	3	-0.45	-0.23	0.11	-0.75	-0.47	1.27
200	2	-0.23	0.43	-0.86	-0.12	0.28	-0.29
	3	-0.30	-0.04	-0.07	-0.24	-0.02	-0.06
avg.		-0.25	0.03	0.34	-0.34	-0.09	0.93

packings exist, i.e. whenever more compartments would be needed than the truck can offer. Then standard best-fit (BF) and best-fit-decreasing (BFD) strategies are applied when packing a single order and a set of orders, respectively. To speed up the packing process process and hence save computation time at a first attempt we try to solve an incremental packing problem via starting with the feasible packing at hand and trying to insert the new orders in a feasible way. Only if this fails we remove all assigned orders and try to find a packing from scratch.

When applying this procedure it is expected that the packing "degrades" over time and one increasingly fails to add orders to an existent packing. Hence, in order to maintain a rather favorable packing we additionally apply a local search specifically aiming at the packing and using the density as objective function to be maximized. This is achieved via once reinserting all orders of a route using BFD similar to the previously mentioned fallback strategy. Next we iteratively empty single compartments followed by applying several order exchange moves similarly to the heuristic for bin packing presented in [9]. The latter two moves are applied in a variable neighborhood descent fashion.

3 Experimental Results

The algorithm was implemented in C++, compiled with GCC 4.3 and executed on a single core of a 2.53 GHz Intel Xeon E5540 with 24 GB RAM, 3 GB RAM dedicated per core. For testing we used the instances introduced in [1] and available online at `http://www.ccdss.org/vrp/` together with the best known solutions. We consistently set a runtime limit of 10 minutes. The instances differ in type (petrol or food), number of customers (10 to 200, either clustered or not) and products (2 or 3), vehicle capacity (600 to 9000), and maximal order demand. We performed 10 runs per instance and setting and state following results:

Table 3. Average results of both VNS variants on instances of type petrol compared to so far best solutions obtained by Derigs et al. [1]

n	p	VNS-FF			VNS-BFD		
		%-gap$_{\text{min.cost}}$	%-gap$_{\text{avg.cost}}$	%-gap$_{\text{avg.dens}}$	%-gap$_{\text{min.cost}}$	%-gap$_{\text{avg.cost}}$	%-gap$_{\text{avg.dens}}$
10	2	0.00	0.08	-0.42	0.00	0.02	1.98
	3	0.00	0.37	0.04	0.00	0.39	0.79
25	2	-0.14	0.11	-0.18	-0.15	-0.01	0.90
	3	-0.37	0.57	-1.68	-0.45	0.43	-0.30
50	2	-0.56	-0.01	0.46	-0.54	-0.18	1.24
	3	-0.70	0.16	0.22	-0.77	0.01	0.70
100	2	-0.56	0.10	0.74	-0.87	-0.26	1.87
	3	-0.90	-0.06	1.98	-1.22	-0.27	2.29
200	2	-0.74	-0.31	-0.62	-1.47	-0.97	1.77
	3	-3.34	-1.63	2.50	-3.67	-2.32	4.79
avg.		-0.62	0.06	0.26	-0.77	-0.16	1.29

Table 4. Statistical significance results using a Wilcoxon rank-sum test with an error level of 5% given per instance type.

	food	petrol
VNS-BFD better than VNS-FF	13 (17.3%)	30 (24%)
VNS-FF better than VNS-BFD	7 (9.3%)	8 (6.4%)

the minimal and average travel cost as well as the average density as percentage gaps to the so far best known solution; note that in contrast to travel costs a higher density and hence a positive gap is generally better. We considered two variants of the VNS: One which utilizes best-fit, the density measure (including the neighborhoods based on it), and if appropriate the repacking heuristics (VNS-BFD), and another one using first-fit and none of the extensions despite the new neighborhoods not relying on the density or the load (VNS-FF). The number of orders to be removed and reinserted during shaking is between two and one third of all orders.

The results on the instances of type food are given in Table 2, those on the instances of type petrol in Table 3, and we averaged them for instances with the same number of customers n and products p. As expected the VNS benefits more from the extensions for the instances of type petrol. However, also for the food instances where we are faced with a considerably simpler packing subproblem a slight gain can be observed. Altogether, the performance of our algorithmic framework is very encouraging: For 145 out of 200 instances a new best known solution could be obtained by VNS-BFD, the same objective value was reached for 32 instances, and only for 23 instances the solution quality is slightly inferior. Remarkably, VNS-FF performs very similar w.r.t. the new best known solutions, but as is shown in Table 4 VNS-BFD is in total 43 times significantly better than VNS-FF and only 15 times worse.

Table 5. Percentage usage and success of the shaking neighborhoods averaged over all runs of VNS-BFD per instance type, newly proposed ones are again marked bold.

k	food		petrol	
	%-use	%-success	%-use	%-success
1	6.67	35.28	6.67	24.40
2	6.67	12.19	6.67	11.32
3	6.67	2.37	6.67	1.23
4	6.67	1.36	6.67	0.76
5	6.67	4.00	6.67	1.24
6	6.67	9.12	6.67	22.51
7	6.67	11.41	6.67	16.22
8	6.67	2.23	6.67	2.38
9	6.67	16.90	6.67	18.76
1–9	60.00	94.86	60.01	98.81
10	6.67	0.87	6.67	0.28
11	6.67	0.89	6.67	0.21
12	6.67	0.83	6.67	0.18
13	6.67	0.79	6.67	0.19
14	6.67	0.90	6.67	0.17
15	6.67	0.85	6.67	0.16
10–15	40.00	5.14	39.99	1.19

To gain insight in the usefulness of the different shaking neighborhoods we state both, the relative usage and the relative success in percent in Table 5, where as success we count all moves leading to an improved solution. The table reveals that the newly proposed neighborhoods achieve around half of all improvements and that especially the compartment-based variants are very successful when having fixed compartment sizes. Further, exchanging route segments yields a relevant benefit only for the food instances. Due to many moves not leading to an improvement, the usage rate is basically the same for all neighborhoods.

We mentioned in Section 2 that it is better to generate several initial solutions using different construction heuristics and select the best one, since the performance of the heuristics heavily depends on the instance type. A rough indication on the overall performance is shown in Table 6, determined over all conducted runs. For the food instances sweep 1 performs best, whereas for the petrol instances sweep 2 most often yields the best initial solutions.

Finally, we take a look at solving the NP-hard packing subproblem occurring at the petrol instances. More concretely we investigate how often the algorithm ends up in a state where it is unclear whether a feasible packing exists or not, i.e. summing up the (non-continuous) free space would allow an insertion but it is unable to pack the order(s); these results are given in Table 7. Inserting single orders with BF becomes harder with increasing instance size, whereas in the case of inserting several orders with BFD the instances having 50 customers appear

Table 6. Number of times in percent the respective construction heuristic yielded the best initial solution.

heuristic	× best initial solution[%]	
	food	petrol
best insertion	22.00	23.36
savings	22.26	9.12
sweep 1	32.66	30.32
sweep 2	23.06	37.20

Table 7. Average frequency of undecided packings of VNS-BFD on instances of type petrol in dependence of the number of nodes and number of products.

n	p	petrol	
		BF[%]	BFD[%]
10	2	0.07	0.00
	3	0.17	0.05
25	2	0.54	0.12
	3	1.13	3.26
50	2	2.46	3.60
	3	2.52	2.48
100	2	4.66	0.89
	3	4.08	0.36
200	2	11.85	0.72
	3	11.64	0.31
		3.21	1.27

hardest. Since the packing subproblem was so far only solved heuristically, this led us to investigate it more closely. We decided to check those cases when BFD could not obtain a feasible packing in an exact way, since it is only applied when exchanging route segments during shaking and in the course of applying 2-opt*, hence much less often than BF. However, for all these instances not even a single additional feasible packing could be found. Therefore it can be concluded that the packing problem appears not to be hard in this case.

4 Conclusions

We presented a variable neighborhood search (VNS) for the vehicle routing problem with compartments. Our study focused on the packing aspect of the problem, considering among others a measure to distinguish packings and favor solutions with a denser packing, additional neighborhood structures for shaking, e.g. emptying weakly packed compartments and clearing routes with bad packing. In addition, we propose best-fit and best-fit-decreasing packing strategies to re-insert orders that were previously removed by shaking operators. The results are very encouraging, for nearly two third of all instances a new best known solution was found. The algorithm performs especially well on the petrol instances where the compartment capacities are fixed and the products are incompatible.

To more thoroughly investigate solving the packing problem we decided to modify the available instances to exhibit a more challenging packing aspect, i.e. being less well-formed w.r.t. the order demands. First results indeed show more clearly a gain due to the proposed extensions, and also when exactly solving some packings a small but significant gain is observable this time. Another future topic could be to utilize methods for determining stronger lower bounds, in order to avoid some cases of wasting time by handing an infeasible packing problem to an exact method. For completeness it is also planned to perform tests on other instances previously used in the literature, though their packing subproblem is also not hard.

References

1. Derigs, U., Gottlieb, J., Kalkoff, J., Piesche, M., Rothlauf, F., Vogel, U.: Vehicle routing with compartments: applications, modelling and heuristics. OR Spectrum (2010) Available online, DOI: 10.1007/s00291-010-0194-3.
2. El Fallahi, A., Prins, C., Calvo, R.W.: A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. Computers & Operations Research **35** (2008) 1725–1741
3. Muyldermans, L., Pang, G.: On the benefits of co-collection: Experiments with a multi-compartment vehicle routing algorithm. European Journal of Operational Research **206** (2010) 93–103
4. Mendoza, J.E., Castanier, B., Guéret, C., Medaglia, A.L., Velasco, N.: A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands. Computers & Operations Research **37**(11) (2010) 1886–1898
5. Mendoza, J.E., Castanier, B., Guéret, C., Medaglia, A.L., Velasco, N.: Constructive heuristics for the multi-compartment vehicle routing problem with stochastic demands. Transportation Science (2010) (Forthcoming).
6. Hansen, P., Mladenović, N., Brimberg, J., Moreno Pérez, J.A.: Variable neighborhood search. In Gendreau, M., Potvin, J.Y., eds.: Handbook of Metaheuristics, 2nd edition. Springer (2010) 61–86
7. Falkenauer, E., Delchambre, A.: A genetic algorithm for bin packing and line balancing. In: Proceedings of the 1992 IEEE International Conference on Robotics and Automation. Volume 2. (May 1992) 1186–1192
8. Pisinger, D., Ropke, S.: A general heuristic for vehicle routing problems. Computers & Operations Research **34**(8) (2007) 2403–2435
9. Levine, J., Ducatelle, F.: Ant colony optimization and local search for bin packing and cutting stock problems. Journal of the Operational Research Society **55** (2004) 705–716