A Tabu Search Algorithm for the Capacitated Vehicle Routing Problem with Three-Dimensional Loading Constraints

Marco A. Wisniewski, Marcus Ritt, Luciana S. Buriol

Instituto de Informática - Universidade Federal do Rio Grande do Sul - UFRGS
{mawisniewski, marcus.ritt, buriol}@inf.ufrgs.br

Abstract

This paper considers the three-dimensional loading capacitated vehicle routing problem. It consists in finding a shortest routing of a fleet of vehicles through a network so as to deliver goods to customers and loading those goods into the vehicle satisfying three-dimensional loading constraints. We address the routing problem by means of a tabu search and a first-improvement local search. The loading is efficiently done by a randomized bottom-left based algorithm that uses a multiple-views data structure. Through computational experiments we show that our approach is able to produce quality solutions comparable to or better than the best ones reported in the literature.
KEYWORDS. 3L-CVRP, Vehicle Routing, Container Loading.

Resumo

Consideramos neste artigo o problema de roteamento de veículos capacitados com restrições de empacotamento tridimensionais (3D-Loading Capacitated Vehicle Routing Problem - 3L-CVRP), que consiste em encontrar um roteamento de menor distância de veículos através de uma rede de rodovias a fim de entregar produtos a clientes e o empacotamento desses produtos nos veículos com restrições tridimensionais. O nosso método utiliza uma busca tabu para o roteamento e um algoritmo randomizado baseado na heurística inferior-esquerda que eficientemente empacota as caixas com o auxílio de uma estrutura de representação com múltiplas vistas. Através de experimentos computacionais nós mostramos que o nosso algoritmo é capaz de produzir soluções comparáveis ou melhores que as publicadas na literatura.
PALAVRAS CHAVE. 3L-CVRP, Roteamento de Veículos, Empacotamento.

1 Introduction

The capacitated vehicle routing problem with three-dimensional loading constraints (3L-CVRP) is characterized by the combination of vehicle routing and container loading in 3D-space. The problem, which was introduced by Gendreau et al. (2006), results from the need to consider loading of rectangular boxes in a fleet of vehicles while determining routes so that these vehicles are able to deliver the previously packed goods to a set of customers. The objective is then to minimize the traveling costs while providing a feasible loading for each vehicle.

The 3L-CVRP generalizes two problems that are by themselves already hard to solve practically, namely the routing of vehicles and the loading of boxes into a three-dimensional space, and is therefore a challenging optimization problem. The problem takes into account a diverse number of constraints taken directly from real-world scenarios which makes it relevant to real cargo transportation. Not only is the optimization problem NP-hard (Gendreau et al., 2006), but so is finding feasible solutions, since the same holds for deciding whether a set of three-dimensional boxes can be packed in a container (Martello et al., 2000).

Being a relatively new problem of high complexity, only a few solutions appear in the literature. Gendreau et al. (2006) proposed a solution where both the routing and loading sub-problems were solved by independent tabu search based algorithms. de Araújo (2006) achieved better results with a more sophisticated packing algorithm. Tarantilis et al. (2009) followed with a tabu search with guided local search coupled with a few fast and simple packing heuristics. The solution quality was then improved by Fuellerer et al. (2010) with an ant colony optimization approach, again with fast heuristics for the loading.

Most recently published solutions utilize some form of tabu search. Tao and Wang (2010) achieve the currently best results with a least waste packing heuristic. Wang et al. (2010) get similar solutions with fine-tuning of the bottom-left and maximum touching area heuristics for the loading. Finally, Bortfeldt (2010) gets comparable results with a fraction of the computational effort needed by the other methods by selectively packing the generated search neighborhood using a tree-search algorithm. Iori et al. (2007) presented an exact branch-and-cut algorithm for the 2L-CVRP, but, to our knowledge, there is still no exact solution for the 3L-CVRP.

In this paper we propose a tabu search with a first improvement local search algorithm for the 3L-CVRP. We found the efficiency of the packing sub-routine to be crucial, and for that we rely on multiple permutations of the bottom-left heuristic adjusted for the 3D-space. Our main contributions are a new randomized packing algorithm, a new multiple-views dynamic matrix representation and a first improvement based tabu-search.

The remainder of this paper is organized as follows. The next section properly defines the problem. Section 3 details our proposed tabu search as well as the packing procedure. The results of computational experiments are shown in Section 4. Our conclusions are presented in Section 5.

2 Problem Definition

We consider a complete graph $G = (V, E)$. Let $V = \{v_0, v_1, \ldots, v_n\}$ be the set of $n + 1$ vertices corresponding to the depot v_0 and the n customers $v_1, \ldots, v_n$. Each edge $e_{ij} \in E$ between v_i and v_j has an associated routing cost c_{ij}. We are given a fleet of v identical vehicles. Each vehicle has a total weight limit D and a three-dimensional container with width W, height H and length L. The vehicle has an opening on the rear of size WH for loading and unloading.

Each customer i has a demand of m_i items. Each item is denoted by I_{ik}, where $i = 1, \ldots, n$ represents the customer to whom the item must be delivered and $k = 1, \ldots, m_i$ specifies the item. The total weight of the set of items m_i is d_i. Each item has width w_{ik}, height h_{ik} and length l_{ik} $(i = 1, \ldots, n; k = 1, \ldots, m_i)$.

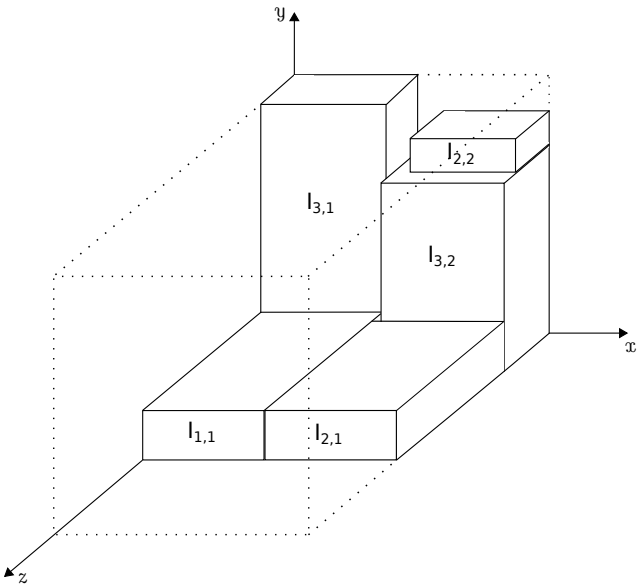


Figure 1: A feasible packing plan.

A valid route is a sequence of three or more nodes starting and ending at v_0 where no customer appears more than once. The total cost of the route is the sum of all of its composing edges. A solution to the 3L-CVRP is a set of at most feasible v routes such that each customer is visited exactly once and a corresponding valid packing plan is provided for the route. The objective is to minimize the total traveling cost associated with routing.

A packing plan is feasible if all of the following constraints are satisfied:

i) All items of the corresponding customers that appear in the corresponding route are properly packed with no overlap and are completely contained by the container.

ii) The orientation of all boxes is fixed with respect to the height and they can only be rotated by 90° horizontally. The edges of the boxes must be parallel to the sides of the container.

iii) Each item has an associated fragility flag f_{ik} which is set to one if item I_{ik} is fragile and zero otherwise. Non-fragile items cannot be placed on top of fragile ones.

iv) Each box must have a minimum supporting area in contact with its base. All items that are not placed directly on the floor must have a percentage of its base in direct contact with other boxes. This is usually set to 75%.

v) When unloading items of a customer i, it must be possible to do so for all of his items by only using movements parallel to the length of the vehicle. This means that no item of another customer which is visited later on the route can be placed on top of or between I_{ik} and the rear of the vehicle for all $k = 1, \ldots, m_i$. This policy is hereupon referred to as LIFO (last in first out).

Figure 1 gives an example of a feasible packing plan.

3 Proposed Algorithm

Our solution uses the same two-stage approach proposed by Gendreau et al. (2006), that is, we consider routing and packing separately, each with its own algorithm, which are then coordinated to produce a complete solution.

We first describe the data structure used to represent the loading space of vehicles. After that we detail our packing algorithm. Then, we present our tabu search strategy which iteratively calls the packing subroutine.

3.1 Container Representation

The packing procedure is called several times throughout a complete execution, therefore it is essential for it to give good results in reasonable time. Furthermore, we have to make sure that our representation is powerful enough so that whenever a configuration that has a feasible solution arises we are able to represent it and not incorrectly discard it.

We achieve this by extending a dynamic matrix proposed by Ngoi et al. (1994) with multiple views. The main idea behind using a dynamic matrix is to represent the vehicle in only two dimensions and to do so by dividing it into a minimum number of cells. Each box is then completely described in terms of the cells it occupies. The container starts with only one cell representing all of its available space. As more items get packed, the cells are subsequently divided as necessary.

The main problem with the dynamic matrix is that it has only partial information with regard to the third dimension (height). This means that space can be wasted if there is only one active view "from above" at each time. A lot of feasible packing plans generated by the algorithm are then overlooked because we are not able to efficiently use empty space below already packed boxes. This problem only arises, however, when a box is positioned with less than 100% of supporting area. To solve this, we create a new view at the height of the base each time a box is placed not completely supported if there is not already a previously created view at that height. This has a very small impact in performance since both cells and views are very few in number given the usual dimensions involved.

This representation has an advantage over simpler ones in cases where an insertion of a new box extends an area of support previously provided by the top side of another adjacent box. For example, in Figure 1, $I_{1,1}$ and $I_{2,1}$ provide a combined surface area upon which a large box may be placed. This also holds even if there are gaps and other boxes in between.

3.2 Packing Algorithm

For a given route, the packing procedure should position all items of all customers that are visited throughout the route, while satisfying all constraints described in Section 2. Our algorithm uses a heuristic similar to one used by Gendreau et al. (2006), which given a sequence of items, greedily places each one on the free position with minimum z, breaking ties by minimum x and then minimum y. The item is first placed with one of its sides parallel to the sides of the container, and then rotated by 90° in the $x - z$ plane if the first orientation does not fit. This is called the 3D bottom-left heuristic.

The algorithm then places all items in a container of width W, height H and infinite length. The goal is to minimize the used length without violating any of the constraints. The resulting packing plan is considered feasible if it has length lower or equal to L.

An important observation is that the ordering of the items greatly impacts the heuristic. By changing it we are able to generate different packing plans. With that in mind, a number of different permutations are tested until a feasible packing is found or the number of iterations reaches a limit θn^2, where n is the number of items in the sequence.

A good starting sequence is found if we sort the items in the reverse order in which their corresponding customers will be visited, so as to increase the chances of generating solutions that satisfie the LIFO constraint. Let $seq_0 = (1, \ldots, n)$ be that sequence. We then perturb it to create a new one with some positions changed. It is fundamental to note that say $seq_1 = (2, 1, 3, 4, \ldots, n)$ will be feasible (will satisfy LIFO) with much greater probability than for ex-

ample $seq_2 = (n, n-1, \ldots, 1)$. Taking that into account, we propose the randomized Algorithm 1 for generating box orderings.

Algorithm 1 Packing Routine

Input: a list of boxes to be packed
Output: a packing plan
repeat
  for $i \leftarrow 0$ **to** θn^2 **do**
    create indexing vector idx where $idx[i]$ points to box i
    for all $idx[i]$ **do**
      $idx[i] \leftarrow random()$ // random positive integer
      $idx[i] \leftarrow idx[i]/(random()$ **mod** $(n-i+1))$
      $idx[i] \leftarrow idx[i]/(random()$ **mod** $(n-i+1))$
    end for
    sort idx in non-decreasing order
    call $3lbottomleft$ heuristic with boxes in order determined by its corresponding idx
  end for
until $solution$ is feasible **or** $solution$ is not improved

When we find a feasible packing for a group of customers we cache the solution so subsequent calls with the same arguments result in only a cache look-up. If we cannot find a feasible packing though, we do not store that result immediately. This is due to the fact that by its randomized nature it is possible for the algorithm to find a better packing in another call with the exact same arguments. We do, however, give up after τ attempts, so as to not waste time on impossible or very difficult routes. In this case the solution is marked infeasible and cached to avoid further attempts to pack it.

3.3 Routing

The routing part of the 3L-CVRP is addressed by a tabu search. The initial solution is constructed by the savings algorithm by Clarke and Wright (1964) and then improved upon by searching through its neighborhood. The savings algorithm works as follows. First each customer is placed in a separate route. A list of savings is then constructed, where for each pair of customers (i, j) we calculate its saving as $c_{i0} + c_{0j} - c_{ij}$ which corresponds to the reduction of the distance when merging two routes with endpoints i and j. We then greedily pick the pair with the largest saving of the merged endpoints iteratively and merge their corresponding routes if we can find a suitable packing plan. If after that we have a routing solution with more routes than v, we again greedily pick the best pair and merge their corresponding routes even if we cannot find a feasible loading, and then repeat that process until the number of routes is equal to v. In this case the initial solution will not be feasible.

After the initial solution is created, we proceed with the optimization part. The following procedure is repeated until a time limit is reached.

In each iteration, the current solution's neighborhood is explored in a particular order. We always need to provide a feasible packing plan for any new routes and for that we need to call the packing procedure each time we evaluate a neighbor. The objective function used is $d+\alpha \cdot e_W + \beta \cdot e_L$ where d is the total length of all routes, e_W is the total excess weight and e_L is total excess length.

We always accept any move without penalties that results in a better solution than the current one. If we cannot find any, we accept the first move that improves the objective function, or the best move found when none improves over the incumbent. This results in a first improvement policy most of the time. The chosen move is then applied and added to the tabu list, which always contains the last moves that were made.

After the first feasible solution is found, no infeasible solutions are accepted anymore (we enter *feasible mode*). This has the advantage that it is unnecessary to generate packing plans for solutions which increase the total distance, since they cannot improve over the feasible incumbent.

The types of moves between two routes R_0 and R_1 that are considered are:

- Shift move: insert customer $i \in R_0$ at a defined position in R_1 and remove it from R_0
- Swap move: given $i \in R_0$ and $j \in R_1$ swap i and j.
- Crossover move: after defining a splitting point in both R_0 and R_1, R_0 becomes the original first half of R_0 and the original second half of R_1. The same is done to R_1.
- Intra-swap move: swap two customers $i, j \in R_0$.

After a move is accepted, every individual move of a customer contained in the move is declared tabu for the following T iterations (tabu tenure). For instance, after a swap, customer i cannot return to R_0 and customer j cannot return to route R_1 during tabu tenure. The same idea applies to all other types of moves, that is, we forbid reversing individual client moves.

We explore the neighborhood in the following order. First, we evaluate all shift moves possible for all pairs of different routes. Then we evaluate all swap moves and after that all crossover moves, trying all possible insertion points. For a particular type of move all routes are iterated in random order. Intra-swap moves are done selectively: we randomly evaluate n_{iswaps} moves. If we are in *feasible mode* and already have a feasible neighbor at any point, we skip all subsequent moves that do not lead to a better solution with respect to total length (we do not need to call the packing algorithm).

If after all the moves are evaluated the best solution is still infeasible but it is better with respect to total length than that of the current incumbent, we intensify the search for a packing plan, calling the loading procedure again for all routes that currently have penalties.

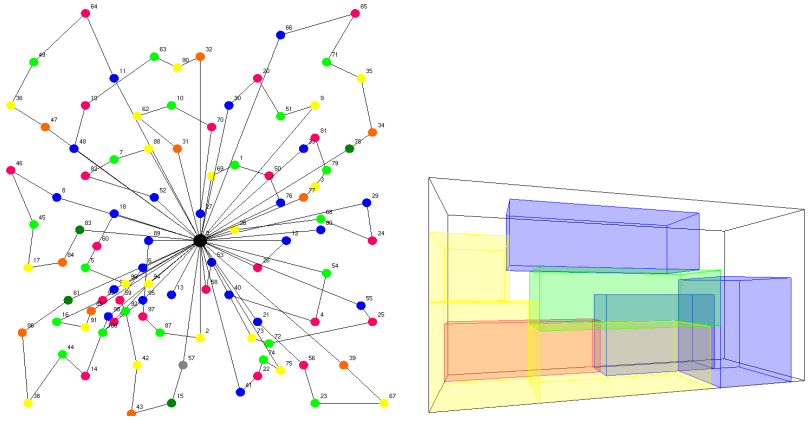
4 Computational Results

Our algorithm was implemented in ANSI C++ and compiled using GCC with optimization flag -O3. All tests were done using a single core of an Intel i7 930 with 2.8GHz and 12GB of RAM. The set of instances used is available at `http://www.or.deis.unibo.it/research.html` and was introduced by Gendreau et al. (2006). Tests for each instance ran until the corresponding *time_limit* was reached. Table 1 shows the settings of all parameters of the algorithm used in the computation experiments.

Table 1: Parameters used in the computational experiments.

parameter	description	value
τ	tries before giving up on packing	10
θ	packing permutations constant	30
n_{iswaps}	number of randomized intra-swaps	$\min(n^2/4, 250)$
T	tabu tenure	$\min(15, n/2)$
α	excess weight penalty	$20\delta/D$, where $\delta = \text{average}(c_{ij})$
β	excess length penalty	$20\delta/W$
$time\_limit$	max time given to the tabu search	$900s$ if $n < 35$, $1800s$ if $35 \leq n < 50$, $3600s$ otherwise

Table 3 presents the results obtained when running our algorithm with all loading constraints imposed. Every solution is feasible. Each instance is identified by its index I together with the



(a) (b)

Figure 2: Routing (a) and packing (b) examples for instance 26.

number of customers n and the total number of boxes M it contains. All results from the literature are given as the average total traveled distance after ten executions of the same method. For all methods we present the average (ttd_{avg}) and the best (ttd_{min}) results found after ten runs with different random seeds, with the exception of Tao and Wang (2010), for which only the average result is published. Best known values for each instance are in bold wherever they appear. The running times are given in seconds and correspond to the average time taken by the algorithms to reach the best solutions, with the exception of Bortfeldt (2010) where it represents the average total running time of the algorithm. Additionally, we provide the average for all columns. For comparison, we show the *gap* between our results and the results from the literature, which corresponds to the average percentage difference of all instances. A positive gap indicates our results are better, on average. The processor used by each method is, in the same order they appear in Table 2: Pentium IV 3.2GHz, Intel Xeon E5430 2.66 GHz, Pentium IV 2.39GHz, Core 2 Duo E8500 3.17GHz and Core i7 930 2.8GHz. Figure 2 gives an example of a routing and a packing produced by our method for instance 26.

The results show that our method is capable of producing solutions of good quality. On average, our algorithm produced better results than all others, ranging from a 0.4% to 1.87% average improvement of the total traveled length. For space reasons, we omitted the results of Gendreau et al. (2006) and Tarantilis et al. (2009). We do note that the average gap in relation to their solutions is 8.16% and 4.79%, respectively. Table 2 compares the number of times each method obtained the best minimum value and the best average value. We can see that with respect to solution quality our method is superior to all others, with the exception of Bortfeldt (2010), which is able to find one more minimum solution. On the other hand, we outperform it on average, as it obtains only six average best solutions, and the average gap is still 1.29% compared to our method. We further observe that our algorithm is particularly good in large instances with 50 or more clients, for which we obtain 6 new best values out of 9 instances. With respect to computational effort, our algorithm is comparable to all the other methods except that of Bortfeldt (2010), which takes significantly less time in a similar setup.

Table 2: Comparison of the number of best solutions obtained by methods from the literature.

	Fuellerer et al. (2009)	Wang et al. (2010)	Tao and Wang (2010)	Bortfeldt (2010)	Our method
Best minimum	2	9	-	13	12
Best average	2	4	8	6	12

Table 3: Performance of our algorithm compared to results from the literature.

			Fuellerer et al. (2009)			Wang et al. (2010)			Tao and Wang (2010)		Bortfeldt (2010)			Our Method		
I	n	M	ttd_{min}	ttd_{avg}	time	ttd_{min}	ttd_{avg}	time	ttd_{avg}	time	ttd_{min}	ttd_{avg}	time	ttd_{min}	ttd_{avg}	time
1	15	32	304.13	305.35	11.2	**301.74**	301.77	193.0	302.02	58.3	302.02	302.02	41.6	304.13	304.84	7.8
2	15	26	**334.96**	**334.96**	0.1	**334.96**	**334.96**	9.3	**334.96**	12.7	**334.96**	**334.96**	0.3	**334.96**	**334.96**	0.5
3	20	37	399.68	409.79	88.5	387.34	387.91	87.0	**381.37**	117.3	388.10	404.34	159.1	**381.37**	384.90	126.4
4	20	36	440.68	440.68	3.9	**437.19**	438.59	91.3	440.68	14.5	**437.19**	437.94	12.4	**437.19**	**437.19**	12.1
5	21	45	450.93	453.19	22.7	**436.48**	440.23	444.7	438.43	163.1	443.61	447.49	170.5	442.21	449.66	138.0
6	21	40	498.32	501.47	17.5	498.32	499.48	125.9	498.32	37.9	**498.16**	501.47	15.3	499.02	501.77	43.0
7	22	46	792.13	797.47	51.4	**767.46**	771.09	394.9	773.01	89.2	769.68	791.03	62.8	769.68	770.22	96.7
8	22	43	820.67	820.67	56.2	**803.98**	805.95	331.0	808.59	113.6	810.89	824.00	98.9	806.25	808.14	337.4
9	25	50	635.50	635.50	15.3	**630.13**	630.90	197.9	634.00	19.5	**630.13**	663.39	11.2	**630.13**	**630.13**	82.1
10	29	62	840.75	841.12	241.2	826.39	832.46	707.0	839.76	293.1	**820.35**	829.31	139.8	822.46	825.86	593.1
11	29	58	818.87	821.04	172.4	**768.25**	781.85	820.9	794.03	273.6	800.52	815.35	118.8	781.17	788.77	643.1
12	30	63	626.37	629.07	46.2	**610.23**	614.78	194.7	616.30	129.3	**610.23**	636.10	12.8	**610.23**	610.43	227.6
13	32	61	2739.80	2739.80	235.4	2697.70	2715.82	859.4	2698.25	382.1	**2679.86**	2701.10	232.9	2693.24	2713.05	971.9
14	32	72	1466.84	1472.26	623.8	1428.99	1456.13	1638.7	1432.44	608.6	**1368.42**	1419.84	312.2	1390.18	1425.18	922.1
15	32	68	1367.58	1405.48	621.0	1352.94	1371.26	1537.3	1377.06	512.8	**1331.27**	1357.01	299.9	1350.85	1370.41	755.5
16	35	63	698.92	698.92	12.8	**698.61**	699.54	46.5	698.92	6.5	**698.61**	704.24	2.4	**698.61**	699.49	225.6
17	40	79	868.59	870.33	11.8	871.63	875.19	731.7	**867.28**	10.3	876.22	969.59	1.7	871.24	871.71	87.7
18	44	94	1255.64	1261.07	2122.2	1227.07	1248.28	1748.8	1236.77	1123.7	**1206.67**	1233.99	315.1	1227.08	1248.09	1045.2
19	50	99	777.18	781.29	614.3	762.47	776.35	1376.9	768.58	572.5	757.04	755.05	419.2	**755.56**	763.44	2147.8
20	71	147	604.28	611.26	3762.3	583.45	593.17	1647.8	603.49	2564.1	585.65	598.23	432.1	**579.18**	586.00	2482.9
21	75	155	1110.09	1124.55	5140.0	1094.78	1121.60	1594.5	1123.37	2043.6	1091.33	1108.40	452.3	**1085.53**	1099.24	2895.0
22	75	146	1194.18	1197.43	2233.6	1170.89	1176.76	1287.7	1179.88	1371.5	1162.11	1175.82	428.6	**1151.48**	1159.85	2056.6
23	75	150	1158.51	1171.77	3693.4	1137.90	1148.02	1091.0	1160.05	1778.1	1144.46	1138.19	430.5	**1119.08**	1128.71	1998.5
24	75	143	1136.80	1148.70	1762.8	1132.05	1144.56	469.8	1141.02	2023.5	**1114.54**	1127.76	413.3	1116.57	1126.55	1232.4
25	100	193	1429.64	1436.32	8619.7	1434.00	1457.09	1582.8	1421.36	3795.7	1398.42	1436.63	463.5	**1389.95**	1405.00	2922.7
26	100	199	1611.78	1616.99	6651.2	1606.85	1616.61	1488.7	1608.16	2550.6	**1590.31**	1630.63	436.7	1594.45	1612.31	1586.5
27	100	198	1560.70	1573.50	10325.8	1551.68	1574.23	1440.1	1555.34	4848.3	**1528.43**	1541.99	441.6	1537.10	1560.06	1481.4
avg			960.87	966.67	1746.54	946.43	956.10	820.02	953.09	944.90	939.97	958.74	219.40	939.96	948.74	930.40
gap				1.87%			0.57%		0.40%			1.29%			0%	

5 Conclusions

The 3L-CVRP is an interesting and challenging problem due both to its theoretical complexity and its direct relation to real-world applications. In this paper we presented a solution based a tabu search algorithm for the routing and a novel packing heuristic. We found the packing to be extremely important for obtaining good final results, and the proposed randomized packing algorithm and the representation of the 3D loading space are largely responsible for the quality of our solutions. Besides the packing, the first-improvement evaluation of the routing's neighborhood also strongly contributed to the results.

Through computational experiments we were able to show that our method obtains equal or better results than those from the literature. We have been able to improve the best known value of six instances. Our method is robust in the sense that on average it dominates all other methods and obtains the best results for large instances.

Acknowledgments

Marco A. Wisniewski received support from CNPq under project no. 139292/2010-1.

References

Baker, B.S., Coffman Jr., E.G. and Rivest, R.L. (1980). Orthogonal packing in two dimensions, *SIAM Journal on Computing* **9**: 846–855.

Bortfeldt, A. (2010). A Hybrid Algorithm for the Capacitated Vehicle Routing Problem with Three-Dimensional Loading Constraints, *Diskussionsbeiträge der Fakultät für Wirtschaftswissenschaft der FernUniversität in Hagen* **460**.

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

de Araújo, O. C. B. (2006). *Problemas de Corte e Empacotamento Tridimensional e Integração com Roteamento de Veículos*, PhD thesis, Faculdade de Engenharia Elétrica e de Computação - UNICAMP, Campinas, Brasil.

Fuellerer, G., Doerner, K.F., Hartl, R. and Iori, M. (2010). Metaheuristics for vehicle routing problems with three-dimensional loading constraints, *Eur J Oper Res* **201**: 751–759.

Gendreau, M., Iori, M., Laporte, G. and Martello, S. (2006). A tabu search algorithm for a routing and container loading problem, *Transportation Science* **40**(3): 342–350.

Iori, M., Salazar Gonzalez. J.J. and Vigo, D. (2007). An exact approach for the symmetric capacitated vehicle routing problem with two dimensional loading constraints, *Transportation Science* **41**(2): 253–264.

Martello, S., Pisinger, D. and Vigo, D.. (2000). The three-dimensional bin packing problem, *Operations Research* **48**: 256–267.

Ngoi, B. K. A., Tay, M. and Chua, E. (1994). Applying spatial representation techniques to the container packing problem, *International Journal of Production Research* **32**: 111–123.

Portal, G., Rocco, R., Ritt, M. and Buriol, L.S. (2009) Uma busca tabu aplicada ao problema de roteamento com restrições de empacotamento tridimensionais, *Anais do XLI Simpósio Brasileiro de Pesquisa Operacional*.

Tao, Y. and Wang, F. (2010). A new packing heuristic based algorithm for Vehicle Routing Problem with Three-dimensional Loading constraints, *IEEE Conference on Automation Science and Engineering (CASE)*, pp. 972–977.

Tarantilis, C.D., Zachariadis, E.E. and Kiranoudis, C.T. (2009). A hybrid metaheuristic algorithm for the integrated vehicle routing and threedimensional container-loading problem, *IEEE Transactions on Intelligent Transportation Systems* **10**: 255–271.

Toth, P. and Vigo, D. (2002). *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, PA.

Wang, L., Guo, S., Chen, S., Zhu, W. and Lim, A. (2010). Two Natural Heuristics for 3D Packing with Practical Loading Constraints, *PRICAI 2010: Trends in Artificial Intelligence*, pp. 256–267.