A Branch-and-Cut Approach for the Vehicle Routing Problem with Two-dimensional Loading Constraints

Bruno L. P. de Azevedo,
Instituto de Computação, IC, UNICAMP,
13084-971, Campinas, SP
e-mail: bruno.pires@students.ic.unicamp.br

Pedro H. Hokama
Instituto de Computação, IC, UNICAMP,
13084-971, Campinas, SP
e-mail: pedro.hokama@students.ic.unicamp.br

Flávio K. Miyazawa,
Instituto de Computação, IC, UNICAMP,
13084-971, Campinas, SP
e-mail: fkm@ic.unicamp.br

Eduardo C. Xavier,
Instituto de Computação, IC, UNICAMP,
13084-971, Campinas, SP
e-mail: ecx@ic.unicamp.br

RESUMO

Apresentamos um algoritmo branch-and-cut para o Problema de Roteamento de Veículos com Restrições Bidimensionais (2L-CVRP). Este é uma combinação do Problema de Roteamento de Veículos Capacitado (CVRP) e do Problema de Empacotamento em Placas. Propomos uma estrategia com ênfase na inserção de cortes de familias conhecidas de desigualdades válidas para o CVRP. Para verificar a viabilidade dos empacotamentos foram usados algoritmos heurísticos e exatos. Consideramos a variantes irrestrita e a sequencial, na qual o descarregamento de itens não pode ser bloqueado por itens de outros clientes, e também a variante sem restrições.

O algoritmo apresentado difere do algoritmo exato existente na literatura através do uso de diferentes familias de desigualdades para o CVRP e de diferentes estratégias de inserção. Resultados computacionais indicaram uma redução satisfatória nos tempos de execução para diversas instâncias na literatura.

PALAVRAS CHAVE. Roteamento de veículos. Empacotamento bidimensional. Branch-and-cut. Otimização Combinatória.

ABSTRACT

In this paper, we present a branch-and-cut algorithm for the Vehicle Routing Problem with Two-dimensional Loading Constraints (2L-CVRP). The 2L-CVRP is a combination of the Capacitated Vehicle Routing Problem (CVRP) and the Two-dimensional

Bin Packing Problem. We propose a strategy based on emphasizing cut generation using known families of valid inequalities for the CVRP. Checking packing feasibility is performed by combining heuristic and exact algorithms. We consider the unrestricted case and the sequential variant, in which unloading items can not be blocked by another customer's items.

The presented algorithm differs from the known exact algorithm in the literature by using different families of CVRP inequalities and different insertion strategies. Computational results showed a satisfactory decrease on the execution time for several instances from the literature.
KEYWORDS. Vehicle routing. Two-dimensional packing. Branch-and-cut. Combinatorial Optimization.

1 Introduction

The Two-dimensional Capacitated Vehicle Routing Problem with Loading Constraints (2L-CVRP) is a special case of the well-known Capacitated Vehicle Routing Problem (CVRP) where the customer's demands are composed of a certain number of rectangular two-dimensional weighted items. The problem is strongly NP-Hard, as it consists of solving the CVRP and a loading problem analogous to the Two-dimensional Bin Packing Problem (2BPP).

There are several practical applications for the 2L-CVRP, given that in many real-world situations, items cannot be stacked on top of each other because of their fragility, height or shape. Also, rectangular shaped structures (pallets) are often used to support goods in a stable way. Many types of pallets have unidirectional bases, allowing it to be lifted from only one direction. Consequently, for this paper we assume items to have a fixed orientation.

The 2L-CVRP was first addressed by Iori et al. (2007), who proposed an exact method using a Branch-and-Cut approach. Later, a Tabu Search heuristic was presented by Gendreau et al. (2008), followed by an Ant Colony Optimization presented by Fuellerer et al. (2009) and a Guided Tabu Search heuristic presented by Zachariadis et al. (2009). Two variants of the problem have been explored, the Sequential and Unrestricted cases. For the Sequential variant, the loading pattern must obey a certain ordering of items, i.e. when unloading items of a customer, these must not be blocked by items belonging to other customers. The Unrestricted 2L-CVRP allows re-arrangement of items at the customers site, hence no ordering is needed. The Sequential constraint occurs in practice because of item characteristics. Items can be fragile or very heavy, so that making re-arrangements during unloading becomes hard to perform or time consuming.

In this paper we present an exact algorithm based on a branch-and-cut approach for the 2L-CVRP. Five families of inequalities for the CVRP are used in our routing strategy. Determining packing feasibility is done through heuristic and exact algorithms. To help decreasing the time spent on the loading problem, we used a hash table to store known routes.

The only exact methodology for the 2L-CVRP was presented by Iori et al. (2007), taking in consideration the Sequential variant. Noticing that for many instances this approach spent a considerable time solving the routing problem, we propose a strategy based on emphasizing cut generation using known families of valid inequalities for the CVRP. We then examine how our approach compares to the method by Iori et al. (2007). Our algorithm exhibited satisfatory results. We also relax the sequential constraint and analyze the results found. Finally, we investigate a simple way to sub-

stantially improve the algorithm performance and suggest work to be accomplished in the future.

This article is organized as follows: section 2 describes the 2L-CVRP. Section 3 presents the integer programming formulation used to model the problem. The Branch-and-Cut algorithm is described in detail in section 4, including the routing and packing strategies. Section 5 shows an analysis of the computacional results. Concluding remarks are done in section 6.

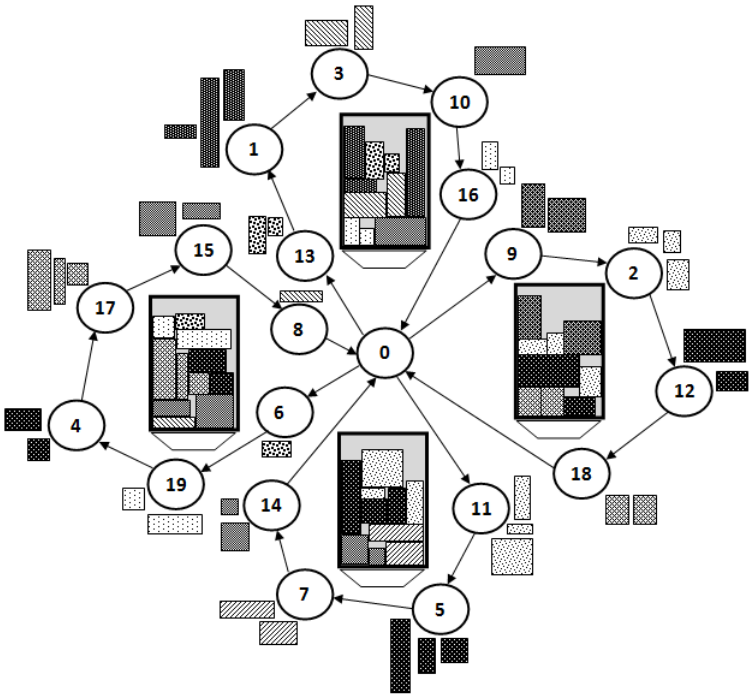

Figure 1: Example of the 2L-CVRP

2 Problem Description

Given a complete undirected graph $G = (V, E)$, V is a set of $n + 1$ vertices corresponding to a depot (vertex 0) and n customers (vertices $1, \ldots, n$). E is a set of edges, where for each $e \in E$ there is an associated cost $c_e \in \mathbb{Q}^+$. There is a set of K identical vehicles, each having a weight capacity of D and a container of width W and length L to carry the customer's items. Let $V_+ = V \setminus \{0\}$, each customer $i \in V_+$ demands a set of items of total weight d_i. Let B be the total set of items where each $b \in B$ is a tuple $b = (w_b, l_b)$, w_b and l_b being its width and length, respectively and $B_i \subseteq B$ is the set of items of customer i.

A packing $\mathcal{P}$ of a list of items $P \subseteq B$ in a container (W, L) is a function $\mathcal{P} : P \to [0, W) \times [0, L)$ such that:

(i) The items in P must not be rotated (fixed orientation).

(ii) The packing must be orthogonal, i.e. the edges of the (rectangular) items must be parallel or perpendicular to the container's edges.

(iii) Items of P must be packed within the container's boundaries. That is, if $\mathcal{P}(b) = (\mathcal{P}^w(b), \mathcal{P}^l(b))$, for each $b \in P$, then

$$\mathcal{P}^w(b) + w_b \leq W \text{ and } \mathcal{P}^l(b) + l_b \leq L. \quad (1)$$

(iv) Items must not overlap. That is, if $\mathcal{R}(b) = [\mathcal{P}^w(b), \mathcal{P}^w(b) + w_b) \times [\mathcal{P}^l(b), \mathcal{P}^l(b) + l_b)$ then

$$\mathcal{R}(b') \cap \mathcal{R}(b'') = \emptyset \text{ for all } b', b'' \in P. \tag{2}$$

A feasible route C is a cycle in G that contains the depot and satisfy the following conditions:

(v) The total weight of all customers in C must not exceed the vehicle load capacity. That is, if V_C is the set of customers in C then $\sum_{i \in V_C} d_i \leq D$.
(vi) There is a packing $\mathcal{P}_C$ of the items in C.

The 2L-CVRP consists in finding a set of feasible routes $\mathcal{C} = \{C_1, \ldots, C_K\}$, such as to minimize the cost of routing $c(\mathcal{C}) = \sum_{i=1}^{k} \sum_{e \in C_i} c_e$.

Since it could be hard or time consuming to make re-arrangements when unloading, it may also be desirable to have the following additional condition:

(vii) Items belonging to an unloading customer are not blocked by another customer's items. More precisely, given a route C with sequence of vertices $(0, i_1, \ldots, i_t, 0)$, a packing $\mathcal{P}$ of $P = B_{i_1} \cup \ldots \cup B_{i_t}$ obeys the sequence of C in the direction l if it satifies

$$\mathcal{R}^l(b') \cap \mathcal{R}(b'') = \emptyset \quad \text{for all } b' \in B_{i_r} \text{ and } b'' \in B_{i_s} \text{ where } 1 \leq r < s \leq t, \tag{3}$$

where $\mathcal{R}^l(b) = [\mathcal{P}^w(b), \mathcal{P}^w(b) + w_b) \times [\mathcal{P}^l(b), L)$. The Sequential 2L-CVRP is a special case of the 2L-CVRP where all routes must satisfy condition *(vii)*. Figure 2 shows a route and two packings for the sequential and unrestricted versions.

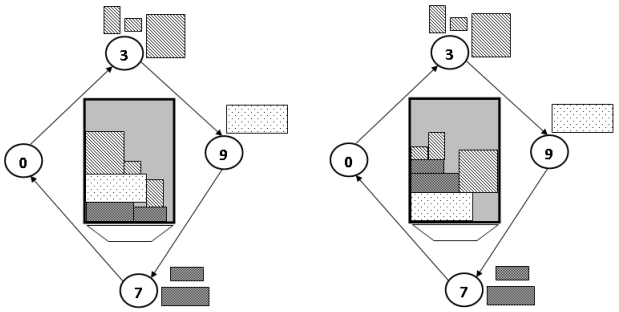

Figure 2: Sequential and Unrestricted loading, respectively.

3 Formulation

The problem was modelled using the so-called two-index formulation for the CVRP. Let x_e be a flow variable equal to one if a vehicle travels along the edge, zero otherwise. Given a subset of customers $S \subseteq V_+$, $d(S)$ is the total weight of all items of the customers in S, i.e., $d(S) = \sum_{i \in S} d_i$. Denote by A the area of the container and $a(S)$ the total area of the items of the customers in S, i.e, $a(S) = \sum_{i \in S} \sum_{b \in B_i} w_b l_b$. Let $\delta(S)$ be the set of edges in G with exactly one vertex in S and let $\gamma(S)$ be the set of edges in G with both vertices in S. Let $r(S)$ be the minimum number of vehicles needed to supply the demand of S and K be the number of vehicles available. The integer programming formulation is:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{e \in E} c_e x_e & & & \\
\text{s.a.} \quad & \sum_{e \in \delta(\{i\})} x_e = 2 & \forall i \in V_+ & & (1) \\
& \sum_{e \in \delta(S)} x_e \geq 2r(S) & \forall S \subseteq V_+, \ |S| \geq 2 & & (2) \\
& \sum_{e \in \delta(\{0\})} x_e = 2K & & & (3) \\
& x_e \in \{0,1\} & \forall e \in E. & & (4)
\end{aligned}
\tag{4}
$$

Constraints (1) ensure that each customer is visited exactly once. Constraints (2), the Capacity Inequalities, impose connectivity and capacity restrictions. Note that to calculate $r(S)$ is NP-Hard, since it is equivalent to finding the optimal solution of the two-dimensional Bin Packing Problem, given the vehicle container and the customers items. But if $r(S)$ is replaced by $k(S) = max\{\lceil d(S)/D \rceil, \lceil a(S)/A \rceil\}$, a valid lower bound is achieved. These are known as Rounded Capacity Inequalities (see Naddef and Rinaldi (2002)). By using the degree equations (1), the Rounded Capacity Inequalities can be re-written in the alternative form:

$$\sum_{e \in \gamma(S)} x_e \leq |S| - k(S). \tag{5}$$

Constraints (3) ensure that exactly K vehicles are used and constraints (4) impose that the variables must be binary. Following Iori et al. (2007), we do not allow routes with only one customer.

4 Branch-and-Cut Algorithm

The 2L-CVRP consists of a routing and a two-dimensional packing problem, therefore routing and packing strategies are needed.

Our routing strategy consists of separating Capacity Inequalities to ensure connectivity and capacity constraints, and using other families of inequalities to try reducing the feasible region. Whenever a routing solution is found, the packing strategy is put into action. It is also used when no more capacity cuts can be found at the current node.

4.1 Separation Routines

There are too many Capacity Inequalities, an exponential number, to be added a priori to the formulation. Constraints (1), (3) and the bounds give us a relaxation that can be easily solved, generating a weak lower bound that can be strenghtened by adding cutting planes. For this reason, the following separation routines were used: Capacity Inequalities, Framed Capacity Inequalities, Multistar Inequalities, 2-Edges Extended Hypotour Inequalities and Strenghtened Comb Inequalities. Observe that we have adapted separation for Capacity Inequalities to take into account not only the capacity of the vehicle, but also its container's area. Details on the cuts used can be found in Lysgaard et al. (2003) and Naddef and Rinaldi (2002).

Separating Capacity Inequalities is NP-Hard (see Naddef and Rinaldi (2002)), consequently separation is done on Rounded Capacity Inequalities, obtained by replacing $r(S)$ by $k(S)$ on (2).

To separate different CVRP families of inequalities we used code disponibilized by Lysgaard et al. (2003).

4.2 Two-dimensional Loading problem (2DLP)

To check if a set of rectangular items can be packed in a container, we used an adaptation of the OneBin algorithm presented in Martello et al. (2000). The algorithm was slightly modified to consider the customer's unloading order when generating a feasible packing.

OneBin works as follows: let I be the set of packed items, $C(I)$ be the set of corner points defined by I, and let $A(I)$ be the area of this envelope. Denote F as the area of the best packing found, B as the area of the container and each item has a area v_i. Let J be the set of items not packed yet. At each iteration $C(I)$ and $A(I)$ are computed. Each item $j \in J$ is assigned for each $c \in C(I)$ and OneBin is called recursively. If necessary, F is updated. Whenever $\sum_{i \in I} v_i + (B - A(I)) \leq F$ happens, backtracking occurs. We used OneBin source code disponibilized by Martello et al. (2000).

To deal with the sequential constraint, we have done a simple modification on OneBin: when an assigned item does not fit, it now checks if there is an violation of the order constraint by this item. If true, it backtracks. We also modified the initial ordering of items, now they are ordered first by its customer's visiting order and then by nondecreasing area.

Besides the exact approach, a fast heuristic was employed to help decrease the time spent solving the 2DLP. OneBin is called only when a feasible packing can not be found by a modified Bottom Left Decreasing Width heuristic (BLDW). For a description of the BLDW algorithm see Baker et al. (1980). The sequential constraint is considered by this heuristic.

To reduce the computational effort, known routes are stored in a hash table. A route is stored whether it is feasible or not, therefore the same route will not be packed twice. The algorithm takes two different behaviors depending on which variant of the 2L-CVRP is being solved: for the Unrestricted 2L-CVRP, an unique hash is generated depending on which customers are covered by a route. But for the sequential variant, two routes could cover the same set of customers and have different hashes. Two routes will have the same hash, excluding collisions, only if they serve the same customers and have an identical visiting order.

4.3 Routing Strategy

To allow greater chances of finding more cutting planes at each iteration at the root node, we decided for two cyclical approaches based on a separation strategy proposed by Lysgaard et al. (2003). At each iteration, the following separation routines can be called: Rounded Capacity Inequalities, Framed Capacity Inequalities, Multistar Inequalities, Strenghtened Comb Inequalities and 2-Edges Extended Hypotour Inequalities. For each separation attempt, every inequality that was found is added to the LP.

Initially, we try to separate Rounded Capacity Inequalities. If there is at least one cut violated by less than a certain limit, we attempt Framed Capacity separation. If no Framed Capacity Inequality is found, two different approaches are used: if the current iteration is a multiple of five, we try to separate Multistar and Strenghtened Comb Inequalities. Otherwise, a circular strategy is put into action: separation of Multistar, Strenghtened Comb and Hypotour occurs every third iteration, e.g., if Multistar Inequalities are found, re-optimization occurs and we will not try to separate this

family again until both Strenghtened Comb and Hypotour Inequalities are found. The order used was Multistar, followed by Strenghtened Comb and then Hypotour Inequalities. It should be noted that separation is pursued only if we found at least one Rounded Capacity Inequality violated by a predefined limit. This limit is different for each family.

After experimenting extensively we found it more effective to treat the non-root nodes of the branching tree differently to the root node. First we call the routine to separate Capacity Inequalities, then we try to separate Framed Capacity Inequalities, followed by Multistar Inequalities and Strenghtened Comb Inequalities. Unlike our approach for the root node, these procedures are called without any conditions (limits). For each subsequent iteration, only Capacity separation is attempted.

4.4 Packing Strategy

For each route found, we perform the following steps: first, we check if it can be found in the hash table, if so, it means that this route's 2DLP was solved previously and its feasibility is known. Otherwise, we try our packing heuristic. The heuristic attempts to pack the route items in both customer sequences of the route. If a feasible packing is found, we update the hash table. If it fails, the exact algorithm is called and the hash table is updated with the results found. We generate a cut equivalent to the route if there is not a feasible packing.

Observe that this strategy is used not only when a feasible integer routing solution is found, but also when no more capacity cuts can be found at the current node, i.e. when capacity separation fails, the solution's integer routes are checked for packing feasibility and a cut is generated for any unfeasible route.

4.5 Branching

We used the standard branching framework provided by the Xpress-Optimizer. For every node, branching (if possible) only occurs when no more Capacity Cuts can be found, otherwise re-optimization occurs.

5 Computational Results

The proposed algorithm was coded in C and tested on a subset of the instances available at `http://www.or.deis.unibo.it/`. The chosen test set was composed of instances in which the approach by Iori et al. (2007) took at least 10 seconds to solve. The choice of the instances was guided mostly by looking for ones that presented a harder routing problem, but the chosen set ended being somewhat varied. The experiments were run on a single 2.40 GHz Core of a Intel Quad Core 2.40 GHz. A CPU time limit of 14400 seconds was imposed for the entire algorithm. The IP model was solved using Xpress-Optimizer v19.00.00.

Following the standards set in Iori (2004) and Gendreau et al. (2004), the Euclidian distances between vertices were truncated and the number of vehicles was set to the minimum possible.

Our root node separation strategy tries to separate different families of cuts only if there is at least one Rounded Capacity Inequality violated by less than a certain limit. For our experiments, these limits were: 0.2 for Framed Capacity Inequalities, 0.05 for Multistar Inequalities, 0.1 for Strenghtened Comb Inequalities and 0.1 for the 2-Edges Extended Hypotour Inequalities. These

limits were established during a tuning session, using a subset of the test instances. Notice that these limits ended being the same as the ones used by Lysgaard et al. (2003).

Besides the five cuts listed, our strategy also generates a certain number of Gomory cuts at the root node and at each subsequent node of the branching tree.

Comparison to other works is difficult, due to hardware differences, implementation details, the choice of the LP solver and even the fact that different versions of a same solver could generate very distinct results. Therefore, the following comparisons are made with reservations.

We summarize our algorithm performance for the sequential variant in Table 1 and compare it to the algorithm presented in Iori et al. (2007), since it is the only exact method for the 2L-CVRP we are aware of. Name and Class identify the instance. Refer to Iori et al. (2007) and Gendreau et al. (2008) for a thorough explanation of how the instances were created. Instance details are found in the next two columns, with n and L being the instance's number of vertices and items respectively. The next column, K, shows the number of vehicles used to solve the instance. Data on the loading effort is detailed on the following three columns: P_{he} represent the number of feasible packings found by the heuristic method and P_e, by the branch-and-bound. P_{ha} gives the number of times the 2DLP did not need to be solved because it could be found in the hash table. i.e. during the 2DLP phase, anytime the algorithm could find a known route this value was updated. C_{bc} shows the number of cuts found by the method in Iori et al. (2007) and C_{seq} gives the number of cuts found by the separation routines used by our algorithm. Nd_{seq} represents the number of explored nodes of the branching tree and z_{seq} gives the best integer solution obtained. Finally, the computing time (in seconds) used by Iori et al. (2007) branch-and-cut and by our algorithm are given by T_{bc} and T_{seq} respectively. Observe that Iori et al. (2007) used a Intel Pentium IV 3.0 GHz to obtain T_{bc}.

Our algorithm performance proved to be satisfatory, having a lower CPU time in 18 of the 22 instances tested. On average, the CPU time difference was considerable.

Instance $E016\text{-}03m\text{-}5$ proved to be an anomaly, making our algorithm spend most of its time on the 2DLP. For instance $E030\text{-}03g\text{-}1$, our algorithm presented very strong results, but it can not be atributed to the cut strategy or the inequalities used, as it can be seen in table 2.

In instance $E026\text{-}08m\text{-}3$, the presented method outperformed the previous approach substantially. Carefully examining the data reveals the efficiency of the inequalities used, helping our algorithm to find the optimum solution after exploring only 35 nodes of the branch and bound tree. For instances $E026\text{-}08m\text{-}5$ and $E036\text{-}11h\text{-}3$, the results also showed a considerable decrease on the execution time.

Our strategy also brought benefits to instances where the previous approach spent most of its time on loading-related procedures (see Iori et al. (2007) for details), such as instance $E022\text{-}04g\text{-}4$, where our algorithm showed a substantial decrease on the CPU time. This particular result is attributed to the strength of our packing heuristic.

As expected, our more comprehensive cut strategy ended obtaining more cuts for almost all instances (the single exception being instance $E030\text{-}03g$). In average, our algorithm generated 24.79 times more cuts than Iori et al. (2007) method. Unfortunately we do not know their number of explored nodes, as it could allow for a better analysis.

For some instances, the number of inserted cuts was much higher than the previous exact approach. The increased computational effort, as a result of using (and generating) a considerable number of cuts, was surpassed by its benefits. For example, for all classes of the instance $E036\text{-}11h$, an average of 87.62 times more cuts were used by our method and the results ended being satisfactory.

Table 1: Performance of the branch-and-cut algorithm for the Sequential 2L-CVRP

Name	Class	n	L	K	P_{he}	P_e	P_{ha}	C_{bc}	C_{seq}	Nd_{seq}	z_{seq}	T_{bc}	T_{seq}
$E016$-$03m$	2	16	24	3	32	39	1224	702	15095	1797	285	15.53	11.56
	3	16	31	3	35	30	880	601	6764	1031	280	21.02	48.23
	5	16	45	4	5	0	10	4	5	2	279	40.03	1290.80
$E016$-$05m$	4	16	40	5	18	3	129	212	2693	139	336	19.86	7.05
$E021$-$06m$	3	21	43	6	17	11	301	587	4170	211	432	11.30	15.48
	5	21	62	6	5	1	6	154	243	9	423	46.74	0.35
$E022$-$04g$	4	22	41	4	4	5	27	383	391	13	377	35.42	3.11
	5	22	57	5	4	1	5	39	47	1	389	1867.38	0.11
$E022$-$06m$	2	22	33	6	22	5	283	1010	1928	123	491	26.52	1.45
	3	22	40	6	24	22	1028	1081	7478	513	496	36.84	5.83
$E023$-$03g$	5	23	55	6	16	7	453	20	55	65	742	787.33	917.16
$E026$-$08m$	1	26	25	8	8	0	8	727	901	9	609	19.80	1.32
	2	26	40	8	9	7	149	1199	4847	123	612	57.55	3.72
	3	26	61	8	7	2	93	1244	1765	35	615	164.22	2.10
	4	26	63	8	38	11	1654	1440	42491	841	626	258.20	72.94
	5	26	91	8	8	0	8	527	901	9	609	43.17	1.30
$E030$-$03g$	1	30	29	3	17	0	45	6207	1435	131	524	54635.61	1.02
$E036$-$11h$	1	36	35	11	48	0	485	1962	118124	975	682	2114.28	164.70
	2	36	56	11	51	19	1173	1857	153477	1309	682	1426.03	329.18
	3	36	74	11	19	11	152	1589	56633	431	682	993.34	56.65
	4	36	93	11	77	47	3049	3413	639759	5543	691	7946.74	4537.89
	5	36	114	11	47	1	485	1637	118124	975	682	1139.86	164.69

We also compared the behavior of the algorithm, restricting the use of the CVRP cutting planes. We consider two cases, one when only Rounded Capacity Inequalities (RCI) are used and the other when all five families of cuts are used. These results are sumarized in table 2. Column OPT shows the value of the optimum solution. Columns LB_{rci} and LB_{seq} give the lower bound found at the root node by executing our algorithm when only using RCI and using all five inequalities, respectively. The number of explored nodes of the branching tree when using only RCI is given by Nd_{rci}. The computing time observed by using only Rounded Capacity Inequalities to solve the instances is given by T_{rci}. Columns Nd_{seq} and T_{seq} are the same as in table 1.

One of the main time consuming points of the algorithm is clearly the packing routine. Considering that it is called for each possible route, it is important to reduce the number of such verifications. The use of more cutting planes helped to reduce the size of the branch-and-cut tree, and consequently, the number of calls to the packing routines.

For the larger instances $E036$-$11h$, the number of explored nodes of the branch-and-cut tree reduced substantially, about 49%. On average, the number of nodes reduced by 40.22%.

In general, the cut intensive approach reduced the time to solve several instances, mainly for larger instances. The time reduction impact is big for instance $E026$-$08m$-5 (that was not solved using only RCI inequalities) and for instance $E026$-$08m$-3 and for all classes of instance $E036$-$11h$ (some with an improvement of an order of magnitude).

Table 3 presents the results found by running our test set with the sequential constraint relaxed. Columns named with the unr suffix represent the observed results. Columns C_{unr} and Nd_{unr} give the number of inserted cuts and the number of explored branch and bound nodes. The best solution found is represented by z_{unr} and T_{unr} shows the elapsed CPU time elapsed.

The unrestricted version showed a small decrease of cost (only 0.58% on average) for our test instances. It seems to be a singular characteristic of our subset of instances, as Fuellerer et al. (2009)

Table 2: Comparison of RCI and all inequalities

Name	Class	n	L	K	OPT	LB_{rci}	LB_{seq}	Nd_{rci}	Nd_{seq}	T_{rci}	T_{seq}
$E016\text{-}03m$	2	16	24	3	285	262	262	3225	1797	18.89	11.56
	3	16	31	3	280	262	262	2491	1031	72.02	48.23
	5	16	45	4	279	279	279	2	2	1298.24	1290.80
$E016\text{-}05m$	4	16	40	5	336	318	319	411	139	10.08	7.05
$E021\text{-}06m$	3	21	43	6	432	420	420	103	211	1.58	15.48
	5	21	62	6	423	420	420	9	9	0.36	0.35
$E022\text{-}04g$	4	22	41	4	377	369	369	15	13	3.11	3.11
	5	22	57	5	389	389	389	1	1	0.12	0.11
$E022\text{-}06m$	2	22	33	6	491	475	478	259	123	1.24	1.45
	3	22	40	6	496	475	478	759	513	7.54	5.83
$E023\text{-}03g$	5	23	55	6	742	722	722	-	65	14400.00*	917.16
$E026\text{-}08m$	1	26	25	8	609	603	604	91	9	2.07	1.32
	2	26	40	8	612	603	604	219	123	3.52	3.72
	3	26	61	8	615	603	604	395	35	78.48	2.10
	4	26	63	8	626	603	604	1827	841	134.16	72.94
	5	26	91	8	609	603	604	189	9	2630.60	1.30
$E030\text{-}03g$	1	30	29	3	524	499	499	335	131	1.04	1.02
$E036\text{-}11h$	1	36	35	11	682	647	660	1991	975	487.53	164.70
	2	36	56	11	682	647	660	1991	1309	487.62	329.18
	3	36	74	11	682	647	660	1917	431	586.96	56.65
	4	36	93	11	691	647	660	8298	5543	14362.62	4537.89
	5	36	114	11	682	647	660	1991	975	488.76	164.69

obtained a 3% decrease for the entire test set. A reasonable reduction in CPU time is explained by the easier 2DLP. It can be observed by analyzing the C_{unr} column, as the number of cuts generated was much smaller then the sequential variant. A primal heuristic was not implemented on the

Table 3: Comparison of the Sequential and Unrestricted 2L-CVRP

Name	Class	n	L	K	C_{seq}	C_{unr}	Nd_{seq}	Nd_{unr}	z_{seq}	z_{unr}	T_{seq}	T_{unr}
$E016\text{-}03m$	2	16	24	3	15095	1971	1797	341	285	273	11.56	2.19
	3	16	31	3	6764	2868	1031	437	280	279	48.23	17.60
	5	16	45	4	5	5	2	2	279	279	1290.80	6697.67
$E016\text{-}05m$	4	16	40	5	2693	284	139	11	336	329	7.05	0.26
$E021\text{-}06m$	3	21	43	6	4170	243	211	9	432	423	15.48	0.18
	5	21	62	6	243	243	9	9	423	423	0.35	0.09
$E022\text{-}04g$	4	22	41	4	391	391	13	13	377	377	3.11	1.45
	5	22	57	5	47	47	1	1	389	389	0.11	0.03
$E022\text{-}06m$	2	22	33	6	1928	1381	123	59	491	488	1.45	0.58
	3	22	40	6	7478	1397	513	61	496	489	5.83	3.25
$E023\text{-}03g$	5	23	55	6	55	55	65	65	742	742	917.16	592.10
$E026\text{-}08m$	1	26	25	8	901	901	9	9	609	609	1.32	1.30
	2	26	40	8	4847	901	123	9	612	609	3.72	1.31
	3	26	61	8	1765	901	35	9	615	609	2.10	1.32
	4	26	63	8	42491	44982	841	915	626	626	72.94	113.49
	5	26	91	8	901	901	9	9	609	609	1.30	1.30
$E030\text{-}03g$	1	30	29	3	1435	1435	131	131	524	524	1.02	0.98
$E036\text{-}11h$	1	36	35	11	118124	118124	975	975	682	682	164.70	164.76
	2	36	56	11	153477	83827	1309	853	682	682	329.18	138.05
	3	36	74	11	56633	56633	431	431	682	682	56.65	56.71
	4	36	93	11	639759	226784	5543	1807	691	687	4537.89	954.91
	5	36	114	11	118124	118124	975	975	682	682	164.69	164.75

algorithm presented in this paper. Finding a good upper bound by means of a heuristic method could help to decrease the CPU time spent. To explore possible benefits of having a primal heuristic, we

decided for an interesting approach as done in Lysgaard et al. (2003): having optimal solutions for all of our test set, we ran the algorithm with the optimum as a primal bound. Table 4 summarizes the results found.

The columns for table 4 are defined as follows: C_{primal}, Nd_{primal} and T_{primal} represent the number of cuts found, the number of nodes explored and the CPU time respectively. For these three columns, the algorithm was executed having received the optimum as upper bound. Column $\%N_{var}$ gives the variation between N_{seq} and Nd_{primal}. The variation between T_{seq} and T_{primal} is given by $\%T_{var}$.

The algorithm showed a substantial performance improvement for most cases, having a negligible gain or loss only for instances already very easy to solve. The total solution time was decreased by 61.13% and the total number of explored nodes decreased by 61.97%. Consequently, the number of cuts found also decreased. These results are not surprising, but they show how we could improve our algorithm performance considerably.

Table 4: Performance with OPT as upper bound

Name	Class	C_{seq}	C_{primal}	Nd_{seq}	Nd_{primal}	$\%N_{var}$	T_{seq}	T_{primal}	$\%T_{var}$
E016-03m	2	15095	4650	1797	511	-71.56	11.56	3.08	-73.36
	3	6764	3393	1031	273	-73.52	48.23	11.75	-75.64
	5	5	2	2	1	-50.00	1290.80	1289.56	-0.10
E016-05m	4	2693	1065	139	31	-77.70	7.05	6.44	-8.65
E021-06m	3	4170	2116	211	97	-54.03	15.48	1.99	-87.14
	5	243	147	9	1	-88.89	0.35	0.38	8.57
E022-04g	4	391	292	13	11	-15.38	3.11	3.07	-1.29
	5	47	47	1	1	-00.00	0.11	0.14	27.27
E022-06m	2	1928	1368	123	61	-50.41	1.45	0.63	-56.55
	3	7478	4790	513	327	-36.26	5.83	3.69	-36.71
E023-03g	5	55	74	65	53	-18.46	917.16	800.25	-12.75
E026-08m	1	901	616	9	1	-88.89	1.32	1.30	-1.52
	2	4847	744	123	3	-97.56	3.72	3.28	-11.83
	3	1765	2096	35	25	-28.57	2.10	2.20	4.76
	4	42491	23748	841	471	-44.00	72.94	20.94	-71.29
	5	901	616	9	1	-88.89	1.30	1.30	-0.00
E030-03g	1	1435	523	131	17	-87.02	1.02	0.32	-68.63
E036-11h	1	118124	42179	975	273	-72.00	164.70	43.03	-73.87
	2	153477	42179	1309	273	-79.14	329.18	42.91	-86.96
	3	56633	42179	431	273	-36.66	56.65	42.84	-24.38
	4	639759	259638	5543	2455	-55.71	4537.89	646.46	-85.75
	5	118124	42179	975	273	-72.00	164.69	42.85	-73.98

6 Conclusions and Future work

In this paper we presented a branch-and-cut algorithm for both variants of the 2L-CVRP. So far, the only exact methodology for the 2L-CVRP was presented by Iori et al. (2007), for the sequential case.

We noticed that for many instances, the previous exact approach spent a considerable time solving the routing problem. Thus, our strategy emphasizes cut generation using five different families of inequalities for the CVRP.

Although the number of inserted cuts was much higher than the previous exact approach, the increased computational effort was compensated by the decrease in the overall time needed to solve

most of the instances. The overall performance was satisfactory and for many instances the gain was substantial.

As future work, we intend to implement a primal heuristic to help finding good upper bounds and inserting new families of valid inequalities for the CVRP. Other interesting ideas would be to find inequalities for the loading problem, improving the current packing heuristic and using other heuristics for the 2DLP.

Many instances solved by Iori et al. (2007) branch-and-cut were not tested, therefore testing using a larger set of instances could provide a better analysis of our algorithm efficacy.

Acknowledgements: This research was partially supported by CAPES, CNPq and FAPESP.

References

B. Baker, E. Coffman, and R. Rivest. Orthogonal packings in two dimensions. *SIAM Journal on Computing*, 9(4):846–855, 1980.

G. Fuellerer, K. Doernera, R. Hartla, and M. Iori. Ant colony optimization for the two-dimensional loading vehicle routing problem. *Computers and Operations Research*, 36(3):655–673, 2009.

M. Gendreau, M. Iori, G. Laporte, and S. Martello. A guided tabu search for the vehicle routing problem with two-dimensional loading constraints. *Technical Report OR/04/1, DEIS, University of Bologna, Italy*, 2004.

M. Gendreau, M. Iori, G. Laporte, and S. Martello. A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. *Networks*, 51(1):4–18, 2008.

M. Iori. Metaheuristic algorithms for combinatorial optimization problems. *PhD thesis, DEIS, University of Bologna, Italy*, 2004.

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

J. Lysgaard, A. Lechtford, and R. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2003. `http://www.hha.dk/~lys/`.

S. Martello, D. Pisinger, and D. Vigo. The three-dimensional bin packing problem. *Operations Research*, 48(2):256–267, 2000. `http://www.diku.dk/~pisinger/`.

D. Naddef and G. Rinaldi. Branch-and-cut algorithms for the capacitated vrp. toth p. vigo d., eds. the vehicle routing problem. *SIAM Monographs on Discrete Mathematics and Applications*, 9: 53–84, 2002.

E. Zachariadis, C. Tarantilis, and C. Kiranoudis. A guided tabu search for the vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, 195(3):729–743, 2009.