Limited Memory Rank-1 Cuts for Vehicle Routing Problems

Diego Pecin[a], Artur Pessoa[b], Marcus Poggi[c], Eduardo Uchoa[b], Haroldo Santos[d]

[a] *École Polytechnique de Montréal and GERAD*
[b] *Universidade Federal Fluminense – Departamento de Engenharia de Produção*
[c] *Pontifícia Universidade Católica do Rio de Janeiro – Departamento de Informática*
[d] *Universidade Federal de Ouro Preto – Departamento de Computação*

Abstract

Pecin et al. (2016) introduced a "limited memory" technique that allows an efficient use of Rank-1 cuts in the Set Partitioning Formulation of Vehicle Routing Problems, motivating a deeper investigation of those cuts. This work presents a computational polyhedral study that determines the best possible sets of multipliers for cuts with up to 5 rows. Experiments with CVRP instances show that the new multipliers lead to significantly improved dual bounds and contributes decisively for solving an open instance with 420 customers.

Keywords: Set Partitioning, Polyhedral Combinatorics, Branch-cut-and-price algorithms

1. Introduction

The Vehicle Routing Problem (VRP) is among the most widely studied combinatorial optimization problems, due to its direct application in modern systems that distribute goods and services. Reflecting the large variety of conditions present in those systems, the VRP literature is spread into dozens of variants. For example, there are variants that consider vehicle capacities, time windows, multiple depots, heterogeneous fleet, pickups and deliveries, etc. The Set Partitioning Formulation (SPF) [1] can model most of those variants, the only assumption is that each customer should be visited once. Let $V = \{1, \ldots, n\}$ be the set of customers and let Ω be the set of all possible routes that respect the conditions of the considered variant. For each route $r \in \Omega$, let c_r be the cost of r, a_i^r be a binary coefficient that indicates whether r visits customer $i \in V$, and λ_r be a variable deciding if the route r is used or not. The formulation follows:

$$\text{Min:} \quad \sum_{r \in \Omega} c_r \lambda_r \tag{1}$$

$$\text{S.t.:} \quad \sum_{r \in \Omega} a_i^r \lambda_r = 1, \quad \forall i \in V, \tag{2}$$

$$\lambda_r \in \{0, 1\}, \quad \forall r \in \Omega. \tag{3}$$

Due to the huge number of variables, the linear relaxation of the SPF must be solved by column generation. The pricing subproblem consists in finding routes with minimum reduced cost, a problem that is often modeled as a shortest path with resource constraints and solved by *labeling algorithms* (see [2]).

The linear relaxation of the SPF is usually not strong enough to be the basis of state-of-the-art exact algorithms, at least for the most classical and competitive variants: the capacited VRP (CVRP) and the VRP with time windows (VRPTW). For that purpose, it must be strengthened with additional cuts. An algorithm that combines column and cut generation in a tree enumeration search is called a *branch-cut-and-price* (BCP) algorithm. According to the classification proposed in [3], *robust cuts* are those that do not change the structure of the pricing subproblem. In contrast, *non-robust cuts* change the pricing structure: each additional cut makes it harder, and so, too many cuts make it intractable. Robust cuts may be effective. In fact, some successful BCP algorithms [4, 5, 6, 7] only use them. However, it seems that the potential for robust cuts in the classical variants is exhausted. In fact, no effective new family of robust cuts was found in the last decade. As a consequence, an important line in cur-

Email addresses: diego.pecin@gerad.ca (Diego Pecin), artur@producao.uff.br (Artur Pessoa), poggi@inf.puc-rio.br (Marcus Poggi), uchoa@producao.uff.br (Eduardo Uchoa), haroldo@iceb.ufop.br (Haroldo Santos)

rent research is finding effective non-robust cuts that are not very harmful to the labeling algorithms used in the pricing.

The Set Partitioning constraints (2) are the natural source of non-robust cuts. However, well-known families of cuts like clique or odd holes [8] make the pricing too expensive [9]. An important advance was the introduction of the Subset Row Cuts (SRCs) by Jepsen et al. [10]. The proposed SRCs are subfamilies of clique and lifted odd holes defined over a small subset of the rows in (2). Those particular non-robust cuts can be better treated by the labeling algorithms. In fact, some of the best exact algorithms for the CVRP and VRPTW use SRCs [10, 11, 12, 13]. A direct generalization of SRCs is to consider any Chvátal-Gomory rank 1 cut [14] obtained over a small subset of the rows. Preliminary experiments with those *Rank-1 Cuts (R1Cs)* were performed in [15].

The above mentioned algorithms separate SRCs in a very careful way, limiting the number of added cuts in order to avoid an excessive impact in the pricing. As a result, the potential of these cuts is not fully exploited. Pecin et al. [16] introduced the *limited memory* technique for making SRCs or R1Cs much less harmful to the labeling algorithms, without necessarily compromising their effectivity. The computational results on CVRP instances show a breakthrough: due to the improved bounds, the size of the largest solved literature instance increased from 150 to 360 customers. The improved bounds made possible by the limited memory also lead to big advances on VRPTW [17]. In this new context, there is a clear motivation for obtaining even better bounds by also separating more complex Rank-1 cuts.

This work is aimed at answering the following question: *what are the optimal sets of multipliers for Rank-1 Cuts with up to 5 rows?* This is done by a computational investigation of the set partitioning polyhedra. Experiments on CVRP instances show that the newly discovered cuts indeed lead to significantly improved bounds. Finally, we show how this allows to solve the Golden_20 instance (420 customers). The paper is organized as follows. Section 2 reviews SRCs, Rank-1 cuts and the limited memory technique. Section 3 describes the metodology used for discovering the optimal sets of multipliers. Section 4 presents experiments on CVRP instances. Finally, Section 5 contains some concluding remarks.

2. Limited Memory Rank-1 Cuts

Given a base set $C \subseteq V$ and a multiplier p_i for each $i \in C$, the following valid inequality for SPF is a Rank-1 Cut (R1C):

$$\sum_{r \in \Omega} \left\lfloor \sum_{i \in C} p_i a_i^r \right\rfloor \lambda_r \leq \left\lfloor \sum_{i \in C} p_i \right\rfloor \tag{4}$$

The Subset Row Cuts (SRCs) were introduced in Jepsen et al. [10] and correspond to the particular case where all the multipliers have the same value $p = 1/k$, for some integer k. The following base set sizes and multipliers were investigated in that work:

- 3SRCs: $|C| = 3$ and $p = 1/2$. Those cuts have RHS 1 and can be viewed as weakened clique cuts. Nevertheless, they are still very effective in improving bounds and were the only SRCs actually separated in [10, 11, 12, 13].
- 5,2SRCs: $|C| = 5$ and $p = 1/2$. Those cuts have RHS 2 and can be viewed as weakened odd hole cuts.

Pecin et al. [16] separated 3SRCs, 5,2SRCs and also:

- 4SRCs: $|C| = 4$ and $p = 2/3$. In spite of not having a multiplier of format $1/k$, they were still called SRCs in that work.
- 5,1SRCs: $|C| = 5$ and $p = 1/3$.

The only work that investigated and separated general R1Cs is Petersen et al. [15]. The used multiplier sets were of mod-k type [18], i.e, each individual multiplier should belong to $\{0, 1/k, \ldots, k-1/k\}$ for some small integer k. Separation was performed exhaustively, testing all possibilities for certain values of C and k. The authors also tested finding the multipliers using the MIP formulation proposed in [19]. The experiments indicated that R1Cs more complex than SRCs could indeed improve bounds. However, those cuts could not be incorporated into state-of-the-art

codes: not only the separation itself is very expensive, but the added cuts slow down too much the labeling algorithm, making the pricing intractable.

Given $C \subseteq V$, a vector of multipliers p of dimension $|C|$, a memory set M, $C \subseteq M \subseteq V$, the limited memory (C, M, p)-Rank 1 Cut (lm-R1C for short) is:

$$\sum_{r \in \Omega} \alpha(C, M, p, r)\lambda_r \leq \left\lfloor \sum_{i \in C} p_i \right\rfloor, \tag{5}$$

where the coefficient of a route r is computed as:

1: **function** $\alpha(C, M, p, r)$
2: *coeff* $\leftarrow 0$, *state* $\leftarrow 0$
3: **for every vertex** $i \in r$ (in order) **do**
4: **if** $i \notin M$ **then**
5: *state* $\leftarrow 0$
6: **else if** $i \in C$ **then**
7: *state* $\leftarrow$ *state* $+ p_i$
8: **if** *state* ≥ 1 **then**
9: *coeff* $\leftarrow$ *coeff* $+ 1$, *state* $\leftarrow$ *state* $- 1$
10: **return** *coeff*

Variable *coeff* stores the coefficient to be returned. Each time a vertex i in C is visited, variable *state* is increased by p_i. When *state* becomes larger or equal to 1, its value is reduced by 1 unit and *coeff* is incremented. The previous definition is completely analogous to that of the limited memory Subset Row Cuts proposed (or lm-SRC) by Pecin et al. [16], the only difference being that the multiplier vector p replaces a single scalar, which was originally the same for all $i \in C$. When $M = V$, the Function α will return $\lfloor \sum_{i \in C} p_i a_i^r \rfloor$ and the lm-R1C will be identical to an R1C. On the other hand, when M is strictly contained in V, the lm-R1C may be a weakening of its corresponding R1C. This happens because every time the route r leaves M, the variable *state* is reset to zero, potentially decreasing the returned coefficient. Nevertheless, using a dynamic adjustment of the memory sets, the lm-R1Cs can still yield exactly the same gap improvements of ordinary R1Cs [16].

The advantage of the lm-R1Cs over R1Cs (or lm-SRCs over SRCs) is their much reduced impact on the labeling algorithm used in the pricing, when $|M| \ll |V|$. Labeling algorithms roughly consist of expanding sets of *non-dominated* partial routes called *buckets*, and then reducing the bucket sizes by eliminating newly dominated routes. Clearly, a crucial step of such algorithm is the bucket reduction by dominance, which heavily depends on the domination rule. Such a rule must ensure that the dominated partial route can always be replaced by the dominating one in any feasible route that contains it, without increasing (and potentially reducing) its cost. For the pricing subproblem that remains after adding lm-R1Cs to the SPF given by (1)-(3), the labeling algorithm keeps track of the state (the current value of variable *state* in the computation of α) of each lm-R1C with non-zero dual variable, for each partial route contained in each bucket. Then, for a partial route r_1 to dominate another partial route r_2, the reduced cost of r_2 must exceed that of r_1 at least in the amount of the sum of the dual variables of all lm-R1Cs whose states in r_1 are larger than that in r_2. This sum represents that worst-case increase in the reduced cost of r_1 that may not occur for r_2 considering the same feasible extension. If, on one hand, the limited memory technique greatly reduces the impact of lm-R1Cs on the pricing by resetting many states to zero, on the other hand, having a small number of possible states is still an essential property of these cuts to keep the pricing subproblem tractable. Note that this number is as small as the lowest common denominator of the components of the vector p for the newly proposed cuts.

3. Finding Optimal Multipliers

In a column generation context it is not enough to separate cuts that are only defined over the variables in the current restricted Master LP, since other unknown variables will be generated by the pricing. This means that a cut must have well-defined coefficients for every possible variable. Therefore, in order to obtain the best R1Cs defined over a small subset of the rows in the SPF, we propose to investigate the following polyhedra, denoted here as the Complete Set Partitioning Polyhedron of order n (CSPP(n)) and defined as:

$$CSPP(n) = \text{Conv}\left\{ \sum_{j=1}^{2^n-1} b_j \lambda_j = e_n,\ \lambda \in \{0,1\}^{2^n-1} \right\},$$

where e_n is the n-dimensional all-ones vector and b_j is the n-dimensional vector that encodes the binary representation of number j. For example, if $n = 3$ then $b_1 = (001), b_2 = (010), b_3 = (011), b_4 = (100), b_5 = (101), b_6 = (110), b_7 = (111)$. Next, we describe each step of the methodology for the computational study of $CSPP(n)$, $2 \leq n \leq 5$.

1. First enumerate the set $P(n)$ of binary points in $CSPP(n)$. Then, give $P(n)$ as input to the PORTA software [20] to compute a minimum representation of $CSPP(n)$ as a set of equalities and inequalities. Let $F(n)$ be the set of computed inequalities, excluding the inequalities that are equivalent to $\lambda_j \geq 0$, for some j, $1 \leq j \leq 2^n - 1$. More precisely, $F(n)$ contains a pair (π, π_0) ($\pi \in \mathbb{R}^{2^n-1}$ and $\pi_0 \in \mathbb{R}$) for each inequality $\pi^\top \lambda \leq \pi_0$ that defines a non-trivial facet of $CSPP(n)$. PORTA already outputs vectors in a normalized format where the coefficients in (π, π_0) are non-negative integers (all non-trivial facets of $CSPP(n)$ can be described by inequalities in this format).
   - $P(2) = \{(110), (001)\}$ and $F(2) = \emptyset$. This means that no cut can be obtained in this way by only looking at two rows of SPF.
   - $P(3) = \{(1101000), (1000010), (0100100), (0011000), (0000001)\}$ and $F(3) = \{((0010111), 1)\}$. The corresponding cut $\lambda_3 + \lambda_5 + \lambda_6 + \lambda_7 \leq 1$ is readily identified as a 3SRC. This means that no other interesting cut can be obtained from 3 rows of SPF.
   - $|P(4)| = 15$ and $|F(4)| = 8$. It is still easy to show that all the inequalities in $F(4)$ are R1Cs. Four of them can be obtained using multipliers $(0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ and its permutations, so they are 3SRCs. The remaining four inequalites are R1Cs obtained with multipliers $(\frac{2}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ and its permutations. This new family of cuts dominates the 4SRCs used in [16].
   - $|P(5)| = 52$ and $|F(5)| = 294$. The inequalities in $F(5)$ were analyzed as described next.
2. In order to analyze a much smaller number of non-trivial facets than originally generated by PORTA, it is useful to identify which facets are symmetric with respect to permutations in the equalities that define $CSPP(n)$. Given a permutation σ of $(1, \ldots, n)$, define $\sigma(b_j)$ as the binary vector obtained from b_j by applying the permutation σ to its components. Then, given two binary points λ and λ' of $CSPP(n)$, λ' is symmetric to λ with respect to σ if for every j and k such that $\sigma(b_k) = b_j$, $\lambda'_k = \lambda_j$. For example, applying the permutation $\sigma = (1, 3, 2)$ to the binary point $\lambda = (0011000)$ of $P(3)$ gives the binary point $\lambda' = (0100100)$ since $j = 3$ ($b_j = (110)$) maps to $k = 5$ ($b_k = (101)$), and $j = 4$ ($b_j = (001)$) maps to $k = 2$ ($b_k = (010)$). As a result, the same family of transformations may be applied to the components of π in an inequality, leading the symmetric cuts. Two inequalities are symmetric if they have the same right-hand side, and there is a permutation σ that maps the vector π of one of them to the other.
3. For each inequality (π, π_0) in $F(5)$ (after removing the symmetric ones), we solve the following linear program, where the variables u_i and f_j respectively represent the i-th multiplier that defines the cut, and the fractional part of the coefficient of λ_j in the cut (or the right-hand side of the cut, for $j = 0$) before it is rounded down:

$$
\begin{aligned}
\text{min.:} \quad & \sum_{i=1}^{n} u_i \\
\text{s.t.:} \quad & \sum_{i=1}^{n} u_i b_{ji} = \pi_j + f_j \ \ \forall j \in \{1, \ldots, 2^n - 1\} \\
& \sum_{i=1}^{n} u_i = \pi_0 + f_0 \\
& 0 \leq u_i \leq 1 - \epsilon \ \ \forall i \in \{1, \ldots, n\} \\
& 0 \leq f_j \leq 1 - \epsilon \ \ \forall j \in \{0, \ldots, 2^n - 1\}.
\end{aligned}
$$

   The inequality has rank 1 if and only if this LP is feasible. This Linear Program is a simplification of the MIP proposed in [19] for separating cuts of rank 1. Since all integer variables from [19] are fixed, the resulting LPs are quite easy to solve. As in that work, we used the tolerance

$\epsilon = 0.01$. These linear programs were solved using the COIN-OR linear programming solver (CLP) version 1.16[21].

4. For the facets that are proved to have rank 1, in order to express the obtained multipliers as fractions with small denominators, we solve again the previous formulation replacing each u_i by v_i/D, where D is a tentative denominator and $v_i \in \{0, \ldots, D-1\}$ is an integer variable. The resulting MIP is solved for increasing values of D starting from 2 until a feasible solution is found. Fortunately, the maximum value of D needed was 5. Even though we are now solving MIPs, these problems have a reduced size and are also very easy. COIN-OR CBC 2.9[21] was used to solve these MIPs.

Our findings regarding $CSPP(5)$ are resumed next. From its 294 non-trivial facets, 117 of these correspond to R1Cs. Ten of those facets are 3 row cuts (multipliers $(0, 0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$), twenty are 4 row cuts (multipliers $(0, \frac{2}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})$), and two facets correspond to the already known `5,1SRCs` and `5,2SRCs`. The remaining 85 facets correspond to the R1Cs obtained with the following multipliers (and their permutations):

$$\left(\frac{2}{4}, \frac{2}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right) \tag{6}$$

$$\left(\frac{3}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right) \tag{7}$$

$$\left(\frac{3}{5}, \frac{2}{5}, \frac{2}{5}, \frac{1}{5}, \frac{1}{5}\right) \tag{8}$$

$$\left(\frac{2}{3}, \frac{2}{3}, \frac{2}{3}, \frac{1}{3}, \frac{1}{3}\right) \tag{9}$$

$$\left(\frac{3}{4}, \frac{3}{4}, \frac{2}{4}, \frac{2}{4}, \frac{1}{4}\right) \tag{10}$$

Those new R1Cs have RHS 1 or 2 and are also subfamilies of weakened cliques and odd holes.

Another question is if there are multipliers capable of generating non facet defining cuts which can still be useful to improve lower bounds for $|C|$ up to 5. By generating all extreme fractional points to the polyhedron defined by only the facets of rank 1 or less of $CSPP(5)$ (say R1-$CSPP(5)$) facets and executing the MIP separation of [19] over those points, we discovered that such multipliers *do not* exist. The 72 fractional points of R1-$CSPP(n)$ becomes only two after eliminating the symmetric ones:

$$\left(0, \frac{1}{4}, 0, \frac{1}{4}, \frac{1}{4}, 0, \frac{1}{4}, \frac{1}{4}, 0, \frac{1}{4}, 0, \frac{1}{4}, 0, 0, 0,\right.$$

$$\left.\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, 0, 0, 0, 0, 0, 0, \frac{1}{4}, 0, 0, 0, 0, 0, 0, 0\right),$$

representing 60 symmetric points, and

$$\left(\frac{2}{11}, \frac{2}{11}, 0, \frac{2}{11}, 0, \frac{3}{11}, \frac{1}{11}, \frac{2}{11}, \frac{3}{11}, 0, \frac{1}{11}, \frac{3}{11}, 0, 0, 0,\right.$$

$$\left.\frac{2}{11}, \frac{3}{11}, \frac{3}{11}, 0, 0, \frac{1}{11}, 0, 0, 0, 0, \frac{1}{11}, 0, \frac{1}{11}, 0, 0, 0\right).$$

representing 12 symmetric points. Therefore, the MIP separation was only executed for those 2 points.

4. Computational Experiments

A separation for the new R1Cs (actually lm-R1Cs) was implemented and added to the BCP code for the CVRP described in [16], using the optimal sets of multipliers pre-computed by the procedure proposed in Section 3. The separation procedure is simple: (i) For every $i \in V$, define $N(i) \subseteq V$ as the set containing the 15 other customers closest to i, ties broken arbitrarily; (ii) Enumerate all base-sets C with cardinalities 3, 4 and 5 such that, for each i and j in C, $j \in N(i)$; (iii) Test the violation of all R1Cs obtained by applying the new multipliers to C.

We performed experiments to determine the efficiency of some combinations of cuts, measuring duality gaps over a set of 11 representative classical instances ranging from 36 to 199 customers: A-n37-k6, A-n62-k8, A-n63-k10, A-n64-k9, B-n50-k8, B-n56-k7, E-n76-k10, E-n76-k14, E-n101-k14, M-n151-k12, and M-n200-k17. Table 1 reports average gaps obtained by the following successively stronger bounding mechanisms: the linear relaxation of SPF, this relaxation plus the cuts defined the over edge variables used in [5], plus 3SRCs, plus 4SRCs and 5SRcs

(the bounding mechanism used in [16]), and finally, plus the newly discovered R1Cs. As it can be seen, on average the new cuts removed 30% of the residual gap (from 0.24% to 0.17%).

	Gap(%)
Only CG (elementary routes)	2.63
+ robust cuts	0.98
+ 3SRCs	0.35
+ 4SRCs + 5SRCs	0.24
+ other R1Cs up to 5 rows	0.17

The stronger lower bounds in the improved BCP code allowed, for the first time, to determine the optimal solution to instance Golden_20, with 420 customers. The best known solutions obtained by recent state-of-the-art heuristic methods were: Vidal et al. [22] 1818.32, [23] 1818.25, Jin et al. [24] 1817.89, and Liu et al. [25] 1817.86. The root of BCP used to find the lower bound of 1814.4, the new cuts increased it to 1815.0. The optimal integral solution found has value 1817.59. The search tree had 370 nodes, the total cpu time spent was 7.1 days on a single core of a i7-3960X 3.30GHz processor. The lower bound improvement at the root was decisive, we estimate that without the new cuts the solution time would be at least one order of magnitude larger.

5. Conclusions

This paper presented the optimal sets of multipliers for obtaining R1Cs with up to 5 rows of the SPF. Computational results show that the newly found cuts are indeed effective on CVRP. Moreover, it provides a methodology for doing similar computational polyhedral investigations, taking advantage of readily available tools, in other problems.

Acknowledgment

We thank the Conselho Nacional de Pesquisa (CNPq) for having partially funded this project.

References

[1] M. Balinski, R. Quandt, On an integer program for a delivery problem, Operations Research 12 (1964) 300–304.

[2] L. Pugliese, F. Guerriero, A survey of resource constrained shortest path problems: Exact solution approaches, Networks 62 (2013) 183–200.

[3] M. Poggi de Aragão, E. Uchoa, Integer program reformulation for robust branch-and-cut-and-price, in: L. Wolsey (Ed.), Annals of Mathematical Programming in Rio, Búzios, Brazil, pp. 56–61.

[4] N. Kohl, J. Desrosiers, O. Madsen, M. Solomon, F. Soumis, 2-path cuts for the vehicle routing problem with time windows, Transportation Science 33 (1999) 101–116.

[5] R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, R. Werneck, Robust branch-and-cut-and-price for the capacitated vehicle routing problem, Mathematical programming 106 (2006) 491–511.

[6] B. Kallehauge, J. Larsen, O. Madsen, Lagrangian duality applied to the vehicle routing problem with time windows, Computers & Operations Research 33 (2006) 1464–1487.

[7] S. Røpke, Branching decisions in branch-and-cut-and-price algorithms for vehicle routing problems, Presentation in Column Generation 2012 (2012).

[8] E. Balas, M. Padberg, Set partitioning: A survey, SIAM review 18 (1976) 710–760.

[9] S. Spoorendonk, G. Desaulniers, Clique inequalities applied to the vehicle routing problem with time windows, INFOR: Information Systems and Operational Research 48 (2010) 53–67.

[10] M. Jepsen, B. Petersen, S. Spoorendonk, D. Pisinger, Subset-row inequalities applied to the vehicle-routing problem with time windows, Operations Research 56 (2008) 497–511.

[11] G. Desaulniers, F. Lessard, A. Hadjar, Tabu search, generalized k-path inequalities, and partial elementarity for the vehicle routing problem with time windows, Transportation Science 42 (2008) 387–404.

[12] R. Baldacci, A. Mingozzi, R. Roberti, New route relaxation and pricing strategies for the vehicle routing problem, Operations Research 59 (2011) 1269–1283.

[13] C. Contardo, R. Martinelli, A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints, Discrete Optimization 12 (2014) 129–146.

[14] V. Chvátal, Edmonds polytopes and a hierarchy of combinatorial problems, Discrete mathematics 4 (1973) 305–337.

[15] B. Petersen, D. Pisinger, S. Spoorendonk, Chvátal-Gomory Rank-1 cuts used in a Dantzig-Wolfe decomposition of the vehicle routing problem with time windows, in: The Vehicle Routing Problem: Latest Advances and New Challenges, Springer, 2008, pp. 397–419.

[16] D. Pecin, A. Pessoa, M. Poggi, E. Uchoa, Improved branch-cut-and-price for capacitated vehicle routing, Mathematical Programming Computation (2016) 1–40.

[17] D. Pecin, C. Contardo, G. Desaulniers, E. Uchoa, New enhancements for the exact solution of the vehicle routing problem with time windows, Technical Report G-2016-13, Les cahiers du GERAD, 2016.

[18] A. Caprara, M. Fischetti, A. N. Letchford, On the separation of maximally violated mod-k cuts, Mathematical Programming 87 (2000) 37–56.

[19] M. Fischetti, A. Lodi, Optimizing over the first Chvátal closure, Mathematical Programming 110 (2007) 3–20.

[20] T. Christof, A. Löbel, PORTA–POlyhedron Representation Transformation Algorithm, 2008, URL: http://www. zib. de/Optimization/Software/Porta (2012).

[21] The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community, IBM Journal of Research and Development 47 (2003) 57.

[22] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, W. Rei, A hybrid genetic algorithm for multidepot and periodic vehicle routing problems, Operations Research 60 (2012) 611–624.

[23] C. Groër, B. Golden, E. Wasil, A parallel algorithm for the vehicle routing problem, INFORMS Journal on Computing 23 (2011) 315–330.

[24] J. Jin, T. G. Crainic, A. Løkketangen, A cooperative parallel metaheuristic for the capacitated vehicle routing problem, Computers & Operations Research 44 (2014) 33–41.

[25] W. Liu, X. Li, A problem-reduction evolutionary algorithm for solving the capacitated vehicle routing problem, Mathematical Problems in Engineering 501 (2014) 165476.