Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem

Ricardo Fukasawa[1], Jens Lysgaard[2], Marcus Poggi de Aragão[3], Marcelo Reis[3], Eduardo Uchoa[4]*, and Renato F. Werneck[5]

[1] School of Industrial and Systems Engineering, GeorgiaTech, USA. `rfukasaw@isye.gatech.edu`.
[2] Department of Management Science and Logistics, Aarhus School of Business, Denmark. `lys@asb.dk`.
[3] Departamento de Informática, PUC Rio de Janeiro, Brazil. {`poggi,mreis`}`@inf.puc-rio.br`.
[4] Departamento de Engenharia de Produção, Universidade Federal Fluminense, Brazil. `uchoa@producao.uff.br`.
[5] Department of Computer Science, Princeton University, USA. `rwerneck@cs.princeton.edu`.

Abstract. The best exact algorithms for the Capacitated Vehicle Routing Problem (CVRP) have been based on either branch-and-cut or Lagrangean relaxation/column generation. This paper presents an algorithm that combines both approaches: it works over the intersection of two polytopes, one associated with a traditional Lagrangean relaxation over q-routes, the other defined by bound, degree and capacity constraints. This is equivalent to a linear program with exponentially many variables and constraints that can lead to lower bounds that are superior to those given by previous methods. The resulting branch-and-cut-and-price algorithm can solve to optimality all instances from the literature with up to 135 vertices. This doubles the size of the instances that can be consistently solved.

1 Introduction

Let $G = (V, E)$ be an undirected graph with vertices $V = \{0, 1, \ldots, n\}$. Vertex 0 represents the *depot*, whereas all others represent *clients*, each with an associated demand $d(\cdot)$. Each edge $e \in E$ has a nonnegative length $\ell(e)$. Given G and two positive integers (K and C), the *Capacitated Vehicle Routing Problem* (CVRP) consists of finding routes for K vehicles satisfying the following constraints: (i) each route starts and ends at the depot, (ii) each client is visited by a single vehicle, and (iii) the total demand of all clients in any route is at most C. The goal is to minimize the sum of the lengths of all routes. This classical NP-hard problem is a natural generalization of the Travelling Salesman Problem (TSP), and has widespread application itself. The CVRP was first proposed in 1959 by

* Corresponding author.

Dantzig and Ramser [13] and has received close attention from the optimization community since then.

A landmark exact algorithm for the CVRP, presented in 1981 by Christofides, Mingozzi and Toth [11], uses a Lagrangean bound from minimum q-route subproblems. A *q-route* is a walk that starts at the depot, traverses a sequence of clients with total demand at most C, and returns to the depot. Some clients may be visited more than once, so the set of valid CVRP routes is strictly contained in the set of q-routes. The resulting branch-and-bound algorithm could solve instances with up to 25 vertices, a respectful size at the time.

Several other algorithms using Lagrangean relaxation appear in the literature. Christofides et al. [11] also describe a lower bound based on k-degree center trees, which are minimum spanning trees having degree $K \leq k \leq 2K$ on the depot, plus $2K - k$ least-cost edges. Lagrangean bounds based on K-trees (sets of $n + K - 1$ edges spanning V) having degree $2K$ in the depot were used by Fisher [14] and by Martinhon, Lucena, and Maculan [24], among others. Miller [25] presented an algorithm based on minimum b-matchings having degree $2K$ at the depot and 2 on the remaining vertices. Lagrangean bounds can be improved by dualizing capacity inequalities [14, 25] and also comb and multistar inequalities [24].

Another family of exact algorithms stems from the formulation of the CVRP as a set partitioning problem by Balinski and Quandt [8]. A column covers a set of vertices S with total demand not exceeding C and has the cost of a minimum route over $\{0\} \cup S$. Unfortunately, the formulation is not practical because pricing over the exponential number of columns requires the solution of capacitated prize-collecting TSPs, a problem almost as difficult as the CVRP itself. Agarwal, Marthur and Salkin [7] proposed a column generation algorithm on a modified set partitioning problem where column costs are given by a linear function over the vertices yielding a lower bound on the actual route cost. Columns with the modified cost can be priced by solving easy knapsack problems. Hadjiconstantinou et al. [17] derive lower bounds from heuristic solutions to the dual of the set partitioning formulation. The dual solutions are obtained by the so-called additive approach, combining the q-route and k-shortest path relaxations.

For further information and comparative results on the algorithms mentioned above, we refer the reader to the surveys by Toth and Vigo [31, 32].

Recent research on the CVRP has been concentrated on the polyhedral description of the convex hull of the edge incidence vectors that correspond to K feasible routes and on the development of effective separation procedures [1, 3, 5, 6, 12, 20, 26]. In particular, Araque et al. [4], Augerat et al. [6], Blasum and Hochstättler [9], Ralphs et al. [30], Achuthan, Caccetta, and Hill [2] and Lysgaard, Letchford, and Eglese [23] describe complete branch-and-cut (BC) algorithms. These are the best exact methods currently available for the CVRP. However, the addition of several elaborate classes of cuts does not guarantee tight lower bounds, especially for large values of K ($K \geq 7$, say). Closing the resulting duality gap usually requires exploring several nodes in the branch-and-cut tree. Even resorting to massive computational power (up to 80 processors running in parallel in a recent work by Ralphs [29, 30]) several instances with fewer than 80 vertices, including some proposed more than 30 years ago by Christofides and Eilon [10], can not be solved at all. In fact, branch-and-cut algorithms for the CVRP seem to be experiencing a "diminishing returns" stage, where substantial theoretical and implementation efforts achieve practical results that are only marginally better than those of previous works.

We present a new exact algorithm for the CVRP that seems to break through this situation. The main idea is to combine the branch-and-cut approach with the q-routes approach (which we interpret as column generation instead of the original Lagrangean relaxation) to derive superior lower bounds. Since the resulting formulation has an exponential number of both columns and rows, this leads to a branch-and-cut-and-price (BCP) algorithm. Computational experiments over the main instances from the literature show that this algorithm is very consistent on solving instances with up to 100 vertices. Eighteen open instances were solved for the first time.

The idea of combining column and cut generation to improve lower bounds has rarely been used, since new dual variables corresponding to separated cuts may have the undesirable effect of changing the structure of the pricing subproblem. However, if cuts are expressed in terms of variables from a suitable original formulation, they can be incorporated into the column generation process without disturbing the pricing. We refer to branch-and-bound procedures based on such formulations as *robust branch-and-cut-and-price* algorithms. Poggi de Aragão and Uchoa [28] present a detailed discussion on this subject, including new reformulation techniques that extend the applicability of robust branch-and-cut-and-price algorithms to virtually any combinatorial optimization problem. This article on the CVRP is part of a larger effort to demonstrate that these methods lead to significant improvements on a wide variety of problems. Major advances have already been reported on two other problems: capacitated minimum spanning tree [15] and generalized assignment [27].

This article is organized as follows. Section 2 describes the integer programming formulation we will deal with. Section 3 gives a general description of our algorithm, including its two main components: column and cut generation. Following the work of Irnich and Villeneuve [18] on the CVRP with time windows, our column generation procedure eliminates q-routes with small cycles. The separation routines are based on the families of inequalities recently discussed by Letchford, Eglese, and Lysgaard [20, 23]. Section 4 presents an empirical analysis of our method. Final remarks are made in Sect. 5.

2 The New Formulation

A classical formulation for the CVRP [19] represents by x_{ij} the number of times a vehicle traverses the edge $(i, j) \in E$. The set of client vertices is denoted by $V_+ = \{1, \ldots, n\}$. Given a set $S \subseteq V_+$, let $d(S)$ be the sum of the demands of all vertices in S, and let $\delta(S)$ be the cut-set defined by S. Also, let $k(S) = \lceil d(S)/C \rceil$.

Define the following polytope in $\mathbb{R}^{|E|}$:

$$P_1 \quad = \quad \begin{cases} \sum\limits_{e\in\delta(\{i\})} x_e = 2 & \forall\, i \in V_+ & (1) \\ \sum\limits_{e\in\delta(\{0\})} x_e = 2\cdot K & & (2) \\ \sum\limits_{e\in\delta(S)} x_e \geq 2\cdot k(S) & \forall\, S \subseteq V_+ & (3) \\ x_e \leq 1 & \forall\, e \in E \setminus \delta(\{0\}) & (4) \\ x_e \geq 0 & \forall\, e \in E \;\; . \end{cases}$$

Constraints (1) state that each client is visited once by some vehicle, whereas (2) states that K vehicles must leave and enter the depot. Constraints (3) are rounded capacity inequalities, which require all subsets to be served by enough vehicles. Constraints (4) enforce that each edge not adjacent to the depot is traversed at most once (edges adjacent to the depot can be used twice when a route serves only one client). The integer vectors x in P_1 define all feasible solutions for the CVRP. There are exponentially many inequalities of type (3), so the lower bound given by

$$L_1 = \min_{x\in P_1} \sum_{e\in E} \ell_e x_e$$

must be computed by a cutting plane algorithm.

Alternatively, a formulation with an exponential number of columns can be obtained by defining variables (columns) that correspond to q-routes without 2-cycles (subpaths $i \to j \to i$, $i \neq 0$). Restricting the q-routes to those without such cycles improves the formulation and does not change the complexity of the pricing [11]. Let Q be an $m \times p$ matrix where the columns are the edge incidence vectors of all p such q-routes. Let q_j^e be the coefficient associated with edge e in the j-th column of Q. Consider the following polytope in $\mathbb{R}^{|E|}$, defined as the projection of a polytope in $\mathbb{R}^{p+|E|}$:

$$P_2 = \;\mathrm{proj}_x \begin{cases} \sum\limits_{j=1}^{p} q_j^e \cdot \lambda_j - \quad x_e & = 0 \;\; \forall e \in E & (5) \\ \sum\limits_{j=1}^{p} \lambda_j & = K & (6) \\ \qquad \sum\limits_{e\in\delta(\{i\})} x_e & = 2 \;\; \forall\, i \in V_+ & (1) \\ \qquad x_e & \geq 0 \;\; \forall\, e \in E \\ \lambda_j & \geq 0 \;\; \forall\, j \in \{1,\ldots,p\} \;\; . \end{cases}$$

Constraints (5) define the coupling between variables x and λ. Constraint (6) defines the number of vehicles to use. It can be shown that the set of integer vectors in P_2 also defines all feasible solutions for the CVRP. Due to the exponential number of variables λ, the lower bound given by

$$L_2 = \min_{x\in P_2} \sum_{e\in E} \ell_e x_e$$

must be computed using column generation or Lagrangean relaxation.

The description of polyhedra associated with column generation or Lagrangean relaxation in terms of two sets of variables, λ and x, used in the definition of P_2, is called *Explicit Master* in [28]. The main contribution of this article is a formulation that amounts to optimizing over the intersection of polytopes P_1 and P_2. The Explicit Master format describes such intersection as follows:

$$P_3 = P_1 \cap P_2 \;=\; \text{proj}_x \left\{ \begin{array}{rlll} & \sum_{e \in \delta(\{i\})} x_e & = 2 & \forall\, i \in V_+ \quad (1) \\ & \sum_{e \in \delta(\{0\})} x_e & = 2 \cdot K & \quad (2) \\ & \sum_{e \in \delta(S)} x_e & \geq 2 \cdot k(S) & \forall\, S \subseteq V_+ \quad (3) \\ & x_e & \leq 1 & \forall\, e \in E \setminus \delta(\{0\}) \quad (4) \\ \sum_{j=1}^{p} q_j^e \cdot \lambda_j \;- & x_e & = 0 & \forall e \in E \quad (5) \\ \sum_{j=1}^{p} \lambda_j & & = K & \quad (6) \\ & x_e & \geq 0 & \forall\, e \in E \\ \lambda_j & & \geq 0 & \forall\, j \in \{1, \ldots, p\}\ . \end{array} \right.$$

Constraint (6) can be discarded, since it is implied by (2) and (5). Computing the improved lower bound

$$L_3 = \min_{x \in P_3} \sum_{e \in E} \ell_e x_e$$

requires solving a linear program with an exponential number of both variables and constraints. A more compact LP is obtained if every occurrence x_e in (1)–(4) is replaced by its equivalent given by (5). The resulting LP will be referred to as the *Dantzig-Wolfe Master problem* (DWM):

$$\text{DWM} \;=\; \left\{ \begin{array}{llll} L_3 = \min \sum_{j=1}^{p} \sum_{e \in E} \ell_e \cdot q_j^e \cdot \lambda_j & & & (7) \\ \text{s.t.} \quad \sum_{j=1}^{p} \sum_{e \in \delta(\{i\})} q_j^e \cdot \lambda_j & = 2 & \forall\, i \in V_+ & (8) \\ \quad \sum_{j=1}^{p} \sum_{e \in \delta(\{0\})} q_j^e \cdot \lambda_j & = 2 \cdot K & & (9) \\ \quad \sum_{j=1}^{p} \sum_{e \in \delta(S)} q_j^e \cdot \lambda_j & \geq 2 \cdot k(S) & \forall\, S \subseteq V_+ & (10) \\ \quad \sum_{j=1}^{p} q_j^e \cdot \lambda_j & \leq 1 & \forall\, e \in E \setminus \delta(\{0\}) & (11) \\ \quad \lambda_j & \geq 0 & \forall\, j \in \{1, \ldots, p\}\ . & \end{array} \right.$$

Capacity inequalities are not the only ones that can appear in the DWM. A generic cut $\sum_{e \in E} a_e x_e \geq b$ can be included as $\sum_{j=1}^{p} (\sum_{e \in E} a_e q_j^e) \cdot \lambda_j \geq b$. In fact, we added all classes of cuts described in [23].

3 The Branch-and-Cut-and-Price Algorithm

Column Generation. The reduced cost of a column (λ variable) is the sum of the reduced costs of the edges in the corresponding q-route. Let μ, ν, π, and ω be the dual variables associated with constraints (8), (9), (10), and (11), respectively. The reduced cost $\bar{c}_e$ of an edge e is given by:

$$\bar{c}_e = \begin{cases} \ell_e - \mu_i - \mu_j - \sum\limits_{S|\delta(S)\ni e} \pi_S - \omega_e & e = \{i,j\} \in E \setminus \delta(\{0\}) \\ \ell_e - \nu - \mu_j - \sum\limits_{S|\delta(S)\ni e} \pi_S & e = \{0,j\} \in \delta(\{0\}) \end{cases} .$$

An additional generic cut $\sum_{j=1}^{p}(\sum_{e\in E} a_e q_j^e)\cdot\lambda_j \geq b$ in DWM, with dual variable α, contributes with the value $-a_e \cdot \alpha$ to the computation of $\bar{c}_e$.

The pricing subproblem consists of finding q-routes (columns) of minimum reduced cost. Although this problem is NP-hard in general, it can be solved in pseudopolynomial time if all demands are integer [11].

The basic data structure is a $C \times n$ matrix M. Each entry $M(d,v)$ will represent the least costly walk that reaches vertex v using total demand exactly d. The entry contains a *label* consisting of the vertex (v), the cost of the walk, and a pointer to a label representing the walk up to the previous vertex. The matrix is filled by dynamic programming, row by row, starting with $d = 1$. For each row d, the algorithm goes through each entry v and, for each neighbor w of v, evaluates the extension of the walk represented by $M(d,v)$ to w. If $\bar{c}(M(d,v)) + \bar{c}_{(v,w)} < \bar{c}(M(d+d(w),w))$, $M(d+d(w),w)$ is updated. Eventually, we will have the most negative walk with accumulated demand at most C that arrives at each vertex v. Extending the walk to the depot (whose demand is zero), we obtain the corresponding q-route. All negative q-routes thus found (there will be at most n) are added to the linear program. There are nC entries in the matrix, and each is processed in $O(n)$ time, so the total running time is $O(n^2C)$.

To strengthen the formulation, we look for s-cycle-free q-routes, for small values of s. The algorithm operates as above, using dynamic programming to fill a $C \times n$ matrix with partial walks. Each entry in the matrix is no longer a single label, but a *bucket* of labels. Bucket $M(d,v)$ represents not only the cheapest s-cycle-free walk with total demand d that ends in v, but also alternative walks that ensure that all possible s-vertex extensions from v are considered. Deciding which walks to keep is trivial for $s = 2$ [11], but quite intricate for $s \geq 3$. We handle this with the approach originally proposed by Irnich and Villeneuve [18] (the reader is referred to their paper for details). They show that buckets must have size at least $s!$, so the method quickly becomes impractical.

We use three heuristics to accelerate the algorithm. They all find only s-cycle-free q-routes, but not necessarily the minimum one. Hence, whenever they find no negative route, the exact algorithm must verify that none exists.

The first acceleration technique we use is *scaling*, which considers $g > 1$ units of demand at a time. The exact algorithm is run with a modified demand $d'_v(g) = \lceil d_v/g \rceil$ for every vertex v, and modified capacity $C' = \lfloor C/g \rfloor$ for the vehicles. The running time will be proportional to C' instead of C, but the solutions found will still be valid in the original setting.

The second technique is *sparsification*. Intuitively, small edges in the original graph are more likely to appear in the solution. With that in mind, we pre-compute a set of five disjoint spanning forests using an extension of Kruskal's algorithm similar to the "edge-greedy heuristic" described in [34]. By considering only edges in those trees, the dynamic programming will be limited to a restricted set of neighbors when processing each vertex.

The third acceleration is *bucket pruning*, which consists of storing only s (instead of $s!$) labels per bucket. To minimize the number of valid s-extensions blocked, we keep only labels that differ significantly from each other.

The algorithm starts using all three acceleration techniques. Whenever it fails to find a negative q-route, it turns one of them off. If the algorithm fails with no acceleration, we can safely declare that no negative q-route exists. On the other hand, if only a subset of the heuristics is in use and they do find a negative q-route, other heuristics are turned back on.

Cut Generation. At first, the LP formulation includes only degree constraints; violated cuts are added as needed. Besides the well-known rounded capacity cuts (10) and bound cuts (11), we also use framed capacity, strengthened comb, multistar, partial multistar, generalized multistar and hypotour cuts, all presented in [23]. We use the CVRPSEP package [22] to separate those cuts. Since they are defined in terms of the x_e^* variables, not readily available in the DWM formulation, we convert λ_j^* variables into the corresponding x_e^* variables as needed.

Branching Rule. The algorithm starts by adding columns to the formulation until there are none with negative reduced costs, then it looks for all violated cuts. This is repeated until both column generation and separation fail. If the node cannot be fathomed at this point, we branch.

The branching rule is an adaptation of the rule used in [23]. We choose as branching candidates sets S such that $2 < x^*(\delta(S)) < 4$ and branch by imposing the disjunction $(x(\delta(S)) = 2) \vee (x(\delta(S)) \geq 4)$.

We use strong branching to select which set to branch on. Since computing the actual lower bound for each child node can be very time-consuming, we estimate it by performing a small number of column generation iterations. Only p candidate sets $(5 \leq p \leq \max\{10 - \text{depth}, 5\})$ are evaluated. This limits the time spent on strong branching, especially when the depth is high. Priority is given to sets with smaller values of $|x^*(\delta(S)) - 2.7|/d(S)$. We use 2.7 because constraint $x(\delta(S)) = 2$ tends to have a greater impact on the LP relaxation than $x(\delta(S)) \geq 4$, leading to an imbalance in the branch-and-bound tree. Values closer to 2 than 4 increase the impact of imposing $x(\delta(S)) \geq 4$.

Node Selection and Primal Bounds. We used as initial primal bound the best previously known solution plus one. The node selection strategy chosen was depth first search, because it requires less memory.

Dynamic Choice of Column Generation. It usually pays off to combine the strengths of cut and column generation into a single algorithm. However, in some instances, the increase in the lower bound is not worth the additional time spent. In these cases pure branch-and-cut performs better than branch-and-cut-and-price. We modified our algorithm to handle such cases. In the root node, we start as in a pure branch-and-cut, without any column generation. After that, the root node is solved again with a branch-and-cut-and-price using the column generation with $s = 2$. Then we solve it once again, changing s to 3. For the rest of the branch-and-bound tree, the algorithm picks the strategy with the best balance between running time and bound quality.

4 Computational Experiments

We tested our algorithm on most instances available at `www.branchandcut.org`. All instances with up to 135 vertices were solved to optimality, eighteen of them for the first time. The tests were run on a Pentium IV, 2.4 GHz, with 1GB of RAM. Throughout this section, we will refer to the branch-and-cut-and-price algorithm with dynamic choice of column generation as *Dyn-BCP* and the branch-and-cut-and-price with pre-determined column generation as *BCP*.

Tables 1 and 2 detail the results obtained by Dyn-BCP. Columns **Root LB** and **Root Time** give the lower bound and total CPU time of the root node of the branch-and-bound tree. The running time includes the dynamic choice of column generation and strong branching. The next column represents the size s of the cycles eliminated by the column generation procedure ("–" indicates that column generation was not used). **Tree Size** represents the total number of branch-and-bound nodes explored, and **Total Time** is the total CPU time spent by the algorithm. Finally, **Prev UB** and **Our UB** indicate the best solution value available in the literature and the one obtained by our procedure. Values in bold indicate proven optimality.

Table 3 compares several different lower bounds obtained for the CVRP. Lower bounds L1, L2 and L3 (defined in Sect. 2) are presented in the first three columns. Column **LLE03** shows the lower bounds obtained by [23], which are the best available in the literature. The next three columns contain the lower bounds obtained at the root node of the BCP with s-cycle elimination (for $s = 2, 3, 4$). Column **OPT** is the optimal solution value and **T(3)** is the total CPU time (in seconds) spent at the root node of BCP for $s = 3$, the value most commonly used. On average, the algorithm for $s = 4$ was 61% slower than for $s = 3$, which in turn was 12% slower than for $s = 2$. The last line shows the average gap of each bound, considering only instances with a known optimum.

Table 3 shows that major improvements can be obtained by using the BCP approach instead of pure branch-and-cut. Also, eliminating longer cycles often increases the lower bounds significantly.

The greatest improvements usually happen when the ratio n/K is smallest. In such cases, columns without small cycles are very close to being actual routes, so one can expect the duality gap to be reduced. Figure 1, created from the instances

Table 1. Results of the Dyn-BCP algorithm for the A and B instances.

Instance	Root LB	Root Time (s)	*s*	Tree Size	Total Time (s)	Prev. UB	Our UB
A-n37-k5	664.8	16	–	8	19	**669**	**669**
A-n37-k6	932.6	30	3	74	379	**949**	**949**
A-n38-k5	716.5	12	–	52	26	**730**	**730**
A-n39-k5	816.6	107	3	11	167	**822**	**822**
A-n39-k6	822.8	39	3	11	98	**831**	**831**
A-n44-k6	934.8	52	2	6	90	**937**	**937**
A-n45-k6	938.1	52	3	11	170	**944**	**944**
A-n45-k7	1139.3	88	3	26	331	**1146**	**1146**
A-n46-k7	914.0	63	2	3	92	**914**	**914**
A-n48-k7	1069.1	72	3	8	166	**1073**	**1073**
A-n53-k7	1003.9	138	3	16	611	**1010**	**1010**
A-n54-k7	1153.9	125	3	90	1409	**1167**	**1167**
A-n55-k9	1067.4	32	3	7	84	**1073**	**1073**
A-n60-k9	1344.4	161	3	224	3080	1354	**1354**
A-n61-k9	1022.5	108	3	121	1883	**1034**	**1034**
A-n62-k8	1280.4	722	3	101	3102	1290	**1288**
A-n63-k9	1607.0	238	3	49	1046	1616	**1616**
A-n63-k10	1299.1	136	3	387	4988	1315	**1314**
A-n64-k9	1385.3	265	3	648	11254	1402	**1401**
A-n65-k9	1166.5	154	3	17	516	**1174**	**1174**
A-n69-k9	1141.4	289	3	391	7171	1159	**1159**
A-n80-k10	1754.0	1120	3	153	6464	1763	**1763**
B-n38-k6	800.2	10	–	14	14	**805**	**805**
B-n39-k5	549.0	3	–	1	3	**549**	**549**
B-n41-k6	826.4	13	–	8	18	**829**	**829**
B-n43-k6	733.7	13	–	74	29	**742**	**742**
B-n44-k7	909.0	9	–	1	9	**909**	**909**
B-n45-k5	747.5	10	–	19	16	**751**	**751**
B-n45-k6	677.5	224	3	3	279	**678**	**678**
B-n50-k7	741.0	5	–	3	6	**741**	**741**
B-n50-k8	1295.0	97	3	287	2845	**1312**	**1312**
B-n51-k7	1025.2	16	–	83	46	**1032**	**1032**
B-n52-k7	745.8	7	–	9	9	**747**	**747**
B-n56-k7	704.0	15	–	9	22	**707**	**707**
B-n57-k7	1149.2	76	–	133	168	**1153**	**1153**
B-n57-k9	1596.0	61	3	15	193	**1598**	**1598**
B-n63-k10	1479.4	231	–	501	682	**1496**	**1496**
B-n64-k9	859.3	70	–	7	86	**861**	**861**
B-n66-k9	1307.5	145	3	126	1778	**1316**	**1316**
B-n67-k10	1024.4	218	–	327	568	**1032**	**1032**
B-n68-k9	1263.0	260	3	5912	87436	1275*	**1272**
B-n78-k10	1215.2	193	3	90	1053	**1221**	**1221**

* An earlier version of our algorithm [16] found a solution of 1272 but could not prove its optimality. Using that information, K. Wenger [33] solved this instance.

Table 2. Results of the Dyn-BCP algorithm for the E, F, M and P instances.

Instance	Root LB	Root Time (s)	*s*	Tree Size	Total Time (s)	Prev. UB	Our UB
E-n13-k4	247.0	0	–	1	0	**247**	**247**
E-n22-k4	375.0	0	–	1	0	**375**	**375**
E-n23-k3	569.0	0	–	1	0	**569**	**569**
E-n30-k3	533.3	7	–	6	7	**534**	**534**
E-n31-k7	378.5	4	2	2	6	**379**	**379**
E-n33-k4	834.5	14	–	5	15	**835**	**835**
E-n51-k5	518.2	51	–	8	65	**521**	**521**
E-n76-k7	670.0	264	2	1712	46520	**682**	**682**
E-n76-k8	726.5	277	2	1031	22891	**735**	**735**
E-n76-k10	817.4	354	3	4292	80722	830	**830**
E-n76-k14	1006.5	224	3	6678	48637	1021	**1021**
E-n101-k8	805.2	1068	3	11622	801963	**815**	**815**
E-n101-k14	1053.8	658	3	5848	116284	1071	**1067**
F-n45-k4	724.0	8	–	1	8	**724**	**724**
F-n72-k4	236.4	70	–	42	121	**237**	**237**
F-n135-k7	1159.9	6618	–	25	7065	**1162**	**1162**
M-n101-k10	820.0	119	–	1	119	**820**	**820**
M-n121-k7	1031.1	5594	3	40	25678	1034	**1034**
P-n16-k8	449.0	1	2	3	1	**450**	**450**
P-n19-k2	212.0	1	–	1	1	**212**	**212**
P-n20-k2	213.0	1	–	9	1	**216**	**216**
P-n21-k2	211.0	1	–	1	1	**211**	**211**
P-n22-k2	216.0	2	–	2	2	**216**	**216**
P-n22-k8	603.0	3	2	1	3	**603**	**603**
P-n23-k8	529.0	18	2	1	18	**529**	**529**
P-n40-k5	456.9	28	–	5	34	**458**	**458**
P-n45-k5	506.6	59	3	11	194	**510**	**510**
P-n50-k7	551.5	79	3	7	143	**554**	**554**
P-n50-k8	616.3	102	3	1084	9272	649	**631**
P-n50-k10	689.3	50	3	65	304	**696**	**696**
P-n51-k10	735.2	35	3	22	105	**741**	**741**
P-n55-k7	557.9	90	2	450	4649	**568**	**568**
P-n55-k8	579.8	42	2	196	1822	**588**	**588**
P-n55-k10	681.4	107	3	1556	9076	699	**694**
P-n55-k15	972.8	251	3	398	1944	993	**989**
P-n60-k10	738.9	126	3	52	570	756	**744**
P-n60-k15	962.8	118	3	76	442	1033	**968**
P-n65-k10	786.0	159	3	23	422	792	**792**
P-n70-k10	814.5	292	3	1752	24039	834	**827**
P-n76-k4	588.8	363	–	59	572	**593**	**593**
P-n76-k5	616.8	273	–	3399	14546	**627**	**627**
P-n101-k4	678.5	1055	–	23	1253	**681**	**681**

Table 3. Comparison of lower bounds.

Instance	Lower Bounds							OPT	T(3)
	L1	L2	L3	LLE03	$s = 2$	$s = 3$	$s = 4$		
A-n53-k7	996.6	978.5	1002.2	998.7	1001.9	1003.8	1004.1	**1010**	29
A-n54-k7	1130.7	1114.0	1150.0	1135.3	1150.9	1153.6	1156.1	**1167**	26
A-n55-k9	1055.9	1025.4	1066.4	1058.3	1066.4	1067.3	1067.6	**1073**	12
A-n60-k9	1316.5	1305.6	1341.6	1319.6	1341.6	1344.4	1345.3	**1354**	27
A-n61-k9	1004.8	996.8	1018.6	1010.2	1018.9	1022.7	1023.3	**1034**	19
A-n62-k8	1244.1	1222.7	1274.1	1251.7	1273.7	1280.5	1281.1	**1288**	36
A-n63-k9	1572.2	1564.8	1603.5	1580.7	1603.6	1607.0	1608.9	**1616**	34
A-n63-k10	1262.2	1267.4	1294.5	1266.6	1294.4	1299.1	1302.5	**1314**	28
A-n64-k9	1340.1	1353.3	1378.9	1351.6	1379.0	1385.3	1389.2	**1401**	39
A-n65-k9	1151.1	1133.0	1163.4	1155.2	1164.4	1166.5	1168.3	**1174**	31
A-n69-k9	1108.9	1113.2	1138.4	1114.4	1139.3	1141.4	1144.1	**1159**	87
A-n80-k10	1699.9	1712.2	1749.7	1709.6	1749.6	1753.9	1755.6	**1763**	940
B-n50-k7	740.0	664.8	741.0	741.0	741.0	741.0	741.0	**741**	8
B-n50-k8	1279.2	1217.5	1291.8	1281.1	1291.8	1295.2	1302.0	**1312**	20
B-n51-k7	1024.6	918.4	1025.9	1025.6	1026.0	1026.4	1026.6	**1032**	21
B-n52-k7	745.0	640.2	746.3	746.0	746.4	746.4	746.4	**747**	17
B-n56-k7	703.4	606.9	704.5	705.0	704.5	705.0	705.0	**707**	19
B-n57-k7	1148.6	1058.5	1150.9	1150.1	1151.1	1151.6	1152.3	**1153**	126
B-n57-k9	1586.7	1511.5	1595.2	1589.2	1595.2	1595.8	1596.1	**1598**	16
B-n63-k10	1478.9	1418.4	1484.2	1481.0	1484.3	1487.2	1487.3	**1496**	32
B-n64-k9	858.5	769.3	860.1	860.5	860.9	861.0	860.5	**861**	44
B-n66-k9	1295.2	1223.1	1302.6	1298.5	1303.7	1307.5	1308.5	**1316**	58
B-n67-k10	1023.8	984.5	1026.4	1024.8	1026.4	1026.9	1027.2	**1032**	25
B-n68-k9	1256.8	1163.9	1261.5	1258.1	1261.6	1262.9	1263.5	**1272**	42
B-n78-k10	1202.3	1124.5	1212.5	1205.6	1212.7	1214.8	1215.7	**1221**	63
E-n51-k5	514.5	512.9	518.0	519.0	519.1	519.1	519.4	**521**	23
E-n76-k7	661.4	663.3	668.4	666.4	670.3	670.8	670.9	**682**	73
E-n76-k8	711.2	716.7	725.1	717.9	726.5	726.8	727.0	**735**	72
E-n76-k10	789.5	811.4	816.5	799.9	817.3	817.4	817.7	**830**	36
E-n76-k14	948.1	999.6	1004.8	969.6	1004.9	1006.5	1007.5	**1021**	15
E-n101-k8	796.4	786.4	801.8	802.6	804.6	805.2	805.2	**815**	320
E-n101-k14	1008.3	1045.1	1051.6	1026.9	1052.2	1053.9	1054.3	**1067**	69
M-n101-k10	819.5	798.1	820.0	820.0	820.0	820.0	820.0	**820**	31
M-n121-k7	1009.7	1013.0	1030.9	1017.4	1031.2	1031.8	1032.0	**1034**	1257
P-n50-k8	596.9	612.5	615.3	602.1	615.8	616.3	617.1	**631**	13
P-n55-k10	646.7	677.2	680.1	662.1	680.2	681.4	681.4	**694**	6
P-n55-k15	895.1	963.0	967.5	906.7	967.7	973.1	973.2	**989**	7
P-n60-k10	708.3	733.5	737.2	718.4	737.7	739.0	739.3	**744**	8
P-n60-k15	903.3	955.6	961.2	929.8	961.7	962.9	963.5	**968**	4
P-n65-k10	756.5	779.8	785.2	767.1	785.6	786.0	787.1	**792**	13
P-n70-k10	786.9	808.3	812.7	795.6	813.6	814.6	814.9	**827**	23
AvgGAP	**2.92%**	**4.76%**	**1.02%**	**2.26%**	**0.97%**	**0.82%**	**0.74%**	—	—

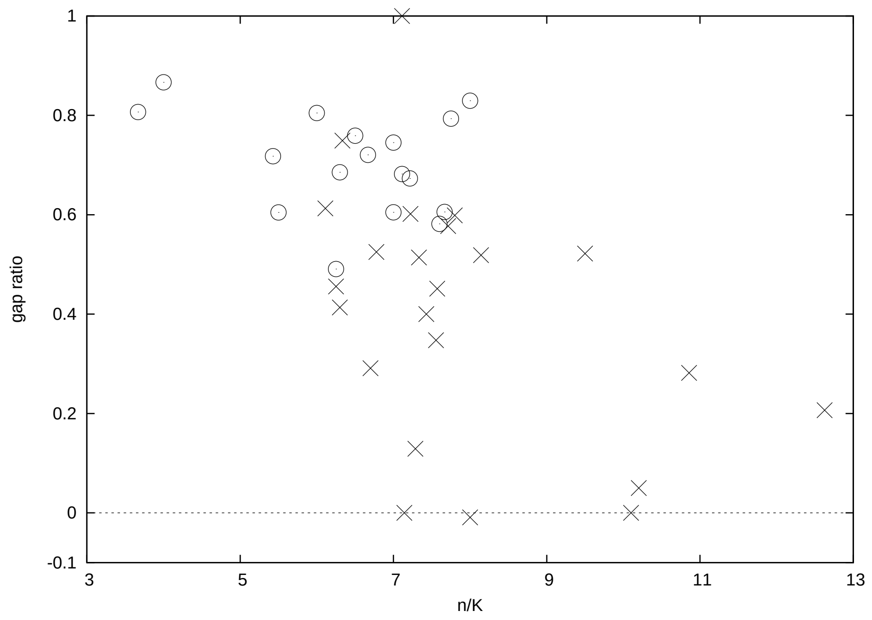

Fig. 1. Gap improvement of BCP with $s = 3$ ($s3$) with respect to [23] (lle). The graph represents $\frac{s3-lle}{opt-lle}$ as a function of $\frac{n}{K}$. Circles represent instances solved to optimality for the first time and crosses the ones already closed.

of Table 3 with a known optimal solution, illustrates this observation. The horizontal axis represents the ratio n/K. The value plotted is $(s3 - lle)/(opt - lle)$, where $s3$ is the bound of the BCP with $s = 3$, lle is the LLE03 bound and opt is the optimal solution value. This ratio represents the reduction of the duality gap relative to the LLE03 gap. The value is zero whenever both methods produce the same result; positive values mean that BCP gaps are smaller, and negative values favor LLE03. Circles represent instances that were solved by our algorithm for the first time, while crosses indicate instances that were already closed. Only one instance (B-n56-k7) had a negative value (-0.009). The gap was often reduced by more than 50%, enough to close several open instances.

5 Conclusion

This paper has shown that a careful combination of column and cut generation leads to a very effective algorithm for the CVRP. We consider the three main aspects that can lead to further improvements: cut generation, column generation, and the integration between them.

Consider cutting planes. Rounded capacity cuts are highly effective in our BCP—the efficient heuristic separation for them contributes significantly to the good overall performance. We did try all other families of cuts for the CVRP known to be effective in a pure BC: framed capacities, generalized capacities, strengthened combs, multistars, and extended hypotours. Surprisingly, however, their effect was quite modest in practice—only on E-n101-k8 the root bound increased by more than two units. Any improvement to the cutting part of our BCP will therefore require new families of inequalities. Unfortunately, looking for facet-defining inequalities may not be enough within the BCP framework, since a facet of the CVRP polyhedron can also be a facet of the polyhedron induced by column generation over the q-routes.[6] A new family of inequalities will only be effective in our context if it significantly cuts polyhedron P_2.

In the short term, improvements are more likely to come from column generation. The goal is to devise efficient ways of pricing over a set of columns as close to cycle-free q-routes as possible. The s-cycle elimination approach we chose was reasonably effective, at least for $s \leq 4$. But other alternatives to further restrict the q-routes can also be tried. One could, for instance, forbid q-routes that visit vertices in a small set S more than once. This set could be dynamically chosen in order to eliminate q-routes with long cycles that are particularly harmful to bound quality.

In principle, several of the combinatorial relaxations of the CVRP—such as k-degree center trees, K-trees, or b-matchings—could also be used instead of (or even in addition to) q-routes. We think they are unlikely to have a noticeable impact, for two reasons. First, among the relaxations mentioned above, only q-routes take client demands and vehicle capacities into account. These "numerical" elements are precisely what makes the CVRP much harder in practice than a "pure" graph problem such as the TSP. It seems that known families of inequalities for the CVRP, mostly inspired by previous work on the TSP, cannot cope well with this numerical aspect. The inequalities implicitly given by P_2 can do better. The second reason for not using the alternatives above is that only q-routes lead to a pricing problem that is superpolynomial, but still practical. The polyhedra associated with polynomial pricing problems can be fully described and efficiently separated, which usually makes a pure BC (instead of BCP) a faster alternative. For instance, we can separate over the b-matching polyhedron in a very efficient way [21]. Separation over the q-route polyhedron, on the other hand, is possible only implicitly, with column generation.

Even if we leave column and cut generation as they are, we can still accelerate the algorithm through better integration. The task of designing, coding and fine-tuning a high performance branch-and-cut or branch-and-price procedure is complex, requiring experience to choose among several possible strategies, and, often, some computational tricks. BCP multiplies the possibilities and, therefore, the difficulties. We believe that there is still plenty of room for such improvements in our BCP.

[6] A. Letchford proved that generalized large multistar inequalities do not cut the polyhedron induced by *cycle-free* q-routes (personal communication). Although this is not necessarily true when using q-routes without s-cycles, the heuristic separation in our BCP never found a violated GL multistar.

Acknowledgements. RF was supported by NSF grant DMI-0245609, having started this work while at Gapso Inc., Brazil. MPA was funded by research grant 300475/93-4 from CNPq. MR was supported by CAPES. EU was funded by scholarships from UFF and CNPq. RW was supported by NSF grant 11201878-123412. We thank Humberto Longo for his help during implementation and with experiments.

References

1. Achuthan, N., Caccetta, L., Hill, S.: Capacited vehicle routing problem: Some new cutting planes. Asia-Pacific J. of Operational Research **15** (1998) 109–123
2. Achuthan, N., Caccetta, L., Hill, S.: An improved branch-and-cut algorithm for the capacitated vehicle routing problem. Transportation Science **37** (2003) 153–169
3. Araque, J., Hall, L., Magnanti, T.: Capacitated trees, capacitated routing and associated polyhedra. Technical Report SOR-90-12, Princeton University (1990)
4. Araque, J., Kudva, G., Morin, T., Pekny, J.: A branch-and-cut algorithm for the vehicle routing problem. Annals of Operations Research **50** (1994) 37–59
5. Augerat, P.: Approche polyèdrale du problème de tournées de véhicles. PhD thesis, Institut National Polytechnique de Grenoble (1995)
6. Augerat, P., Belenguer, J., Benavent, E., Corberán, A., Naddef, D., Rinaldi, G.: Computational results with a branch and cut code for the capacitated vehicle routing problem. Technical Report 949-M, Université Joseph Fourier, Grenoble, France (1995)
7. Agarwal, Y., Mathur, K., Salkin, H.: A set-partitioning based exact algorithm for the vehicle routing problem. Networks **19** (1989) 731–739
8. Balinski, M., Quandt, R.: On an integer program for a delivery problem. Operations Research **12** (1964) 300–304
9. Blasum, U., Hochstättler, W.: Application of the branch and cut method to the vehicle routing problem. Technical Report ZPR2000-386, Zentrum fur Angewandte Informatik Köln (2000)
10. Christofides, N., Eilon, S.: An algorithm for the vehicle-dispatching problem. Operational Research Quarterly **20** (1969) 309–318
11. Christofides, N., Mingozzi, A., Toth, P.: Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. Mathematical Programming **20** (1981) 255–282
12. Cornuéjols, G., Harche, F.: Polyhedral study of the capacitated vehicle routing problem. Mathematical Programming **60** (1993) 21–52
13. Dantzig, G., Ramser, R.: The truck dispatching problem. Management Science **6** (1959) 80–91
14. Fisher, M.: Optimal solution of vehicle routing problem using minimum k-trees. Operations Research **42** (1994) 626–642
15. Fukasawa, R., Poggi de Aragão, M., Porto, O., Uchoa, E.: Robust branch-and-cut-and-price for the capacitated minimum spanning tree problem. In: Proc. of the International Network Optimization Conference, Evry, France (2003) 231–236
16. Fukasawa, R., Reis, M., Poggi de Aragão, M., Uchoa, E.: Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Technical Report RPEP Vol.3 no.8, Universidade Federal Fluminense, Engenharia de Produção, Niterói, Brazil (2003)

17. Hadjiconstantinou, E., Christofides, N., Mingozzi, A.: A new exact algorithm from the vehicle routing problem based on q-paths and k-shortest paths relaxations. In Laporte, G., Gendreau, M., eds.: Freight Transportation. Number 61 in Annals of Operations Research. Baltzer Science Publishers (1995) 21–44
18. Irnich, S., Villeneuve, D.: The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. Unpublished manuscript (2003)
19. Laporte, G., Norbert, Y.: A branch and bound algorithm for the capacitated vehicle routing problem. Operations Research Spektrum **5** (1983) 77–85
20. Letchford, A., Eglese, R., Lysgaard, J.: Multistars, partial multistars and the capacitated vehicle routing problem. Mathematical Programming **94** (2002) 21–40
21. Letchford, A., Reinelt, G., Theis, D.: A faster exact separation algorithm for blossom inequalities. This volume (2004)
22. Lysgaard, J.: CVRPSEP: A package of separation routines for the capacitated vehicle routing problem (2003) Available at `www.asb.dk/~lys`.
23. Lysgaard, J., Letchford, A., Eglese, R.: A new branch-and-cut algorithm for the capacitated vehicle routing problem. Mathematical Programming (2003) (To appear).
24. Martinhon, C., Lucena, A., Maculan, N.: A relax and cut algorithm for the vehicle routing problem. European J. of Operational Research (2003) (To appear).
25. Miller, D.: A matching based exact algorithm for capacitated vehicle routing problems. ORSA J. on Computing **7** (1995) 1–9
26. Naddef, D., Rinaldi, G.: Branch-and-cut algorithms for the capacitated VRP. In Toth, P., Vigo, D., eds.: The Vehicle Routing Problem. SIAM (2002) 53–84
27. Pigatti, A.: Modelos e algoritmos para o problema de alocação generalizada e aplicações. Master's thesis, Pontifícia Universidade Católica do Rio de Janeiro, Brazil (2003)
28. Poggi de Aragão, M., Uchoa, E.: Integer program reformulation for robust branch-and-cut-and-price. In: Annals of Mathematical Programming in Rio, Búzios, Brazil (2003) 56–61
29. Ralphs, T.: Parallel branch and cut for capacitated vehicle routing. Parallel Computing **29** (2003) 607–629
30. Ralphs, T., Kopman, L., Pulleyblank, W., Jr., L.T.: On the capacitated vehicle routing problem. Mathematical Programming **94** (2003) 343–359
31. Toth, P., Vigo, D.: Models, relaxations and exact approaches for the capacitated vehicle routing problem. Discrete Applied Mathematics **123** (2002) 487–512
32. Toth, P., Vigo, D.: The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications. SIAM (2002)
33. Wenger, K.: Generic Cut Generation Methods for Routing Problems. PhD thesis, Institute of Computer Science, University of Heidelberg (2003)
34. Werneck, R.F., Setubal, J.C.: Finding minimum congestion spanning trees. ACM J. of Experimental Algorithmics **5** (2000)