Robust Branch-Cut-and-Price Algorithms for Vehicle Routing Problems

Artur Pessoa[1], Marcus Poggi de Aragão[2] and Eduardo Uchoa[1]

[1] Departamento de Engenharia de Produção
Universidade Federal Fluminense
R. Passo da Pátria, 156 24210-240, Niterói, Brazil
artur@producao.uff.br, uchoa@producao.uff.br

[2] Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
R. Marquês de São Vicente, 225 22453-900, Rio de Janeiro, Brazil
poggi@inf.puc-rio.br

Summary. This article presents techniques for constructing robust Branch-Cut-and-Price algorithms on a number of Vehicle Routing Problem variants. The word "robust" stress the effort of controlling the worst-case complexity of the pricing subproblem, keeping it pseudo-polynomial. Besides summarizing older research on the topic, some promising new lines of investigation are also presented, specially the development of new families of cuts over large extended formulations. Computational experiments over benchmark instances from ACVRP, COVRP, CVRP and HFVRP variants are provided.

Key words: Column generation; cutting plane algorithms; extended formulations; branch-and-bound.

1 Introduction

Branch-and-cut and branch-and-price were established in the 1980's as two of the most important techniques in integer programming and combinatorial optimization. Of course, some people considered the possibility of combining the strengths of both techniques into a more powerful Branch-Cut-and-Price (BCP) algorithm. However, for some time this was not considered to be practical [8], since the new dual variables corresponding to separated cuts would have the undesirable effect of changing the structure of the pricing subproblem, making it potentially intractable. However, in the late 1990's, some researchers [7, 15, 26, 27, 37, 38] independently noted that cuts expressed in terms of variables from a suitable original formulation could be dynamically separated, translated and added to the master problem. Those cuts would

not change the structure of the pricing subproblem. This last property defines what Poggi de Aragão and Uchoa called *Robust* Branch-Cut-and-Price (RBCP) algorithms [35].

Robustness is a desirable property of a BCP. There is an asymmetry between the cutting and pricing operations in that kind of algorithm. If the separation subproblem for some family of cuts happens to be intractable, heuristics may be used. When the heuristics fail, some violated cuts may have been missed, but one certainly has a valid dual bound. There is no need to ever call an exact separation. On the other hand, even with good pricing heuristics, at least one call to the exact pricing is necessary to establish a valid dual bound. If the addition of cuts changes the structure of the pricing, making it unpredictably harder, there is a risk of having to solve to optimality an intractable subproblem. In such a case the whole algorithm would fail.

A RBCP for the Capacitated Vehicle Routing Problem (CVRP) was presented in Fukasawa et al. [17, 18]. It combined column generation over q-routes with cuts over the edge CVRP formulation. A *q-route* is a walk that starts at the depot vertex, traverses a sequence of client vertices with total demand at most equal to the capacity, and returns to the depot. While pricing actual CVRP routes – capacitated elementary cycles – would lead to a strongly NP-hard problem, q-routes can be priced in pseudo-polynomial time by dynamic programming. Since the values of new dual variables corresponding to cuts over the edge variables can be translated into dynamic programming costs, the pricing is guaranteed to remain tractable in that RBCP. The algorithm actually showed a steady behavior and could solve all benchmark instances from the literature with up to 135 vertices, improving significantly upon previous algorithms. It should be remarked that a RBCP algorithm by Kohl et al. [27] had already been successful on the Vehicle Routing Problem with Time Windows (VRPTW), consistently solving tightly constrained instances (those with narrow time windows) with up to 100 clients.

While it is clear that VRP algorithms have a lot to gain from the combination of cutting and pricing, it is interesting (and somehow surprisingly to us) to know that successful *non-robust* BCPs were also proposed recently. The VRPTW algorithm by Jepsen et al. [24] uses clique and odd-hole cuts defined over the variables of the master problem. The complicated strongly NP-hard pricing is tackled by a dynamic programming that tries to avoid the combinatorial explosion using clever state dominance rules. Some formerly open instances could be solved with almost no branching, but the algorithm failed completely on a few other instances already solved by other authors. The CVRP algorithm by Baldacci et al. [6] also uses cuts over the master variables, including cliques. Besides having a good dynamic programming to handle the complicated pricing, there is another crucial idea. A sequence of cheaper lower bounding procedures produces good estimates of the optimal dual variable values. The non-robust BCP is called with the dual variables bounded to be above and close to the estimates, similarly to what is done in stabilized column generation [14]. Convergence (to a bound slightly below the

theoretical optimal) is obtained with very few calls to the unpredictably expensive pricing. The overall algorithm could solve almost all instances solved by Fukasawa et al., usually taking much less time.

Anyway, this article is about *Robust* BCP algorithms for some VRP variants with only capacity constraints. The main ideas will be presented in the Asymmetric Capacitated Vehicle Routing Problem (ACVRP). As shown in the sequence, a BCP for the ACVRP can be easily adapted to its symmetric counterpart (CVRP) and also to Open (COVRP), and Heterogeneous Fleet (HFVRP) VRP variants. Besides presenting techniques similar to those utilized in [17], we also introduce a discussion about promising new families of cuts defined over a pseudo-polynomially large extended formulation. We may view those extended cuts as lying at the frontier of robustness, i.e., they are the most general kind of cuts that can be added without disturbing the dynamic programming pricing of q-routes.

2 Formulations and Valid Inequalities for the ACVRP

Let $G = (V, A)$ be a directed graph with vertices $V = \{0, 1, \ldots, n\}$ and $m = |A|$ arcs. Vertex 0 is the *depot*, other vertices are *clients*. Each client vertex i is associated with a positive integer demand $d(i)$. Depot demand $d(0)$ is defined as zero. Each arc $a \in A$ has a nonnegative cost c_a. Given a positive integer C greater than or equal to the maximum demand, the *Asymmetric Capacitated Vehicle Routing Problem* (ACVRP) problem consists of finding a set of routes satisfying the following constraints: (i) each route starts and ends at the depot, (ii) each client is visited by a single route, and (iii) the total demand of all clients in any route is at most C. The goal is to minimize total route cost. In many cases there are lower and/or upper bounds on the number of routes. We omit those possible additional constraints for sake of simplicity.

2.1 Arc Formulation

The arc formulation (a.k.a. two-index formulation) for the ACVRP uses binary variables x_a to indicate whether arc a belongs to the optimal solution. Define $V_+ = \{1, \ldots, n\}$ as the set of all clients. For any set $S \subseteq V$, we let $d(S) = \sum_{i \in S} d(i)$, $k(S) = \lceil d(S)/C \rceil$, $A(S) = \{(i, j) \in A : i, j \in S\}$, $\delta^-(S) = \{(i, j) \in A : i \in V \setminus S, j \in S\}$, and $\delta^+(S) = \{(i, j) \in A : i \in S, j \in V \setminus S\}$. Let $x(A') = \sum_{a \in A'} x_a$ for any $A' \subseteq A$. The formulation follows:

$$\text{Minimize} \quad \sum_{a \in A} c_a x_a \tag{1a}$$

S.t.

$$x(\delta^-(\{i\})) = 1 \quad (\forall\, i \in V_+), \tag{1b}$$

$$x(\delta^+(\{i\})) = 1 \quad (\forall\, i \in V_+), \tag{1c}$$

$$x(\delta^-(S)) \geq k(S) \ (\forall\, S \subseteq V_+), \tag{1d}$$

$$x_a \in \{0,1\} \quad (\forall\, a \in A). \tag{1e}$$

The *in-degree* constraints (1b) state that exactly one arc must enter each non-depot vertex. Similarly, the *out-degree* constraints (1c) state that exactly one arc must leave each client vertex. The *rounded capacity cuts* (1d) state that at least $k(S)$ arcs must enter each set S. Even for other VRP variants, inequalities of the form

$$x(\delta^-(S)) \geq K(S) \ (\forall\, S \subseteq V_+), \tag{2}$$

where $K(S)$ is a lower bound on the minimum number of vehicles necessary to cover the clients in set S will be called *capacity cuts*. For example, the path inequalities for the VRPTW [25, 27] can be viewed as capacity cuts.

The arc formulation can be improved by adding an exponential number of variables corresponding to the q-routes. Number all possible q-routes from 1 to p. Define q_a^j as the number of times arc a appears in j-th q-route and λ_j as the positive variable associated to that q-route.

$$\text{Minimize} \quad \sum_{a \in A} c_a x_a \tag{3a}$$

S.t.

$$\sum_{j=1}^{p} q_a^j \lambda_j - x_a = 0 \ (\forall\, a \in A), \tag{3b}$$

$$x(\delta^-(\{i\})) = 1 \quad (\forall\, i \in V_+), \tag{3c}$$

$$x(\delta^-(S)) \geq k(S) \ (\forall\, S \subseteq V_+), \tag{3d}$$

$$\lambda_j \geq 0 \quad (j = 1, \ldots, p), \tag{3e}$$

$$x_a \in \{0,1\} \quad (\forall\, a \in A). \tag{3f}$$

This formulation includes all variables and constraints from the arc formulation, but new constraints (3b) impose that x must also be a weighted sum of arc-incidence vectors of q-routes. This restriction leads to a significantly stronger formulation. Note that out-degree constraints (1c) are implied by (3b) and (3c).

Pricing the λ variables requires the solution of minimum cost q-route problems, which can be solved in $O(n^2C)$ time. Houck et al. [22] and Christofides et al. [11] noticed that one can find minimum cost q-routes without 2-cycles (subpaths $i \to j \to i$, $i \neq 0$) without changing this complexity. This immediately leads to a stronger formulation. Minimum cost q-routes without s-cycles

can be found in $O(s!s^2n^2C)$ time [23, 17], which is still pseudo-polynomial for fixed s. Of course, larger values of s give stronger formulations. Pricing q-routes without any sub-cycle (i.e., elementary routes) is a strongly NP-hard problem.

When solving the linear relaxation of (3) by column and row generation, a more compact Master LP is obtained if every occurrence x_a in (3c)–(3d) is replaced by its equivalent given by (3b). The resulting LP will be referred to as the *Dantzig-Wolfe Master* (DWM):

$$\text{Minimize} \quad \sum_{j=1}^{p} \Big(\sum_{a \in A} c_a q_a^j \Big) \lambda_j \tag{4a}$$

S.t.

$$\sum_{j=1}^{p} \Big(\sum_{a \in \delta^-(\{i\})} q_a^j \Big) \lambda_j = 1 \quad (\forall\, i \in V_+), \tag{4b}$$

$$\sum_{j=1}^{p} \Big(\sum_{a \in \delta^-(S)} q_a^j \Big) \lambda_j \geq k(S) \ (\forall\, S \subseteq V_+), \tag{4c}$$

$$\lambda_j \geq 0 \qquad (j = 1, \ldots, p). \tag{4d}$$

The reduced cost of a λ variable is the sum of the reduced costs of the arcs in the corresponding q-route. Let ω and π be the dual variables associated with constraints (4b) and (4c), respectively. The reduced cost $\bar{c}_a$ of an arc a is given by:

$$\bar{c}_a = c_a - \omega_j - \sum_{S | a \in \delta^-(S)} \pi_S \ (\forall a = (i,j) \in A). \tag{5}$$

Capacity Cuts are not the only ones that can appear in the DWM. A generic cut $\sum_{a \in A} \alpha_a x_a \geq b$ can be included as $\sum_{j=1}^{p} (\sum_{a \in A} \alpha_a q_a^j) \lambda_j \geq b$. This cut contributes to the computation of $\bar{c}_a$ with the value $-\alpha_a \beta$, where β is the new dual variable. The addition of cuts to the DWM affects the reduced costs but not the structure of the subproblem.

The possibility of adding such extra cuts naturally leads to the question of which cuts to add. Many cuts valid for the symmetric cost case (CVRP) are known [1, 2, 3, 4, 12, 28, 33, 32]. They are defined over undirected edge variables, but can also be used on the ACVRP by replacing each such variable by the corresponding pair of opposite arcs. There are directed cuts devised for the VRPTW [25, 31], but, as far as we know, no cuts were specifically suggested for the ACVRP. However, cuts for the directed arc formulation of the capacitated minimum spanning tree problem (CMST) [2, 21] can be applied on the ACVRP. For example, the following cuts, known as *root cutsets*, are valid for the ACVRP. Define $S_\alpha = \{i \in V \setminus S : k(S \cup \{i\}) = k(S)\}$ and $S_\beta = (V \setminus S) \setminus S_\alpha$. Note that the depot always belongs to S_α.

$$\tfrac{k(S)+1}{k(S)} x(\delta^-(S) \cap \delta^+(S_\alpha)) + x(\delta^-(S) \cap \delta^+(S_\beta)) \geq k(S) + 1 \ (\forall\, S \subseteq V_+). \tag{6}$$

Those constraints are actually a strengthening of the rounded capacity cuts, based on the observation that if one of the routes covering S comes from a higher demand client in S_β, at least $k(S)+1$ routes must enter S.

Unhappily, computational experiments have shown that adding other known arc or edge inequalities other than the capacity cuts only improve bounds modestly. For example, consider the CVRP algorithm in [17]. In that case, several complex families of cuts known to be effective in a pure branch-and-cut algorithm [32] were separated: framed capacities, generalized capacities, strengthened combs, multistars, and extended hypotours. Surprisingly, however, the resulting bounds were not significantly better than those obtained by only separating rounded capacity cuts. A possible explanation is that most such cuts are already implicitly given by the q-route structure used in the column generation part. This explanation received some theoretical support. Letchford and Salazar [30] proved that generalized large multistar inequalities are indeed implied by the q-route definition.

In order to improve significantly over the bounds given by (4) in a robust way, we can look for other families of cuts, radically different from those currently used on the arc or edge formulations.

2.2 Introducing Capacity-Indexed Variables

There is an extended formulation for the ACVRP similar to a formulation by Picard and Queyranne [34] for the time-dependant TSP. Godinho et al. [19] used it for the case of unitary demands. Let binary variables x_a^d indicate that arc $a=(i,j)$ belongs to a route and that the total demand of j and of the vertices following j in the route is exactly d. The arcs returning to the depot must have $d=0$.

$$\text{Minimize} \quad \sum_{a\in A} c_a \sum_{d=0}^{C} x_a^d \tag{7a}$$

S.t.

$$\sum_{a\in\delta^-(\{i\})} \sum_{d=1}^{C} x_a^d = 1 \qquad (\forall\, i \in V_+), \tag{7b}$$

$$\sum_{a\in\delta^+(\{i\})} \sum_{d=1}^{C} x_a^d = 1 \qquad (\forall\, i \in V_+), \tag{7c}$$

$$\sum_{a\in\delta^-(\{i\})} x_a^d - \sum_{a\in\delta^+(\{i\})} x_a^{d-d(i)} = 0 \quad (\forall\, i \in V_+;\ d = d(i),\ldots,C), \tag{7d}$$

$$x_a^d \in \{0,1\} \qquad (\forall\, a \in A;\ d = 1,\ldots,C), \tag{7e}$$

$$x_{(i,0)}^d = 0 \qquad (\forall\, i \in V_+;\ d = 1,\ldots,C). \tag{7f}$$

Equations (7b) and (7c) are in-degree and out-degree constraints. Equations (7d) state that if an arc with index d enters vertex i then an arc with index $d-d(i)$ must leave i. This both prevents cycles and routes with total demand

greater than C. Variables with index distinct from 0 to the depot can be removed. Note that variables x_{ij}^d with $d > C - d(i)$ can also be removed. To provide a more simple and precise notation of this formulation, we define a directed multigraph $G_C = (V, A_C)$, where A_C contains arcs $(i,j)^d$, for each $(i,j) \in A$, $d = 1, \ldots, C - d(i)$, and $(i,0)^0$, $i \in V_+$. When working with variables x_a^d it is assumed that $\delta^-(S)$ and $\delta^+(S)$ are the subsets of arcs in A_C, with any index, entering and leaving S, respectively. Denote by $\delta_d^-(S)$ and $\delta_d^+(S)$ the sets of arcs with index d entering and leaving S. For example, equations (7b) will be written as:

$$\sum_{a^d \in \delta^-(\{i\})} x_a^d = 1 \ (\forall\, i \in V_+). \tag{8}$$

The linear relaxation of this formulation yields a weak bound, the same of a column generation over the q-routes without any cycle elimination (i.e., it is equivalent to a formulation with constraints (3b), (3c), (3e), and (3f)). Therefore, using the capacity-indexed formulation directly in a branch-and-bound algorithm is not interesting. However, this formulation may be useful in a branch-and-cut approach. Of course, since x_a can be defined as the sum of the x_a^d variables, for all existing d, any inequality valid for the arc formulation could be used in that algorithm. But the potential advantage of the capacity-indexed formulation is to allow the derivation and separation of new families of cuts defined over this pseudo-polynomially large extended variable space. Anyway, working directly with this formulation is only practical for small values of capacity, as there are $O(mC)$ variables and $O(nC)$ constraints.

The capacity-indexed formulation can be naturally rewritten in terms of q-routes. Define q_a^{dj} as the number of arcs a carrying exactly d units of capacity in the j-th q-route. Also including the rounded capacity cuts, we get:

$$\text{Minimize} \quad \sum_{a^d \in A} c_a x_a^d \tag{9a}$$

S.t.

$$\sum_{j=1}^{p} q_a^{dj} \lambda_j - x_a^d = 0 \ (\forall\, a^d \in A_C), \tag{9b}$$

$$\sum_{a^d \in \delta^-(\{i\})} x_a^d = 1 \quad (\forall\, i \in V_+), \tag{9c}$$

$$\sum_{a^d \in \delta^-(S)} x_a^d \geq k(S) \ (\forall\, S \subseteq V_+), \tag{9d}$$

$$\lambda_j \geq 0 \qquad (j = 1, \ldots, p), \tag{9e}$$

$$x_a^d \in \{0,1\} \qquad (\forall\, a^d \in A_C). \tag{9f}$$

It can be noted that equalities (7d) are already implied by the definition of q-routes together with (9b). Eliminating the x variables, we can write the Dantzig-Wolfe Master (DWM) as:

$$\text{Minimize} \quad \sum_{j=1}^{p} \Big(\sum_{a^d \in A_C} q_a^{dj} c_a \Big) \lambda_j \tag{10a}$$

S.t.

$$\sum_{j=1}^{p} \Big(\sum_{a^d \in \delta^-(\{i\})} q_a^{dj} \Big) \lambda_j = 1 \quad (\forall\, i \in V_+), \tag{10b}$$

$$\sum_{j=1}^{p} \Big(\sum_{a^d \in \delta^-(S)} q_a^{dj} \Big) \lambda_j \geq k(S) \ (\forall\, S \subseteq V_+), \tag{10c}$$

$$\lambda_j \geq 0 \qquad (j = 1, \ldots, p). \tag{10d}$$

This LP and (4) are exactly the same. But now it is clear that a generic cut i over the extended variables

$$\sum_{a^d \in A_C} \alpha_{ai}^d x_a^d \geq b_i \tag{11}$$

can be also included in the DWM as

$$\sum_{j=1}^{p} \Big(\sum_{a^d \in A_C} \alpha_{ai}^d q_a^{dj} \Big) \lambda_j \geq b_i. \tag{12}$$

Suppose that, at a given instant, we have a total of m constraints (possibly including in-degree and capacity cuts) in the DWM, the i-th constraint having dual variable β_i. The reduced cost of arc a with index d is given by:

$$\bar{c}_a^d = c_a - \sum_{i=1}^{m} \alpha_{ai}^d \beta_i. \tag{13}$$

The above reformulation presents some remarkable features in a branch-cut-and-price context. It allows the introduction of new cuts over the capacity-indexed variables, even for large values of C, without having to explicitly introduce any new variables. This means that the size of the LPs that are actually solved is basically unchanged. Moreover, those new cuts are robust with respect to the pricing of q-routes. This means that computing a minimum q-route using reduced costs $\bar{c}_a^d$ can still be done in pseudo-polynomial time, basically by the same dynamic programming algorithm.

2.3 Extended Capacity Cuts

We introduce a family of cuts over the capacity-indexed variables. For each vertex $i \in V_+$ the following balance equation is valid:

$$\sum_{a^d \in \delta^-(i)} d x_a^d - \sum_{a^d \in \delta^+(i)} d x_a^d = d(i)\ . \tag{14}$$

Let $S \subseteq V_+$ be a set of vertices. Summing the equalities (14) corresponding to each $i \in S$, we get the *capacity-balance equation over* S:

$$\sum_{a^d \in \delta^-(S)} dx_a^d - \sum_{a^d \in \delta^+(S)} dx_a^d = d(S) \ . \tag{15}$$

It can be noted that those equations are always satisfied by the solutions of (10) (translated to the x^d space by (9b)). Nevertheless, they can be viewed as the source of a rich family of cuts.

Definition 1. *An* Extended Capacity Cut *(ECC) over S is any inequality valid for $P(S)$, the polyhedron given by the convex hull of the 0-1 solutions of* (15)*.*

The traditional rounded capacity cuts (1d) could be derived only from the above definition: for a given S relax (15) to $\geq$, divide both sides by C and round coefficients up. Remember that $\delta^+(S)$ contains no arc with capacity C, so all such coefficients are rounded to zero. All coefficients corresponding to $\delta^-(S)$ are rounded to one. Therefore they are ECCs. A slightly more complex reasoning shows that:

Proposition 1. *The root cutset inequalities* (6) *are ECCs.*

Proof. For any $S \subseteq V_+$, the following inequality is clearly valid for $P(S)$:

$$\sum_{a^d \in \delta^-(S)} dx_a^d \geq d(S). \tag{16}$$

Define $d^* = d(S) - C(k(S) - 1) - 1$. If at least one variable x_a^d with $d \leq d^*$ is set to one, we still need $k(S)$ additional variables set to one to satisfy (16). Otherwise, if all variables set to one have $d > d^*$, we need at least $k(S)$ such variables to satisfy (16). Hence

$$\frac{k(S)+1}{k(S)} \sum_{a^d \in \delta^-(S) \,:\, d > d^*} x_a^d + \sum_{a^d \in \delta^-(S) \,:\, d \leq d^*} x_a^d \geq k(S) + 1 \tag{17}$$

is also valid for $P(S)$ and dominates (6). □

An even stronger inequality can be obtained by considering that some arcs leaving S with a sufficiently large demand index may receive a negative coefficient:

$$\begin{aligned} \frac{k(S)+1}{k(S)} \sum_{a^d \in \delta^-(S) \,:\, d > d^*} x_a^d + \sum_{a^d \in \delta^-(S) \,:\, d \leq d^*} x_a^d \\ - \frac{1}{k(S)} \sum_{a^d \in \delta^-(S) \,:\, d \geq d'} x_a^d \geq k(S) + 1, \end{aligned} \tag{18}$$

where d' is the smallest integer such that

$$d' \geq d^* + 1 - \min_{i \in S} \{d(i)\} \tag{19}$$

and

$$d' \geq C - d^*. \tag{20}$$

Observe that, in any feasible ACVRP solution, each arc $a_1^{d_1} \in \delta^-(S)$ is associated to the next arc $a_2^{d_2} \in \delta^+(S)$ in the same route. Then, if at least one pair of associated arcs has $d_1 - d_2 \leq d^*$, we still need $k(S)$ additional entering arcs to satisfy (15). If a variable x_a^d with $a \in \delta^+(S)$ and $d \geq d'$ is set to one, then (19) ensures that the corresponding arc enters S with demand greater than d^* and (20) ensures that the difference between the entering and the leaving arc demands is not greater than d^*. In this case, the coefficient $-1/k(S)$ reduces the total contribution of such pair of arcs to the left-hand side of (18) from $(k(S)+1)/k(S)$ to 1. This particular ECC (18) will be referred as a *strengthened rounded capacity cut*. However, many other kinds of ECCs can be derived.

The *Homogeneous Extended Capacity Cuts* (HECCs) are a subset of the ECCs where all entering variables with the same capacity have the same coefficients, the same happening with the leaving variables. For a given set S, define aggregated variables y^d and z^d as follows:

$$y^d = \sum_{a^d \in \delta_d^-(S)} x_a^d \qquad (d = 1, \ldots, C), \tag{21}$$

$$z^d = \sum_{a^d \in \delta_d^+(S)} x_a^d \qquad (d = 0, \ldots, C). \tag{22}$$

The Capacity-Balance equation over those variables is:

$$\sum_{d=1}^{C} dy^d - \sum_{d=0}^{C} dz^d = d(S)\ . \tag{23}$$

For each possible pair of values of C and $D = d(S)$, we may define the polyhedron $P(C, D)$ induced by the integral solutions of (23). The inequalities that are valid for those polyhedra are HECCs. In Subsection 3.2 we illustrate how valid cuts can be derived and separated from that equality.

2.4 Triangle Clique Cuts

Let $S \subseteq V_+$ be a set of exactly three vertices. The triangle clique cuts are a family of cuts defined over the variables x_a^d, with $a^d = (i,j)^d \in A_C$ and $i, j \in S$. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the compatibility graph where each vertex of $\mathcal{V}$ is a capacity-indexed arc $a^d = (i,j)^d \in A_C$ with $i, j \in S$. In this case, an edge $e = (a_1^{d_1}, a_2^{d_2})$ belongs to $\mathcal{E}$ if and only if $a_1^{d_1}$ and $a_2^{d_2}$ are *compatible*. There are four cases:

Case 1: if $e = ((i,j)^{d_1}, (i,k)^{d_2})$, then $e \notin \mathcal{E}$;
Case 2: if $e = ((i,j)^{d_1}, (k,j)^{d_2})$, then $e \notin \mathcal{E}$;
Case 3: if $e = ((i,j)^{d_1}, (j,k)^{d_2})$ and $d_1 \neq d_2 + d(j)$, then $e \notin \mathcal{E}$;

Case 4: if $e = ((i,j)^{d_1}, (j,k)^{d_2})$ and $d_1 = d_2 + d(j)$, then $e \in \mathcal{E}$;

For any independent set $I \subset \mathcal{V}$, it is clear that the following inequality is valid

$$\sum_{a^d \in I} x_a^d \leq 1. \tag{24}$$

This is a well-known clique cut. However $\mathcal{G}$ has a nice structure that can be explored to build a very efficient separation procedure, as will be shown in Subsection 3.2.

2.5 Cuts over the arc variables *versus* cuts over the extended variables

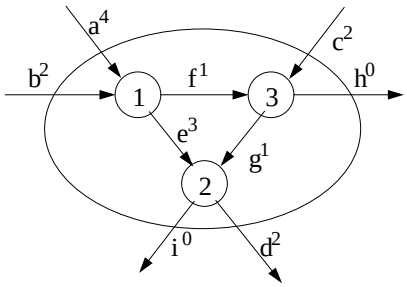

Fig. 1. Part of a fractional solution over the extended variables.

We now present an example to illustrate why it can be much easier to find a violated cut over the capacity-indexed extended variables than over the traditional arc variables. Figure 1 displays part of a fractional x_a^d solution of a unitary demand instance with $C = 4$, over a set $S = \{1, 2, 3\}$. The set S is being covered by 3 different q-routes with no cycles, each one with associated λ variable equal to 1/2. The first q-route enters S at vertex 1 with index 4 (arc a) and leaves the set at vertex 2 (arc d) with index 2. The second q-route enters at vertex 1 (arc b) with index 2 and leaves the set directly to the depot (arc h). The third q-route enters at vertex 3 (arc c) with index 2 and also leaves the set to the depot (arc i). The capacity-balance equality over the non-zero variables entering and leaving S is:

$$4x_a^4 + 2x_b^2 + 2x_c^2 - 2x_d^2 = 3.$$

As this equation has no 0-1 solution, there must be some violated ECC over S. For example, relaxing equation (23) to $\geq$, multiplying by 1/2 and performing integer rounding, a violated HECC is found:

$$2y^4 + 2y^3 + y^2 + y^1 - z^3 - z^2 \geq 2.$$

The same solution can also be cut by a triangle clique inequality:

$$x^3_e + x^1_f + x^1_g \leq 1.$$

In fact, taking a maximal independent set in $\mathcal{G}$ we can get a stronger lifted triangle clique:

$$x^3_e + x^3_{(2,3)} + x^3_{(3,1)} + x^3_f + x^3_g + x^3_{(2,1)} + x^1_e + x^1_{(2,3)} + x^1_{(3,1)} + x^1_f + x^1_g + x^1_{(2,1)} \leq 1.$$

On the other hand, it is impossible to cut this fractional solution in the x_a space by only looking at the variables entering and leaving S, and even by also looking at those inside S. This is true because the incidence vector formed by $x_a = x_b = x_c = x_d = x_e = x_f = x_g = x_h = x_i = 1/2$ and all the remaining variables in $(\delta^-(S) \cup \delta^+(S) \cup A(S))$ equal to 0 is a convex combination of two valid ACVRP solutions: the first with one route covering S using arcs $\{b, f, g, i\}$; the second solution covering S with two routes, using arcs $\{a, e, d\}$ and $\{c, h\}$. Of course, there must be some violated cut over the x_a space. But such a cut must involve other variables and is likely to be much more complex to identify and separate.

3 A Robust Branch-Cut-and-Price Algorithm

3.1 Column generation

Recall that the reduced cost of a λ variable in DWM (10) is the sum of the reduced costs $\bar{c}^d_a$ of the arcs in the corresponding q-route. Those reduced costs are calculated using equations (13). The pricing subproblem of finding the q-routes yielding a variable with minimum reduced cost is NP-hard (it contains the capacitated shortest path problem), but can be solved in pseudo-polynomial time. The basic data structure is a $C \times n$ matrix R. Each entry $R(d, v)$ represents the least costly walk that starts at vertex v with total demand exactly d and ends at the depot. The entry contains a *label* consisting of the vertex (v), the cost of the walk (denoted by $\bar{c}(R(d, v))$), and a pointer to a label representing the walk up to the next vertex. Initially, the only known label represents an empty path and has cost zero (it can be seen as entry $R(0, 0)$); all other entries are initialized with labels representing empty walks with infinite cost. The dynamic programming recursion given by

$$\bar{c}(R(d, v)) = \min_{w \in \delta^+(\{v\})} \{\bar{c}(R(d - d(v), w)) + \bar{c}^d_{vw}\}$$

is used to fill the entries of matrix R. 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, one coming from each vertex) 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 $O(n^2C)$.

Cycle elimination.

To strengthen the formulation, we could look for q-routes without cycles. Since this problem is strongly NP-hard, we settle 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. Labels retain the exact same meaning as before, but now each entry in the matrix contains no longer a single label, but a *bucket* of labels. Therefore, bucket $R(d, v)$ 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 extensions from v with exactly s vertices are considered. Only non-dominated labels are kept. A label ℓ is *s-dominated* by a set of labels $\mathcal{L}$ if no label in $\mathcal{L}$ costs more than ℓ and if every path p of length s that can legally extend ℓ (i.e., without creating an s-cycle) can also extend some label of $\mathcal{L}$.

Houck et al. [22] and Christofides et al. [11] already noted only two labels must be kept for the case $s = 2$. Given any three labels in the same bucket, the one with highest cost will be dominated by one of the others. So in this case the overall complexity is still $O(n^2C)$. For larger values of s, deciding which labels to keep becomes significantly more complicated, see [23, 17] for details. It is worth mentioning that buckets must have size at least $s!$, yielding a complexity of $O(s!s^2n^2C)$. The value $s = 3$ usually gives a good balance between formulation strength and pricing time.

3.2 Separation routines

Let $\bar{\lambda}$ be a fractional solution of the DWM LP. This solution can be converted into a $\bar{x}$ solution over the capacity-indexed arc space using equations (9b). Violated cuts of form (11) can be separated and added to the DWM as (12). In this subsection, we describe separation procedures for the families of cuts discussed previously.

CVRP Cuts

As mentioned before, all cuts known to be valid for the CVRP can be separated in this ACVRP algorithm. However, there is no point in introducing CVRP rounded capacity cuts, since they are dominated by inequalities (18). Among the many other families of CVRP cuts used in [32] and [17], we only separated strengthened combs. They improve bounds modestly. The other families did not produced any significant improvement.

Extended Capacity Cuts

Our procedure starts by choosing candidate sets S. Those candidates include:

- All sets S up to cardinality 6 which are connected in the support graph of the fractional solution $\bar{x}$, i.e., the subgraph of G containing only the arcs a

where some value $\bar{x}_a^d$ is positive. This connectivity restriction prevents an explosion on the number of enumerated sets. As proved in Uchoa et al. [36], if an ECC is violated over a set S composed of two or more disconnected components, there exists another violated ECC over one of those smaller components.

- The sets with cardinality larger than 6 that are inspected in the heuristic separation routines of rounded capacity cuts presented in [32]. The rationale is that if the rounded capacity cut is almost violated for a given set S, it is plausible that an extended capacity can be violated over that set. In particular, if the rounded capacity cut is violated, the ECC (18) will be certainly violated.

So, for each candidate set S, we first check if the strengthened rounded capacity cut (18) is violated. Then we try to separate HECCs from the equation (23) over S. In particular, we look for inequalities of the following form:

$$\sum_{d=1}^{C} \lceil rd \rceil y^d - \sum_{d=1}^{C-1} \lfloor rd \rfloor z^d \geq \lceil rd(S) \rceil \ , \tag{25}$$

where $0 < r \leq 1$. As discussed in [36], at most $0.3C^2$ rational multipliers r need to be tried in this integer rounding procedure.

Triangle Clique Cuts

The separation procedure for the triangle clique cuts finds the independent set $I \subset \mathcal{V}$ in $\mathcal{G}$ that maximizes $\sum_{a^d \in I} \bar{x}_a^d$. Although the problem of finding a maximum-weight independent set is strongly NP-hard for general graphs, such an independent set can be found for $\mathcal{G}$ in a linear time by exploiting its specific structure of chains.

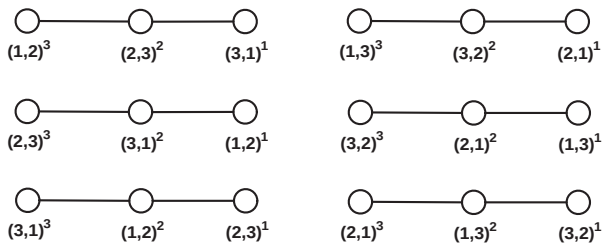

Fig. 2. The compatibility graph $\mathcal{G}$ for the example of Figure 1.

Figure 2 shows that the compatibility graph $\mathcal{G}$ for the example of Figure 1 is a set of 6 chains. For example, the arc $(1,2)^3$ is only compatible with $(2,3)^2$ in the triangle $\{1,2,3\}$. The arc $(2,3)^2$ is compatible with both $(1,2)^3$ and

$(3,1)^1$, and $(3,1)^1$ is only compatible with $(2,3)^2$. Observe that $(1,2)^3$ and $(3,1)^1$ are not compatible. Our separation procedure uses the following two facts.

Fact 1 *The graph $\mathcal{G}$ is a set of chains for any ACVRP instance.*

Fact 2 *A set I is a maximum-weight independent set for a set of chains if and only if it is the union of maximum-weight independent sets for each single chain.*

The maximum weight independent set for a single chain H can be obtained in a linear time through a simple dynamic programming procedure. Let $a_i^{d_i}$ be the ith vertex in H, for $i = 1, \ldots, |H|$. Also, let us define $I^*(i,1)$ (resp. $I^*(i,0)$) as the maximum independent set for the subchain containing the first i vertices of H that does (resp. does not) use the ith vertex. Finally, let $c(I) = \sum_{a^d \in I} \bar{x}_a^d$. We have the following recurrency:

$$c(I^*(i,1)) = \bar{x}_{a_i}^{d_i} + c(I^*(i-1,0));$$
$$c(I^*(i,0)) = \max\{c(I^*(i-1,0)), c(I^*(i-1,0))\}.$$

It is interesting to add suitable positive perturbations to the values of $\bar{x}_a^d$ that are zero, in order to generate inequalities with as many non-zero coefficients as possible.

3.3 Branching with route enumeration

We branch over the edges of the undirected graph associated to G. We choose the pair $\{i,j\}$ such that the value $\bar{x}_{\{i,j\}} = \sum_{d=0}^{C}(\bar{x}_{(i,j)}^d + \bar{x}_{(j,i)}^d)$ is closer to 0.65. On the left branch node we require that $\bar{x}_{\{i,j\}}$ must be 0, on the right branch node this must be greater or equal to 1. It can be shown that this is a valid branching strategy.

However, in order to improve the performance of our algorithm, we combine this traditional branching with a route enumeration technique inspired by the one described in Baldacci et al. [5]. When the integrality gap, the difference between the best known feasible solution and the current LP relaxation is sufficiently small, those authors found that it may be practical to enumerate all possible relevant elementary q-routes, i.e., all routes that have a chance of being part of the optimal solution. A route is non-relevant if (i) its reduced cost (with respect to the current values of (13)) is greater than the gap, or (ii) there exists another route visiting the same set of clients with smaller cost (with respect to the original arc costs c_a). If the number of relevant routes is not too large (say, in the range of tenths of thousands), the overall problem may be solved by feeding a general MIP solver with a set-partition formulation containing only those routes. If this set-partition can be solved, the optimal solution will be found and no branch will be necessary. Sometimes this leads to very significant speedups when compared to traditional branch strategies.

However, it should be remarked that such route enumeration is an inherently exponential procedure. Its practical performance depends crucially on the gap value and it is also sensitive to the characteristics of the instance that is being solved. There is no guarantee that a combinatorial explosion will not happen, even for small sized instances.

Our hybrid strategy, devised to provide a robust approach, is to perform limited route enumerations after each branch-and-bound node is solved. This means that the enumeration is aborted if more than 80,000 relevant routes or if more than 800,000 states (partial non-dominated routes) are being kept by our dynamic programming algorithm. If those limits are not reached, a set-partition containing all relevant routes is given to a MIP solver. Then, the original node is declared as *solved* and no branch will occur. Otherwise, if the route enumeration fails, then the edge branching is performed and two more nodes must be solved. Of course, since deeper nodes will have smaller gaps, at some point the enumeration will work. The overall effect may be a substantially smaller branch-and-bound tree. For example, where the traditional branching would need to reach depth 10, the hybrid strategy may not go beyond depth 5.

Our BCP also uses the route enumeration as an heuristic at the root node. If the actual gap g of this node is still too large and the limits are reached, we try the enumeration with a dummy gap of $g/2$. If this is still not enough, we try with $g/4$ and so on. If the enumeration now succeeds, we try an increased dummy gap of $(g/2 + g/4)/2$. In short, we perform a sort of binary search to determine a dummy gap that will yield a set-partition of reasonable size. The solution of such MIPs may provide very good upper bounds.

Finally, we should remark that the route enumeration is a quite sophisticated dynamic programming procedure, several tricks are necessary to prevent an early explosion on the number of states.

4 Adapting this RBCP for Related Routing Problems

This section shows how some related problems can be solved by slightly modifying the solution approach proposed above to the ACVRP.

4.1 Capacitated Open Vehicle Routing Problem

The Capacitated Open Vehicle Routing Problem (COVRP) [29] is a variant of the classical CVRP where the vehicle needs not to return to the depot once no more clients need to be visited (or, symmetrically, the vehicle needs not to start at the depot). This routing problem covers the case where vehicles are hired for each route job, having a cost model that charges only while the vehicle is loaded.

The ACVRP approach above described can be easily converted to solve the COVRP. For this we only need not to charge for the return of the vehicle.

This can be done by setting to zero the cost of all arcs that have as endpoint the depot.

4.2 Capacitated Vehicle Routing Problem

Solving the classical CVRP by a ACVRP approach should require no change other than the input having a symmetric cost matrix. However, this would lead to allowing two representations for a same route, i.e., the two possible orientations of its edges. The solution cost is indifferent regarding the orientation of a route (or q-route) and, as a consequence, a convergence difficulty may appear. Cuts that are asymmetric regarding the arcs that enter (or exit) a subset of vertices may become not violated by simply changing the orientation of one or a few routes. In particular, this is the case for all cuts derived from the extended formulation above.

We can deal with this difficulty by forbidding the generation of equivalent routes, i.e., the same route in its two senses. This can be achieved by requiring that every route with more than a single client has its last visited client with a larger index than its first client. The modification to the column generation dynamic programming algorithm amounts to adding n cells to each of nC cells of matrix R, which will now require $O(n^2C)$ positions. These extra cells will be used to store the last client visited (our algorithm constructs the q-route backwards). Remark that at most n routes will have to be stored. With this information at hand, one can easily check this anti-symmetry condition.

Although the resulting algorithm has a worst case complexity of $O(n^3C)$, several specific data structures can be used to improve the average case performance. This is also the case for the standard dynamic programming algorithm. They run much faster than their worst case and this symmetry breaking strategy seems to introduce a small factor on the computing time.

4.3 Heterogeneous Fleet Vehicle Routing Problem

We consider the Heterogeneous Fleet Vehicle Routing Problem (HFVRP) as the generalization of the classical CVRP where there is a set of vehicles types, with different capacities and fixed costs. This version of the HFVRP is considered in Yaman [39] and in Choi and Tcha [9].

Three minor modifications are needed to apply the ACVRP approach to the HFVRP. In all cuts that depend on C, one must use always the maximum capacity available among the vehicles. The second modification occurs in the pricing. Now the column generation algorithm executes for the largest capacity. The reduced cost of q-routes by other vehicles is then obtained by adding the fixed costs to the appropriate intermediate value in the matrix R. As in the CVRP, the symmetry breaking strategy should be used. The third modification is in the route enumeration, that must be performed for each vehicle type.

5 Experiments

We tested the resulting algorithm for the ACVRP, COVRP, CVRP and HFVRP on selected sets of instances from the literature. Our experiments were executed on a Pentium IV running at 3.0 GHz with 1 GB of RAM. Linear programs and set-partition IPs were solved by CPLEX 10.0.

5.1 ACVRP

We used the 8 ACVRP instances proposed by Fischetti et al. [16]. As all those instances have very large capacities ($C = 1000$) and use very few vehicles (2 or 3), in order to have a more representative benchmark set, we also solved the same instances by considering capacities 500, 250 and 150. Of course, this implies using more vehicles.

We first remark that the CVRP algorithm presented in Fukasawa et al. [17] can be used as an ACVRP algorithm. One only needs to use the asymmetric arc costs in the dynamic programming pricing. Of course, only CVRP cuts will be employed. Table 1 allows the evaluation of the impact of introducing the cuts from the extended formulation in the new BCP algorithm. The first column contains the name of the instance while the second (k) and third (C) specify the number and the capacity of the vehicles used, respectively. The fourth and fifth columns (**Fuk.**), give the lower bound obtained in the root node of the BCP algorithm in [17] and the CPU time (in seconds) spent. The following two columns give this same information for the new BCP showed in the previous sections, that also uses asymmetric cuts from the extended formulation (**New**). Next column (**Prev UB**) gives the optimal solution values for the original 8 instances with capacity 1000, obtained by [16]. For the remaining 24 instances introduced here, this column gives the best upper bound found by running the first BCP algorithm with a time limit of 10,000 seconds. Proven optima values appear in boldface. The last column is the best upper bound found by the new BCP algorithm with a time limit of 10,000 seconds, proven optima are also in boldface. Blank entries indicate that no feasible solution was found at within that time limit. We remark that both algorithms price q-routes with 3-cycle elimination. It can be seen that the new cuts can indeed reduce gaps substantially, but this also has a substantial price in terms of computational time. The new BCP is efficient on the instances with smaller capacities and more vehicles, but performs poorly on the instances with large capacities and very few vehicles, due to cut and column convergence problems. It could not even finish the root node of instance a071-03f within the time limit. This behavior is consistent with previous experiences [17]. The instances with only 3 vehicles are much better solved by the branch-and-bound by Fischetti et al., the solution times reported in [16] would correspond to less than one second in a modern machine. We expect that a branch-and-cut over the arc formulation, similar to the one by Lysgaard et al [32], would also be efficient in those cases.

Table 1. Bounds on the ACVRP instances.

Instance	k	C	Fuk. LB	Fuk. Time	New LB	New Time	Prev UB	New UB
a034-14f	14	150	4046.00	0.8	4046.00	0.2	**4046**	**4046**
a036-18f	18	150	5296.00	0.2	5296.00	0.4	**5296**	**5296**
a039-20f	20	150	5903.00	0.2	5903.00	0.3	**5903**	**5903**
a045-18f	18	150	6365.00	1.2	6374.50	4.7	**6399**	**6399**
a048-16f	16	150	4905.61	1.9	4910.82	7.6	**4955**	**4955**
a056-17f	17	150	4974.21	2.9	4976.69	25.2	**4998**	**4998**
a065-19f	19	150	5972.55	3.9	5986.58	37.0	**6014**	**6014**
a071-17f	17	150	4937.41	6.5	4949.08	89.8	**5006**	**5006**
a034-08f	8	250	2643.47	1.8	2654.33	23.9	**2672**	**2672**
a036-10f	10	250	3306.09	1.9	3313.51	29.2	**3338**	**3338**
a039-12f	12	250	3705.00	0.8	3705.00	5.2	**3705**	**3705**
a045-11f	11	250	3542.17	1.9	3544.00	12.1	**3544**	**3544**
a048-10f	10	250	3298.27	2.9	3306.28	59.3	**3325**	**3325**
a056-10f	10	250	3258.57	5.8	3262.08	89.3	**3263**	**3263**
a065-12f	12	250	3848.72	8.4	3856.14	150.7	**3902**	**3902**
a071-10f	10	250	3415.03	25.1	3423.69	231.5	3486	3486
a034-04f	4	500	1759.41	4.4	1766.52	97.3	**1773**	**1773**
a036-05f	5	500	2084.27	4.9	2088.17	67.6	**2110**	**2110**
a039-06f	6	500	2270.60	5.5	2277.15	141.8	**2289**	**2289**
a045-06f	6	500	2289.81	8.5	2294.55	195.8	**2303**	**2303**
a048-05f	5	500	2260.29	11.7	2265.36	213.0	**2283**	**2283**
a056-05f	5	500	2144.09	30.9	2152.87	695.9	**2165**	**2165**
a065-06f	6	500	2516.90	29.8	2521.69	694.9	**2567**	2567
a071-05f	5	500	2403.12	94.0	2411.79	1443.3	2475	–
a034-02f	2	1000	1381.12	25.5	1392.41	1388.2	**1406**	**1406**
a036-03f	3	1000	1635.27	22.9	1638.46	1009.4	**1644**	**1644**
a039-03f	3	1000	1654.00	15.7	1654.00	61.9	**1654**	**1654**
a045-03f	3	1000	1740.00	39.2	1740.00	325.5	**1740**	**1740**
a048-03f	3	1000	1891.00	106.3	1891.00	825.6	**1891**	**1891**
a056-03f	3	1000	1725.28	256.5	1727.60	5441.1	**1739**	–
a065-03f	3	1000	1956.75	320.3	1969.72	11688.4	**1974**	–
a071-03f	3	1000	2037.67	1465.7	–	–	**2054**	–
avg gap			0.79%		0.59%			

Detailed statistics on the new algorithm are presented in Tables 2 and 3. The 2nd, 3rd and 4th columns contain the number of cuts of each type inserted in the root node. The headers **SRCC**, **Rd ECC** and **Clique** mean strengthened rounded capacity cuts, HECCs obtained by integer rounding and triangle clique cuts, respectively. The following four columns contain the time spent in the root node with column generation, LP solving, cut separation, and route enumeration + set-partition solving, respectively. The remaining columns give information about the complete algorithm: **# BCP Nodes** is the total number of BCP nodes solved, **# SP Nodes** is the total number of nodes used by CPLEX to solve all set-partition problems, and **Total Time** is the overall time.

Some instances were not solved to optimality, in those cases we give an estimative of how far the algorithm was from finishing. The BCP performs depth-first search. Every time a leaf node at level l is solved (root level is 0), we consider that $100/2^l$ % of the search tree was solved. Those values are accumulated. Since our branch rule was devised to yield an statistically balanced tree, this provides a reasonably good estimate of algorithm progress, at least after several leaf nodes are solved. For example, instance a071-10f was halted with 10,000 seconds. We report that the 67 nodes already solved at that time correspond to 26.6% of the estimated tree size.

Table 2. Statistics on the ACVRP instances.

Instance	# Root Cuts			Root Times				# BCP	# SP	Total
	SRCC	Rd ECC	Clique	Col Gen	LP	Cut Sep	Enum +SP	Nodes	Nodes	Time
a034-14f	0	0	0	0.1	0.1	0.0	0.1	1	1	0.4
a036-18f	2	0	0	0.2	0.1	0.0	0.1	1	1	0.5
a039-20f	0	0	0	0.2	0.0	0.0	0.1	1	1	0.4
a045-18f	13	14	1	2.0	0.2	1.7	0.3	1	1	5.0
a048-16f	40	20	10	2.6	0.3	3.5	1.4	1	175	9.0
a056-17f	50	40	4	8.0	0.4	10.1	1.1	1	1	26.3
a065-19f	39	75	10	12.1	0.5	18.3	1.3	1	1	38.4
a071-17f	113	99	32	36.8	1.1	22.0	43.4	1	596	133.2
a034-08f	56	50	13	12.5	0.7	5.4	0.8	1	1	24.7
a036-10f	77	54	9	11.4	0.6	7.9	0.9	1	14	30.0
a039-12f	58	0	0	2.3	0.3	1.8	0.6	1	1	5.8
a045-11f	18	21	4	8.0	0.4	1.5	1.1	1	1	13.2
a048-10f	78	59	14	34.7	1.1	11.0	2.9	1	1	62.1
a056-10f	53	85	18	51.2	1.5	19.3	2.4	1	1	91.7
a065-12f	70	73	23	76.8	1.7	35.9	263.2	3	118	663.8
a071-10f	147	119	39	141.2	3.7	41.8	641.5	(26.6%) 67	31	10000.0

Table 3. Statistics on the ACVRP instances.

Instance	# Root Cuts			Root Times				# BCP	# SP	Total
	SRCC	Rd ECC	Clique	Col Gen	LP	Cut Sep	Enum +SP	Nodes	Nodes	Time
a034-04f	48	55	21	57.8	1.6	19.2	4.0	1	1	101.3
a036-05f	46	26	15	39.2	1.0	15.2	5.0	1	1	72.6
a039-06f	46	104	10	71.0	1.4	40.0	4.5	1	1	146.4
a045-06f	117	65	21	109.6	2.0	47.1	6.8	1	1	202.6
a048-05f	100	63	28	150.3	3.0	34.9	25.8	1	1	238.8
a056-05f	166	74	30	493.2	7.6	101.9	65.6	1	1	761.5
a065-06f	129	99	28	427.8	8.6	130.0	985.9	(2.3%) 24	9	10000.0
a071-05f	120	135	39	1119.1	31.2	147.0	1311.0	(0.1%) 12	1	10000.0
a034-02f	41	338	18	864.8	8.5	292.3	45.6	1	1	1433.8
a036-03f	91	201	10	555.3	5.9	245.1	22.7	1	1	1032.0
a039-03f	0	0	0	59.8	1.2	0.8	21.0	1	1	82.9
a045-03f	84	0	0	292.5	12.4	14.4	29.4	1	1	354.9
a048-03f	90	0	0	778.8	20.7	17.8	35.1	1	1	860.7
a056-03f	138	411	14	4403.0	118.2	517.9	699.8	(0.0%) 3	0	10000.0
a065-03f	109	312	51	7140.4	80.0	2367.4	1120.6	(0.0%) 1	0	10000.0
a071-03f	-	-	-	-	-	-	-	(0.0%) 0	0	10000.0

5.2 COVRP

We chose a set of 15 instances from the literature. They include most of the classical E instances from Christofides and Eilon [10] and also some representative instances from sets A, B and P available at `www.branchandcut.org`. We compare our bounds with those obtained by the branch-and-cut by Letchford et al. [29], **Let.** columns. Their times were obtained with a Pentium M 1.6 GHz processor. All such instances are Euclidean, the costs are obtained from the depot and client coordinates, without rounding. The results are presented in Tables 4 and 5, analogous to Tables 1 and 2, respectively. The upper bounds in column **Prev UB** in Table 4 are from [29] too. As instances E-n101-k8 and E-n101-k14 had never been solved before, we allowed a little more than 10,000 seconds for their runs.

Table 4. Bounds on the COVRP instances.

Instance	k	C	Let. LB	Let. Time(s)	New LB	New Time(s)	Prev UB	New UB
A-n63-k10	10	100	745.94	5	775.26	63.2	–	**778.46**
A-n64-k9	9	100	811.98	4	839.25	208.6	–	**848.15**
A-n69-k9	9	100	732.27	5	754.53	71.4	**757.76**	**757.76**
A-n80-k10	10	100	1019.84	9	1061.90	126.1	–	**1067.09**
B-n50-k8	8	100	703.57	4	717.05	49.5	–	**720.79**
B-n68-k9	9	100	694.16	6	701.71	187.4	**701.71**	**701.71**
E-n51-k5	5	160	411.48	1	414.32	88.7	**416.06**	**416.06**
E-n76-k7	7	220	522.27	10	525.21	245.7	**530.02**	**530.02**
E-n76-k8	8	180	529.37	12	534.16	127.2	**537.24**	**537.24**
E-n76-k10	10	140	547.83	21	559.62	97.0	–	**567.14**
E-n76-k14	14	100	602.01	24	621.27	70.0	–	**623.55**
E-n101-k8	8	200	633.85	6	636.09	423.4	**639.74**	**639.74**
E-n101-k14	14	112	692.15	33	704.69	227.8	–	**711.58**
P-n50-k8	8	120	405.85	2	422.99	26.9	–	**436.51**
P-n70-k10	10	135	536.04	9	547.51	80.4	–	**552.65**
avg gap			2.95%		0.80%			

Table 5. Statistics on the COVRP instances.

Instance	# Root Cuts			Root Times				# BCP	# SP	Total
	SRCC	Rd ECC	Clique	Col Gen	LP	Cut Sep	Enum +SP	Nodes	Nodes	Time
A-n63-k10	98	81	23	34.6	3.9	9.4	187.4	1	1	250.6
A-n64-k9	168	144	28	119.5	8.7	27.0	659.6	9	59	1884.1
A-n69-k9	78	148	14	39.4	6.7	10.1	3.8	1	1	75.2
A-n80-k10	77	113	41	76.7	8.9	18.3	458.8	1	5	585.0
B-n50-k8	171	33	14	31.5	4.6	7.3	176.5	1	101	226.0
B-n68-k9	143	138	8	120.2	42.7	12.5	1.3	1	1	188.7
E-n51-k5	45	137	20	45.7	6.5	15.0	5.0	1	1	93.7
E-n76-k7	23	202	33	133.3	12.8	38.6	319.9	17	9	4373.7
E-n76-k8	21	131	26	71.8	5.8	22.1	36.1	1	1	163.3
E-n76-k10	40	171	44	40.4	3.7	24.6	425.9	5	3	996.7
E-n76-k14	56	133	39	29.2	1.9	15.8	167.4	1	1	237.4
E-n101-k8	108	184	35	206.4	48.8	73.9	303.2	49	25	15941.9
E-n101-k14	117	259	82	90.3	10.0	53.5	1290.7	57	47	10335.3
P-n50-k8	63	81	32	9.8	1.2	7.5	933.6	1	1077	960.5
P-n70-k10	90	109	40	33.3	3.0	19.3	269.8	1	1	350.2

5.3 CVRP

We used the same 15 instances of the previous experiment. However, as usual in the CVRP literature, the costs are rounded Euclidean distances, following the TSPLIB convention. We compare the bounds and root times from the new BCP with those from the BCP presented in Fukasawa et al. [17]. We remark that while that algorithm was executed using 3-cycle elimination, our new algorithm, in this case, used 2-cycle elimination together with route symmetry breaking. Table 6 is analogous to Table 1, with **Fuk.** indicating the columns with results from [17] obtained in a Pentium IV 2.4 GHz machine. Table 7 shows that 5 instances could not be finished by the complete BCP in the allotted time of 10,000 seconds.

Table 6. Bounds on the CVRP instances.

Instance	k	C	Fuk. LB	Fuk. Time(s)	New LB	New Time(s)	Prev UB	New UB
A-n63-k10	10	100	1299.1	136	1299.82	114.9	**1314**	**1314**
A-n64-k9	9	100	1385.3	265	1385.21	141.4	**1401**	**1401**
A-n69-k9	9	100	1141.4	289	1147.38	151.0	**1159**	**1159**
A-n80-k10	10	100	1754.0	1120	1755.14	242.7	**1763**	**1763**
B-n50-k8	8	100	1295.0	97	1299.62	46.6	**1312**	1312
B-n68-k9	9	100	1263.0	260	1264.14	258.8	**1272**	1272
E-n51-k5	5	160	518.2	51	520.09	141.0	**521**	**521**
E-n76-k7	7	220	670.0	264	671.81	216.0	**682**	682
E-n76-k8	8	180	726.5	277	727.93	177.5	**735**	**735**
E-n76-k10	10	140	817.4	354	819.56	131.6	**830**	**830**
E-n76-k14	14	100	1006.5	224	1008.67	47.4	**1021**	**1021**
E-n101-k8	8	200	805.2	1068	806.01	705.5	**815**	–
E-n101-k14	14	112	1053.8	658	1056.27	214.7	**1067**	1067
P-n50-k8	8	120	616.3	102	618.96	28.6	**631**	**631**
P-n70-k10	10	135	814.5	292	816.88	131.3	**827**	**827**
avg gap			1.26%		1.04%			

Table 7. Statistics on the CVRP instances.

Instance	# Root Cuts			Root Times				# BCP	# SP	Total
	SRCC	Rd ECC	Clique	Col Gen	LP	Cut Sep	Enum +SP	Nodes	Nodes	Time
A-n63-k10	198	96	48	56.2	3.4	19.5	316.7	3	16	728.9
A-n64-k9	260	147	55	75.3	5.4	24.7	654.8	55	28	9161.0
A-n69-k9	230	137	65	78.7	5.3	29.7	433.8	5	3	1248.3
A-n80-k10	314	113	69	115.0	10.3	45.3	660.0	3	2	1530.9
B-n50-k8	214	46	19	26.8	2.2	8.0	309.5	(69%) 128	58	10000.0
B-n68-k9	285	149	29	140.4	11.0	31.5	526.5	(25.5%) 52	25	10000.0
E-n51-k5	147	110	46	51.3	4.4	28.8	4.8	1	1	145.8
E-n76-k7	110	93	59	120.7	5.0	43.3	1040.3	(11%) 32	14	10000.0
E-n76-k8	175	95	64	93.1	5.2	41.2	534.0	5	3	1840.1
E-n76-k10	193	78	68	60.8	2.8	35.8	471.5	39	20	5794.4
E-n76-k14	60	90	45	22.0	0.9	12.2	246.3	3	151	479.4
E-n101-k8	659	159	114	270.2	36.5	224.9	1462.3	(0.1%) 20	5	10000.0
E-n101-k14	237	122	95	95.6	3.7	63.1	3469.4	(4.7%) 31	13	10000.0
P-n50-k8	54	90	48	10.5	0.8	8.3	42.0	1	16	70.6
P-n70-k10	126	102	50	54.6	2.1	37.0	399.9	5	23	1112.1

5.4 HFVRP

We used a set of instances proposed in Golden et al. [20], the same set was used in the experiments in Yaman [39] and Choi and Tcha [9] which are compared with our approach. Bound comparisons are presented in table 8 where the columns with header t and C contain the number of different types of vehicles and the largest capacity, respectively. It worth noting that the bounds from Yaman [39], given in the columns **Yam.**, come from a cutting plane algorithm on a flow formulation. Her times were obtained in a Sun Ultra 400 MHz machine. The lower bounds from Choi and Tcha [9], columns **Choi**, come from a column generation algorithm on q-routes with 2-cycle elimination. Their times were obtained in a Pentium IV 2.6 GHz machine. The previous best upper bounds were also collected from [9]. Our algorithm for the HFVRP executes as for the CVRP, with 2-cycle elimination and route symmetry breaking. Table 9 give detailed statistics on the new algorithm. We allowed more than 10,000 seconds for solving instance c75-18. Those results represent a major improvement over previous exact algorithms for that problem. We could not find any work claiming proven optimal solutions even for the instances with just 20 vertices.

Table 8. Bounds on the HFVRP instances.

Instance	t	C	Yam. LB	Yam. Time(s)	Choi LB	Choi Time(s)	New LB	New Time(s)	Prev UB	New UB
c20-3	5	120	912.40	–	951.61	0.4	961.03	2.5	961.03	**961.03**
c20-4	3	150	6369.51	–	6369.15	0.7	6437.33	5.2	6437.33	**6437.34**
c20-5	5	120	959.29	–	988.01	0.8	1001.12	7.2	1007.05	**1007.05**
c20-6	3	150	6468.44	–	6451.62	0.4	6515.82	4.7	6516.47	**6516.47**
c50-13	6	200	2365.78	397.1	2392.77	10.1	2400.93	107.0	2406.36	**2406.36**
c50-14	3	300	8943.94	175.6	8748.57	50.8	9111.81	277.2	9119.03	**9119.03**
c50-15	3	160	2503.61	142.8	2544.84	10.0	2572.99	72.3	2586.37	**2586.37**
c50-16	3	140	2650.76	142.1	2685.92	11.3	2705.35	48.8	2720.43	**2720.43**
c75-17	4	350	1689.93	1344.8	1709.85	206.9	1717.37	456.9	1744.83	1734.52
c75-18	6	400	2276.31	1922.8	2342.84	70.1	2351.07	347.5	2371.49	**2369.64**
c100-19	3	300	8574.33	1721.2	8431.87	1178.9	8648.68	1017.3	8661.81	8661.81
c100-20	3	200	3931.79	2904.0	3995.16	264.0	4005.29	407.1	4039.49	–
avg gap			2.63%		1.60%		0.44%			

Table 9. Statistics on the HFVRP instances.

Instance	# Root Cuts SRCC	Rd ECC	Clique	Root Times Col Gen	LP	Cut Sep	Enum +SP	# BCP Nodes	# SP Nodes	Total Time
c20-3	13	50	15	1.4	0.4	0.3	0.2	1	1	2.7
c20-4	6	169	14	2.6	0.6	1.1	0.3	1	1	5.5
c20-5	34	81	19	3.5	0.8	1.2	1.0	1	1	8.2
c20-6	1	103	3	2.8	0.4	0.9	0.3	1	1	5.0
c50-13	37	250	38	49.1	3.5	20.8	14.3	1	1	121.3
c50-14	13	1148	51	96.5	27.5	82.9	28.6	1	5	305.8
c50-15	16	475	40	30.8	6.9	17.2	282.9	1	238	355.2
c50-16	21	254	51	23.3	2.4	10.6	211.4	1	91	260.2
c75-17	15	497	87	236.5	20.5	109.8	6271.9	(0.0%) 11	0	10000.0
c75-18	27	296	55	179.4	3.8	80.2	3326.4	79	475	41607.1
c100-19	41	830	109	598.0	45.2	202.5	1063.8	(0.2%) 19	5	10000.0
c100-20	21	265	87	236.3	6.4	84.3	6265.9	(0.0%) 15	0	10000.0

6 Comments

This text presented a RBCP for the ACVRP that was also shown to be effective on a number of related problems. In particular, the use of cuts defined over the extended formulation seems very promising and deservers further development. The ECCs here utilized, strengthened rounded capacity cuts and HECCs obtained by simple integer rounding can still be improved and better separated. Fukasawa, Dash and Gunluk [13] have just characterized the facets of the polyhedron $P(C, D)$ induced by the integral solutions of (23). This may immediately lead to the separation of the strongest possible HECCs for a given set S. Moreover, the current choice of candidate sets for separation is still naive and could be improved. Another line of research is the development of cuts from the arc-indexed compatibility graph over sets with cardinality larger than 3. Odd-hole cuts and even more complex families of cuts are already known to exist even for sets of cardinality 4 or 5.

The new extended cuts have shown to be particularly powerful on the HFVRP. While traditional arcs cuts are weakened by the existence of several capacities, the ECCs and triangle cliques are insensitive to that. In this case, any arc a^d is viewed as bringing d units of capacity, the total capacity of the vehicles associated to the q-routes that contributed to the value of x_a^d is irrelevant.

We believe that the development of robust BCP algorithms for the VRP, guaranteeing a pseudo-polynomial pricing complexity, is an important issue. Some non-robust BCPs are being developed based on the use of clever techniques to solve strongly NP-hard pricing subproblems. They proved to be quite successful on current typical instances, with up to 100 clients. However those approaches are much less likely to work on instances with several hundreds of clients (specially those more than a dozen clients per route) that will certainly arise in the near future.

Acknowledgments

The authors thank two anonymous referees for helpful comments. AP, MPA and EU were partially financed by CNPq grants 301175/2006-3, 311997/06-6 and 304533/02-5. AP and EU received support from Mestrado em Engenharia de Produção-UFF. The authors also thank the technical support by GAPSO Inc., Brazil.

References

1. N. Achuthan, L. Caccetta, and S. Hill. Capacited vehicle routing problem: Some new cutting planes. *Asia-Pacific J. of Operational Research*, 15:109–123, 1998.

2. J. Araque, L. Hall, and T. Magnanti. Capacitated trees, capacitated routing, and associated polyhedra. Technical report, MIT, Operations Research Center, 1990.
3. P. Augerat. *Approche polyèdrale du problème de tournées de véhicles.* PhD thesis, Institut National Polytechnique de Grenoble, 1995.
4. P. Augerat, J. Belenguer, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi. Computational results with a branch and cut code for the capacitated vehicle routing problem. Technical Report 949-M, Université Joseph Fourier, Grenoble, France, 1995.
5. R. Baldacci, L. Bodin, and A. Mingozzi. The multiple disposal facilities and multiple inventory locations rollon-rolloff vehicle routing problem. *Computers and Operation Research*, 33:2667–2702, 2006.
6. R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. Submitted, 2007.
7. C. Barnhart, C. Hane, and P. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 40:318–326, 2000.
8. C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, and P. Vance. Branch-and-price: column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1998.
9. E. Choi and D-W Tcha. A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers and Operations Research*, 34:2080–2095, 2007.
10. N. Christofides and S. Eilon. An algorithm for the vehicle-dispatching problem. *Operational Research Quarterly*, 20:309–318, 1969.
11. N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20:255–282, 1981.
12. G. Cornuéjols and F. Harche. Polyhedral study of the capacitated vehicle routing problem. *Mathematical Programming*, 60:21–52, 1993.
13. S. Dash, R. Fukasawa, and O. Gunluk. On the generalized master knapsack polyhedron. In *Proceedings of the IPCO 2007*, 2007.
14. du Merle, O. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194:229–237, 1999.
15. G. Felici, C. Gentile, and G. Rinaldi. Solving large MIP models in supply chain management by branch & cut. Technical report, Istituto di Analisi dei Sistemi ed Informatica del CNR, Italy, 2000.
16. M. Fischetti, P. Toth, and D. Vigo. A branch and bound algorithm for the capacitated vehicle routing problem on directed graphs. *Operations Research*, 42:846–859, 1994.
17. R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, and R. F. Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106:491–511, 2006.
18. R. Fukasawa, M. Reis, M. Poggi de Aragão, and E. Uchoa. 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.

19. M. T. Godinho, L. Gouveia, and T. Magnanti. Combined route capacity and path length models for unit demand vehicle routing problems. In *Proceedings of the INOC*, volume 1, pages 8–15, Lisbon, 2005.
20. B. Golden, A. Assad, L. Levy, and F. Gheysens. The fleet size and mix vehicle routing problem. *Computers and Operations Research*, 11:49–66, 1984.
21. L. Hall and T. Magnanti. A polyhedral intersection theorem for capacitated spanning trees. *Mathematics of Operations Research*, 17, 1992.
22. D. Houck, J. Picard, M. Queyranne, and R. Vegamundi. The travelling salesman problem as a constrained shortest path problem. *Opsearch*, 17:93–109, 1980.
23. S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, 18:391–406, 2006.
24. M. Jepsen, S. Spoorendonk, B. Petersen, and D. Pisinger. A non-robust branch-and-cut-and-price for the vehicle routing problem with time windows. Technical Report 06/03, University of Copenhagen, 2006.
25. B. Kallehauge, N. Boland, and O. Madsen. Path inequalities for the vehicle routing problem with time windows. Technical report, Technical University of Denmark, 2005.
26. D. Kim, C. Barnhart, K. Ware, and G. Reinhardt. Multimodal express package delivery: A service network design application. *Transportation Science*, 33:391–407, 1999.
27. N. Kohl, J. Desrosiers, O. Madsen, M. Solomon, and F. Soumis. 2-Path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33:101–116, 1999.
28. A. Letchford, R. Eglese, and J. Lysgaard. Multistars, partial multistars and the capacitated vehicle routing problem. *Mathematical Programming*, 94:21–40, 2002.
29. A. Letchford, J. Lysgaard, and R. Eglese. A branch-and-cut algorithm for the capacitated open vehicle routing problem. *Journal of the Operational Research Society*, 2007.
30. A. Letchford and J-J. Salazar. Projection results for vehicle routing. *Mathematical Programming*, 105:251–274, 2006.
31. J. Lysgaard. Reachability cuts for the vehicle routing problem with time windows. *European Journal of Operational Research*, 175:210–233, 2006.
32. J. Lysgaard, A. Letchford, and R. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100:423–445, 2004.
33. D. Naddef and G. Rinaldi. Branch-and-cut algorithms for the capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, chapter 3, pages 53–84. SIAM, 2002.
34. J. Picard and M. Queyranne. The time-dependant traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Operations Research*, 26:86–110, 1978.
35. M. Poggi de Aragão and E. Uchoa. Integer program reformulation for robust branch-and-cut-and-price. In L. Wolsey, editor, *Annals of Mathematical Programming in Rio*, pages 56–61, Búzios, Brazil, 2003.
36. E. Uchoa, R. Fukasawa, J. Lysgaard, A. Pessoa, M. Poggi de Aragão, and D. Andrade. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, on-line first, 2007.

37. J. Van den Akker, C. Hurkens, and M. Savelsbergh. Time-indexed formulation for machine scheduling problems: column generation. *INFORMS J. on Computing*, 12:111–124, 2000.
38. F. Vanderbeck. Lot-sizing with start-up times. *Management Science*, 44:1409–1425, 1998.
39. H. Yaman. Formulations and valid inequalities for the heterogeneous vehicle routing problem. *Mathematical Programming*, 106:365–390, 2006.