Chapter 3

New Exact Algorithms for the Capacitated Vehicle Routing Problem

Marcus Poggi
Eduardo Uchoa

3.1 ▪ Introduction

Since the seminal work by Desrosiers, Soumis, and Desrochers [15], column generation has been the dominant approach for building exact algorithms for the *Vehicle Routing Problem with Time Windows* (VRPTW). This technique performed very well on tightly constrained instances (those with narrow time windows). As the *Capacitated Vehicle Routing Problem* (CVRP) can be regarded as the particular case of VRPTW where time windows are arbitrarily large, column generation was viewed as a non-promising approach for the problem. In fact, in the early 2000's, the best performing algorithms for the CVRP were Branch-and-Cut algorithms that separated quite complex families of cuts identified by polyhedral investigation (see Naddef and Rinaldi [31] and Chapter 2). In spite of their sophistication, some instances from the literature with only 50 customers could not be solved to optimality. At that moment, the *Branch-and-Cut-and-Price algorithm* (BCP) by Fukasawa et al. [19] showed that the combination of cut and column generation could be much more effective than each of those techniques taken alone. Since then, the most performing exact algorithms proposed for the CVRP are based on that combination.

According to the classification proposed in Poggi de Aragão and Uchoa [36], the BCP algorithm in Fukasawa et al. [19] only uses *robust cuts*. A cut is said to be robust when the value of the dual variable associated with it can be translated into costs in the pricing subproblem. Therefore, the structure and the size of that subproblem remain unaltered, regardless of the number of robust cuts added. On the other hand, *non-robust cuts* are those that change the structure and/or the size of the pricing subproblem; each additional cut makes it harder.

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. With the addition of non-robust cuts, there is a risk of having to solve to optimality an intractable subproblem.

Nevertheless, since Jepsen et al. [25] and Baldacci, Christofides, and Mingozzi [3] it has been known that some non-robust cuts can be effectively used, at least if they are separated in a controlled way, avoiding an excessive impact on the pricing. While the adjective *non-robust* focuses on their negative aspect, some authors mentioned in this chapter call them *strong cuts*, focusing on their positive aspect, a greater potential for significantly reducing the integrality gaps. In fact, some of the best algorithms available today rely heavily on them.

3.2 ▪ Main Exact Approaches

We briefly review the most recent works proposing exact algorithms for the CVRP; all of them are based on the combination of column and cut generation. The main elements of those algorithms will be described in more depth (and even some terms will only be properly defined) in the remaining sections of this chapter.

1. **Fukasawa et al. [19]** presented a BCP algorithm having the following features:

   - The columns are associated with the q-routes without k-cycles, a relaxation of the elementary routes that allow multiple visits to a customer, on the condition that at least k other costumers are visited between successive visits. The separated cuts are the same used in previous Branch-and-Cut algorithms over the two-index CVRP formulation. Those cuts are robust with respect to q-route pricing.
   - If the column generation at the root node is found to be too slow, the algorithm automatically switches to a Branch-and-Cut.

   All benchmark instances from the literature with up to 135 vertices could be solved to optimality, a significant improvement over previous methods.

2. **Baldacci, Christofides, and Mingozzi [3]** presented an algorithm based on column and cut generation with the following features:

   - The columns are associated with elementary routes. Besides cuts for the two-index formulation, Strengthened Capacity Cuts and Clique Cuts are separated. Those latter cuts are effective but non-robust; they make the pricing harder.
   - A sequence of cheaper lower bounding procedures produces good estimates of the optimal dual variable values. The potentially expensive pricing is only called in the last stage, with the dual variables bounded to be above the estimates, obtaining convergence (to a bound slightly below the theoretical optimal) in fewer iterations.
   - Instead of branching, the algorithm finishes in the root node (therefore, it is not a BCP) by enumerating all elementary routes with reduced cost smaller than the duality gap. A set-partitioning problem containing all those routes is then given to a *Mixed-Integer Programming* (MIP) solver.

The algorithm could solve almost all instances solved in [19], usually taking much less time. However, the exponential nature of some algorithmic elements, in particular the route enumeration, made it fail on some instances with many customers per vehicle.

3. **Pessoa, Poggi de Aragão, and Uchoa [33]** presented some improvements over Fukasawa et al. [19]:

   - Cuts from an extended formulation with capacity indices were also separated. Those cuts do not change the complexity of the pricing of q-routes by dynamic programming.
   - The idea of performing elementary route enumeration and MIP solving to finish a node was borrowed from [3]. However, in order to avoid a premature failure when the root gap is too large, it was hybridized with traditional branching.

4. **Baldacci, Mingozzi, and Roberti [4]** improves upon Baldacci, Christofides, and Mingozzi [3] by the introduction of the following elements:

   - The ng-routes, a newly proposed relaxation that is more effective than the q-routes without k-cycles, are used in the earlier bounding procedures and also to accelerate the pricing/enumeration of elementary routes. This latter part of the algorithm is also enhanced by considering multiple dual solutions. A route having reduced cost larger, with respect to any dual solution, than the current duality gap can be discarded.
   - Subset Row Cuts and Weak Subset Row Cuts, which have less impact on the pricing with respect to Clique Cuts, are separated.

   The resulting algorithm is not only faster on average, but it is much more stable than the algorithm in [3], being able to solve even some instances with many customers per vehicle.

5. **Contardo [12]** introduced new twists on the use of non-robust cuts and on route enumeration:

   - The columns are associated with q-routes without 2-cycles, a relatively poor relaxation. The partial elementarity of the routes is enforced by non-robust Strong Degree Cuts. Robust cuts from two-index formulation and non-robust Strengthened Capacity Cuts and Subset Row Cuts are also separated.
   - The enumeration of elementary routes is directed to a pool of columns. As soon as the duality gap is sufficiently small to produce a pool with reasonable size (a few million routes), the pricing starts to be performed by inspection. From this point, an aggressive separation of non-robust cuts can be performed, leading to very small gaps.

   The reported computational results are very consistent. In particular, a hard instance from the literature, M-n151-k12 (151 vertices, 12 vehicles), was solved to optimality, setting a new record.

6. **Røpke [39]** went back to robust BCP. The main differences from [19] are the following:

   - Instead of q-routes without k-cycles, the more effective ng-routes are used.

- A sophisticated and aggressive strong branching is performed, drastically reducing the average size of the enumeration trees.

The overall results are comparable with the results in [12] and [4]. A long run of that algorithm could also solve M-n151-k12.

7. **Contardo and Martinelli [14]** improved upon Contardo [12]:
   - Instead of q-routes without 2-cycles, ng-routes are used. Moreover, the performance of the dynamic programming pricing was enhanced by the use of the DSSR technique [38].
   - Edge variables are fixed by reduced costs, using the procedure proposed in [23].
8. Finally, **Pecin et al. [32]** proposed a BCP that incorporates elements from all the previous algorithms, usually enhanced and combined with new elements:
   - The most important original contribution is the introduction of the limited memory Subset Row Cuts. They are a weakening of the traditional Subset Row Cuts that can be dynamically adjusted, making them much less costly in the pricing, and yet without compromising their effectiveness.
   - It uses capacity indices [33] in its underlying formulation. This allows a fixing of variables by reduced costs that is superior to fixing in [23].
   - The columns in the BCP are associated with ng-routes. The dynamic programming pricing uses bidirectional search that differs a little from that proposed in [37] because the concatenation phase is not necessarily performed at half of the capacity. A column generation stabilization by dual smoothing [34] may also be employed.
   - The BCP hybridizes branching with route enumeration. Actually, it performs an aggressive hierarchical strong branching.
   - When the addition of a round of non-robust cuts makes the pricing too slow, the BCP performs a rollback. The offending cuts are removed even if the lower bound of the node decreases.

   The algorithm could solve all the classical instances from the literature with up to 200 vertices in reasonable times, including the hard instances M-n200-k17 and M-n200-k16.

3.3 ▪ Formulations

This section revisits the CVRP formulations that may be considered as starting points in the design of the algorithms mentioned in Section 3.2. Let $G = (V, E)$ be a complete graph where $V = \{0, \ldots, n\}$ is the vertex set and E is the arc set. Vertices in $N = \{1, \ldots, n\}$ correspond to the *customers*, whereas vertex 0 corresponds to the *depot*. A non-negative cost, c_{ij}, is associated with each edge $(i, j) \in E$ and represents the *travel cost* spent to go from vertex i to vertex j. Edges are also indicated through a single index $e = (i, j)$. Let q_i denote the *demand* of customer i for $i \in N$, q_0 is defined as 0, and Q denote the *capacity* of each vehicle k in the set K of available vehicles. Given a vertex set $S \subseteq V$, let $\delta(S)$ denote the set of edges $e \in E$ which have only one endpoint in S. As usual, when a single vertex $i \in V$ is considered, we write $\delta(i)$ rather than $\delta(\{i\})$. For any set $S \subseteq V$, let $q(S) = \sum_{i \in S} q_i$, and $r(S) = \lceil q(S)/Q \rceil$. Further details on formulations for the CVRP can be found in Chapters 1 and 2.

3.3.1 ▪ The Two-Index Formulation

The classical *two-index formulation* is also known as *edge formulation* and is denoted as VRP2 in Chapter 1. The formulation, proposed by Laporte and Nobert [26], uses variables x_e that indicate how many times an edge $e \in E$ is traversed. In its summation form it is defined as

$$\text{(VRP2)} \quad \text{minimize} \sum_{e \in E} c_e x_e \tag{3.1}$$

$$\text{s.t.} \sum_{e \in \delta(i)} x_e = 2 \qquad \forall\, i \in N, \tag{3.2}$$

$$\sum_{e \in \delta(0)} x_e = 2|K|, \tag{3.3}$$

$$\sum_{e \in \delta(S)} x_e \geq 2r(S) \qquad \forall\, S \subseteq N, S \neq \emptyset, \tag{3.4}$$

$$x_e \in \{0,1\} \qquad \forall\, e \in E \setminus \delta(0), \tag{3.5}$$

$$x_e \in \{0,1,2\} \qquad \forall\, e \in \delta(0). \tag{3.6}$$

Constraints (3.2) and (3.3) are the degree constraints for the customers and the depot, respectively. Constraints (3.4) are the so-called *Rounded Capacity Cuts* (RCCs). In spite of the fact that $r(S)$ is only a lower bound for the minimum number of vehicles that must visit S, VRP2 is not a relaxation but a complete formulation for the CVRP.

3.3.2 ▪ The Set Partitioning Formulation

Balinski and Quandt [5] proposed a formulation where a binary variable is associated with each possible route respecting the capacity constraint. Let Ω be the set of routes, given by a sequence of edges that describe a path from the depot to the customers and back. The cost c_r of a route $r \in \Omega$ is given by the sum of the cost of the edges in its path. Let a_{ir} represent the number of the times customer i is visited by route r. Finally, variable λ_r indicates whether a route $r \in \Omega$ is used or not. The *set partitioning formulation* is denoted as VRP4 in Chapter 1 and, in its summation form, is as follows:

$$\text{(VRP4)} \quad \text{minimize} \sum_{r \in \Omega} c_r \lambda_r \tag{3.7}$$

$$\text{s.t.} \sum_{r \in \Omega} a_{ir} \lambda_r = 1 \qquad \forall\, i \in N, \tag{3.8}$$

$$\sum_{r \in \Omega} \lambda_r = |K|, \tag{3.9}$$

$$\lambda_r \in \{0,1\} \qquad \forall\, r \in \Omega. \tag{3.10}$$

The objective function (3.7) minimizes the overall cost of the selected routes. Constraints (3.8) guarantee that each customer is serviced by exactly one route. Constraint (3.9) imposes the use of $|K|$ vehicles. Finally, constraints (3.10) force the variables to be binary.

Balinski and Quandt defined Ω as the set of elementary routes, those visiting a customer at most once. Therefore, in this definition, coefficients a_{ir} are always binary. Formulation VRP4 has two important characteristics. The positive one is its gap of

integrality, which is consistently observed to be reasonably small. The negative one is the exponential growth of the number of columns with the number of customers. Researchers seeking to take advantage of the former characteristic had to deal with the huge number of columns that instances with a few dozen customers already have. This led to the, today central in vehicle routing exact algorithms, use of column generation techniques. The pricing subproblem requires finding minimum cost capacitated elementary routes over a graph with both positive and negative edge costs, a strongly NP-hard problem. The cost of an edge $e = (u, v) \in E$ in those problems is given by its reduced cost $\bar{c}_e = c_e - (\pi_u + \pi_v)/2$, where π_i, $i \in N$, are the dual variables of constraints (3.8) and π_0 is the dual variable of constraint (3.9).

In the early 1980's, it was already observed that the formulation VRP4 could be turned into a more practical tool by changing the definition of the Ω set in order to obtain a more tractable pricing problem. Suppose that Ω is enlarged to contain all walks leaving and returning to the depot such that the sum of the demands of the visits of the walk does not exceed the vehicle capacity. The same customer can be visited more than once, but its demand will be counted again in each additional visit. The coefficient a_{ir} would assume a value equal to the number of visits to customer i in the walk. These walks were coined q-routes and used in the Lagrangian method proposed in Christofides, Mingozzi, and Toth [10]. The associated pricing problem is now weakly NP-hard, and a pseudo-polynomial dynamic programming algorithm for its resolution is available. This enlargement of Ω still lets VRP4 be a formulation for the CVRP, since constraints (3.8) forbid $\lambda_r = 1$ for any route r with a coefficient $a_{ir} > 1$. However, the lower bound given by the linear relaxation of the new model can be significantly worse.

The choice of set Ω plays an important role in the design of algorithms based on formulation VRP4. One idea is to impose some controlled amount of partial elementarity on the routes, in order to obtain bounds as close as possible to the elementary route bound, while still keeping the associated pricing problem tractable. There is an alternative that only allows q-routes without k-cycles, subcycles of length k or less, for some chosen k, as will be described in Section 3.5.2. Another more recent alternative is working with the so-called ng-routes, discussed in Section 3.5.3.

Anyway, even if Ω only contains elementary routes, the bounds given by formulation VRP4 (gaps between 1% and 4% are typical in the instances from the literature) are *not* good enough to be the basis of efficient exact algorithms. For this purpose, VRP4 must be reinforced with additional cuts. In fact, all the recent methods mentioned in Section 3.2 perform some kind of combined generation of both columns and cuts. Section 3.4 will present the main families of cuts used in those methods. Some of those cuts are robust; they do not change the structure of the pricing. However, the stronger cuts are nonrobust. In this last case, the algorithm designer has to balance the strength of the cut (its capacity of improving the bounds) against its impact in the pricing complexity.

3.3.3 ▪ Capacity-Indexed Formulation

This extended formulation for the *Asymmetric CVRP* (ACVRP) (therefore valid for the CVRP) was presented in Pessoa, Poggi de Aragão, and Uchoa [33] (also in Godinho et al. [20] for the unitary demand case). Now $G_D = (V, A)$ is a complete directed graph with arc costs c_a, $a \in A$. For any set $S \subseteq V$, $\delta^-(S) = \{(i, j) \in A : i \in V \setminus S, j \in S\}$, and $\delta^+(S) = \{(i, j) \in A : i \in S, j \in V \setminus S\}$. Define binary variables x_a^q indicating 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 q. The arcs returning to the depot must have $q = 0$. The

Capacity-Indexed Formulation (CIF) is

$$\text{(CIF)} \quad \text{minimize} \sum_{a\in A} c_a \sum_{q=0}^{Q} x_a^q \tag{3.11}$$

$$\text{s.t.} \sum_{a\in\delta^-(\{i\})} \sum_{q=1}^{Q} x_a^q = 1 \qquad \forall\, i \in N, \tag{3.12}$$

$$\sum_{q=1}^{Q} \sum_{i\in N} x_{0i}^q = |K|, \tag{3.13}$$

$$\sum_{a\in\delta^-(\{i\})} x_a^q - \sum_{a\in\delta^+(\{i\})} x_a^{q-q_i} = 0 \quad \forall\, i \in N,\ q = q_i,\dots,Q, \tag{3.14}$$

$$x_a^q \in \{0,1\} \qquad \forall\, a \in A,\ q = 1,\dots,Q, \tag{3.15}$$

$$x_{(i,0)}^q = 0 \qquad \forall\ i \in N,\ q = 1,\dots,Q. \tag{3.16}$$

Equations (3.12) and (3.13) are customer and depot degree constraints. Balance equations (3.14) state that if an arc with index q enters vertex i, then an arc with index $q - q_i$ must leave i. This both prevents cycles and routes with total demand greater than Q. The interest in the CIF lies in the fact that cuts expressed over its variables can be added into formulation VRP4 in a robust way, at least if the pricing is being solved by dynamic programming, as will be shown in Section 3.4.6.

3.4 ▪ Valid Cuts

This section discusses the main families of inequalities that have been used in recent works in order to reinforce formulation VRP2.

3.4.1 ▪ Cuts over the Edge Variables

A general inequality $\sum_{e\in E} \alpha_e x_e \geq b$ for model VRP2 can be included in the formulation VRP4 as $\sum_{r\in\Omega}(\sum_{e\in E} \alpha_e a_{er})\lambda_j \geq b$, where a_{er} is the number of times that edge e appears in route r. In a column generation approach for solving the linear relaxation of VRP4, this additional cut contributes to the computation of reduced cost $\overline{c}_e$ of an edge $e \in E$ with the value $-\alpha_e\beta$, where β is the corresponding dual variable. Therefore, the structure of the pricing problem is not changed.

Besides RCCs (3.4), there are several known families of valid CVRP cuts over the edge variables. The package CVRPSEP (see Lysgaard [28]) contains effective heuristic separation procedures for the following families of cuts: rounded capacity, framed capacities, strengthened combs, multistars, and extended hypotours. While all those families may play a significant role in Branch-and-Cut algorithms over the formulation VRP2 (like Lysgaard, Letchford, and Eglese [29]), only RCCs and (by a much smaller degree) strengthened comb cuts can strengthen VRP4 in a significant way (see [19]). It seems that most cuts in the other families are already implicitly given by constraints (3.8)–(3.9). Indeed, Letchford and Salazar González [27] proved that all generalized large multistar cuts are implied by those constraints even if the definition of Ω includes all q-routes.

3.4.2 ▪ Strengthened Capacity Cuts

Baldacci, Christofides, and Mingozzi [3] introduced a family of cuts defined over the variables of the formulation VRP4. For each $S \subseteq N$ and for each $r \in \Omega$, define binary coefficient ζ_{Sr} as 1 if and only if the route r visits at least one vertex in S. The *Strengthened Capacity Cuts* (SCCs) are

$$\sum_{r\in\Omega} \zeta_{Sr}\lambda_r \geq r(S) \qquad \forall\, S \subseteq N. \tag{3.17}$$

Remark that an RCC over S corresponds to $\sum_{r\in\Omega} a_{Sr}\lambda_r \geq r(S)$, where a_{Sr} counts how many times the route r enters (or leaves) S. Inequalities (3.17) are stronger because they are not "fooled" by routes that enter and leave S more than once. On the other hand, SCCs are non-robust, since they change the pricing subproblem for the linear relaxation of VRP4. In order to continue solving it by dynamic programming, it is necessary to include an additional binary dimension in each label indicating whether a partial route has already visited S or not. Notwithstanding this fact, the algorithms in [3, 4] and [12, 14] have successfully used (in a controlled way, limiting the number of separated cuts) SCCs. It can also be observed that if there exists a set $S' \subset S$ where $r(S') = r(S)$, then the SCC corresponding to S' dominates the SCC corresponding to S. This means that only the cuts corresponding to minimal sets (with respect to the function $r(\cdot)$) need to be separated.

3.4.3 ▪ Subset Row Cuts

Jepsen et al. [25] introduced the following family of cuts defined over the variables of the formulation VRP4. Given a set $C \subseteq N$ and a multiplier p, $0 < p < 1$, the (C, p)-Subset Row Cut (SRC)

$$\sum_{r\in\Omega} \left\lfloor p \sum_{i\in C} a_i^r \right\rfloor \lambda_r \leq \lfloor p|C| \rfloor \tag{3.18}$$

is valid, since it can be obtained by a Chvátal–Gomory rounding of the corresponding constraints in (3.8).

The cuts where $|C| = 3$ and $p = 1/2$ are called *3-Subset Row Cuts* (3SRCs). The earlier algorithm by Baldacci, Christofides, and Mingozzi [3] used clique cuts, which are more general than 3SRCs. However, 3SRCs are favored in more recent works [4] and [12, 14], since they seem to be more suited to a column generation context, having less impact in the pricing subproblem. Each 3SRC requires an additional binary dimension in each dynamic programming label to indicate the parity of the number of visits made by a route to a vertex in a triplet C. The successful use of those cuts depends on a carefully controlled separation, in order to avoid pricing intractability.

Baldacci, Mingozzi, and Roberti [4] have also used a weakened variant of the 3SRCs for the case where Ω only contains elementary routes. Once a binary coefficient ξ_{Cr} is defined as 1 if and only if the route r visits at least one edge with both endpoints in C, the corresponding weakened 3SRC is

$$\sum_{r\in\Omega} \xi_{Cr}\lambda_r \leq 1. \tag{3.19}$$

Since an elementary route can only visit two such edges if they are consecutive, it is easy to include the effect of the dual variables of weakened 3SRCs in a dynamic programming pricing. Therefore, there are no restrictions to its separation. A variant of this cut, suitable to the case where Ω contains non-elementary routes, needs to define ξ_{Cr} as the number of times that a route r visits non-consecutive edges with both endpoints in C.

3.4.4 ▪ Strong Degree Cuts

Contardo, Cordeau, and Gendron [13] introduced a family of cuts that correspond to SCCs over sets S of cardinality 1. Given a vertex $i \in N$ and for $r \in \Omega$, define binary coefficient ζ_{ir} as 1 if and only if route r visits i. The *Strong Degree Cut* (SDC) is

$$\sum_{r\in\Omega} \zeta_{ir}\lambda_r \geq 1. \tag{3.20}$$

An SDC can only be non-redundant if the definition of set Ω allows non-elementary routes. In this case, the effect of SDCs is forbidding routes with cycles at certain vertices. Therefore, they are alternative ways of enforcing partial route elementarity. As happens with the more general SCCs, each SDC requires one additional binary dimension in the dynamic programming labels.

Contardo [12] and Contardo and Martinelli [14] also used weakened versions of the strong degree cuts. For an integer k, $i \in N$ and for $r \in \Omega$, let ν_{irk} be the number of times that route r visits i with at least k vertices between two consecutive visits. The corresponding *k-Cycle Elimination Cut* (k-CEC) is

$$\sum_{r\in\Omega} \nu_{irk}\lambda_r \geq 1. \tag{3.21}$$

The effect of this cut is forbidding routes that have cycles over i of length k or less. The k-CECs have less impact in the dynamic programming pricing than the SDCs. The information that a vertex i was already visited is "forgotten" after k visits to other vertices.

3.4.5 ▪ Limited Memory Subset Row Cuts

Introduced in Pecin et al. [32], this generalization of a (C, p)-SRC requires an additional set M, $C \subseteq M \subseteq N$. The limited memory (C, M, p)-Subset Row Cut (lm-SRC) can be written as

$$\sum_{r\in\Omega} \alpha(C, M, p, r)\lambda_r \leq \lfloor p|C| \rfloor, \tag{3.22}$$

where the coefficients α are computed by the following procedural function:

Function $\alpha(C, M, p, r)$
$coeff \leftarrow 0$, $state \leftarrow 0$
for every vertex $i \in r$ (in order) **do**
  if $i \notin M$ **then**
    $state \leftarrow 0$
  else if $i \in C$ **then**
    $state \leftarrow state + p$
    if $state \geq 1$ **then**
      $coeff \leftarrow coeff+1$, $state \leftarrow state - 1$
return *coeff*

When $M = N$, the Function α will return $\lfloor p \sum_{i\in C} a_i^r \rfloor$ and the lm-SRC will be identical to an SRC. On the other hand, when M is not equal to N, the lm-SRC may be a weakening of its corresponding SRC. This happens because every time the route r leaves M, the variable $state$ is reset to zero, potentially decreasing the returned coefficient.

The potential advantage of the lm-SRCs over classical SRCs is their much reduced impact on the labeling algorithms used in the pricing, when $|M| \ll |N|$. The reasons for that reduction will be explained in Section 3.5.4. In order to obtain small memory sets, the separation of lm-SRCs presented in [32] used the following strategy. First, it identifies a violated (C, p)-SRC. Then, it determines a *minimal set M such that the lm-(C, M, p)-SRC has the same violation*. In practice, even on instances with hundreds of customers, those minimal sets seldom have cardinality larger than 15.

Given a base set $C \subseteq N$, for each integer d, $1 \leq d \leq n$, define a non-negative integer variable y_C^d as the sum of all variables λ_r such that $\sum_{i \in C} a_i^r = d$. Variables with $d > |C|$ can only be non-zero if Ω contains non-elementary routes. The interesting SRCs (and the corresponding lm-SRCs), for sets C with cardinality up to 5, are the following:

- The cuts where $|C| = 3$ and $p = 1/2$, the 3SRCs, can be expressed as $y_C^2 + y_C^3 + 2y_C^4 + 2y_C^5 + \cdots \leq 1$. The weak 3SRCs in [4] are equivalent (when all routes are elementary) to lm-3SRCs with $M = C$.
- Taking $|C| = 1$ and $p = 1/2$, the 1-Subset Row Cuts (1SRCs) $y_C^2 + y_C^3 + 2y_C^4 + \cdots \leq 0$ are obtained. They are equivalent to the SDCs $y_C^1 \geq 1$; both families forbid cycles over a vertex i ($C = \{i\}$). The weaker k-CECs [12, 14] only forbid cycles over i of size k or less. An lm-1SRC is a different kind of weakening; it forbids cycles over i contained in the set M. Of course, all these cuts can only be useful when the Ω set contains non-elementary routes.
- The cuts where $|C| = 4$ and $p = 2/3$ are 4-Subset Row Cuts (4SRCs), expressed as $y_C^2 + 2y_C^3 + 2y_C^4 + 3y_C^5 + 4y_C^6 + \cdots \leq 2$.
- There are two interesting families of cuts with $|C| = 5$. Those with $p = 1/3$ will be called 5,1SRCs, $y_C^3 + y_C^4 + y_C^5 + 2y_C^6 + \cdots \leq 1$; whereas those with $p = 1/2$ are 5,2SRCs, having the format $y_C^2 + y_C^3 + 2y_C^4 + 2y_C^5 + 3y_C^6 + \cdots \leq 2$. The latter family was already used in [12, 14].

3.4.6 ▪ Cuts over the CIF Variables

A general inequality $\sum_{a \in A} \sum_{q=0}^{Q} \alpha_a^q x_a^q \geq b$ for the CIF can be included in VRP4 as $\sum_{r \in \Omega} (\sum_{a \in A} \sum_{q=0}^{Q} \alpha_a^q a_{ar}^q) \lambda_j \geq b$, where a_{ar}^q is the number of times (always 0 or 1) that arc a appears with capacity q in route r. This additional cut contributes to the computation of an arc-capacity reduced cost $\overline{c}_a^q$ with the value $-\alpha_a^q \beta$, where β is the corresponding dual variable. This more general reduced cost structure does not affect the complexity of the dynamic programming algorithms usually employed for pricing routes. This happens because those algorithms already have a dimension for the capacity consumption in their labels.

There are potential gains of using cuts defined over the capacity-indexed extended variable space. For example, the reasoning behind the RCCs and SCCs assumes that every route visiting a set S can contribute with up to Q units for satisfying the demand $q(S)$. However, if the route also visits other vertices outside S, the actual contribution can be much smaller than Q. The *Extended Capacity Cuts* (ECCs) (see Pessoa, Poggi de Aragão, and Uchoa [33] and Pessoa, Uchoa, and Poggi de Aragão [35]) are a generalization of the RCCs devised to capture that fact, taking advantage of the knowledge of the actual route capacity when entering and leaving S.

As surveyed in Uchoa [41], ECCs (and some other families of cuts related to the CIF) have been successfully used in a number of vehicle routing variants, on parallel machine

scheduling, and even on network design problems. However, the results for the CVRP reported in Pessoa, Poggi de Aragão, and Uchoa [33] and Uchoa [41] are more modest. In this case, the inclusion of ECCs in a robust BCP algorithm provides root bounds that are inferior to those that can be obtained by adding non-robust SCCs and SRCs, not good enough for solving instances larger than those already solved by previous algorithms. Nevertheless, we still believe in the possibility that cuts over the CIF can be a useful addition to future CVRP algorithms.

3.5 ▪ Pricing

This section presents the main ideas proposed in the literature for pricing routes, relaxed or not, for the formulation VRP4, with or without additional non-robust cuts.

3.5.1 ▪ *q*-Routes and Elementary Routes

The pricing subproblem for solving the linear relaxation of VRP4 by column generation, defined in Section 3.3, has been modeled by many authors as a *Shortest Path Problem with Resource Constraints* (SPPRC). The problem is defined over directed graphs $(V \cup \{n+1\}, A)$, where vertex $n+1$ is a copy of the depot vertex 0; and A is the complete arc set minus $\{(0,n+1),(n+1,0)\}$. For each $(i,j) \in A$, there are costs $\overline{c}_{ij}$, unrestricted in sign. Define a set H of resources, numbered from 0 to $|H|-1$. Each $h \in H$ has a positive integer availability C^h. For each arc $(i,j) \in A$ and each $h \in H$, d_{ij}^h are non-negative integer resource consumption quantities. The SPPRC has the objective of finding the shortest, not necessarily elementary, path $P = (0, i_1, i_2, \ldots, i_p, n+1)$ such that, for each resource $h \in H$, the total consumption T^h does not exceed C^h. The pricing of q-routes corresponds to an SPPRC with a single resource 0, such that $C^0 = Q$. For each $(i,j) \in A$, $d_{ij}^0 = q_j$ and $\overline{c}_{ij}$ is equal to the reduced cost of the corresponding edge. However, the pricing of elementary routes requires the definition of n additional resources, such that $C^i = 1$ for each $i \in N$. For each arc $(u,v) \in A$ and each $i \in N$, d_{uv}^i is defined as 1 if $i = v$ and 0 otherwise.

A forward dynamic programming label setting algorithm for the SPPRC represents a path P starting at 0 and ending at vertex $i \in V \cup \{n+1\}$ as a label $L(P) = (i, \overline{c}(P), T^0(P), \ldots, T^{|H|-1}(P))$, where $\overline{c}(P)$ is the cost and $T^h(P)$, $h \in H$, are the resource consumptions in the path. Given a label $L = L(P)$, we define $v(L)$ as the ending vertex of P, $\overline{c}(L) = \overline{c}(P)$ and $T^h(L) = T^h(P)$, for each $h \in H$, and $P(L) = P$. The list of labels $\mathscr{L}$ is initialized with a single label representing a null path that ends at the depot with zero cost and no consumption of resources. As it iterates, the algorithm tries to extend a label $L \in \mathscr{L}$, creating new labels corresponding to the paths obtained by adding a vertex in $(N \cup \{n+1\}) \setminus \{v(L)\}$ to $P(L)$, being successful only when all resource constraints are still satisfied. Each newly created label L', if any, is added to $\mathscr{L}$ and linked to L by backward pointers; we say that $pred(L') = L$. When it is not possible to extend any label in $\mathscr{L}$, the minimum cost labels ending at $n+1$ correspond to optimal solutions of the SPPRC.

When designing a dynamic programming label setting algorithm for the SPPRC, the main concern is how to mitigate the efficiency deterioration as the number of resources grows (the "curse of dimensionality"), keeping the list of labels $\mathscr{L}$ as small as possible. It is clear that if two labels ending at the same vertex also have the same resource consumption, only the least costly one (ties broken arbitrarily) should be kept. For pricing q-routes, when the algorithm considers only the capacity resource, this basic dominance rule is enough to show that at most $n(Q+1)$ labels need to kept, giving a total complexity of

$O(n^2Q)$ operations. This pseudo-polynomial complexity is not satisfactory when Q is very large. In those cases, it is recommended to scale the capacity and the demands, as done in Subramanian et al. [40]. However, for pricing elementary routes, this reasoning would only prove an exponential complexity of $O(2^nQ)$ for the label setting algorithm. We now present some of the ideas that have been introduced (Boland, Dethridge, and Dumitrescu [6], Chabrier [7], Feillet et al. [17], Righini and Salani [37], and Righini and Salani [38]) to tame the worst-case complexity of that algorithm, enough to turn it into a practical tool for pricing elementary routes on many instances.

1. **Stronger Dominance Rules.** The basic dominance rule can be improved by considering the monotonic effect of the resource consumption in the possible extensions of a label. If two labels L and L', both ending at the same vertex i, have $\overline{c}(L) \leq \overline{c}(L')$ and $T^h(L) \leq T^h(L')$, for $h \in H$, then L dominates L'. Therefore, L' can be removed from the set of labels. Whenever a new label L is created, it is interesting to check whether L is not dominated by some other label already in $\mathscr{L}$ or whether L dominates some labels in $\mathscr{L}$. While this is conceptually simple, efficient implementation may require proper data structures.

2. **Bidirectional Search.** The 2^n factor in the complexity is far too pessimistic for typical CVRP instances, when the capacity Q limits the number of customers in most routes to a value that is not much larger than $n/|K|$. In fact, the ratio $n/|K|$ is indeed a strong predictor of the algorithm performance. In order to better profit from this effect, Righini and Salani [37] proposed not expanding the labels with a capacity consumption larger than $\lceil Q/2 \rceil$. This usually reduces a lot the final size of $\mathscr{L}$. The remaining routes can be obtained by a concatenation phase, taking advantage of the symmetry of the CVRP. Each pair of labels L_1 and L_2 in $\mathscr{L}$ such that $T^h(L_1) + T^h(L_2) \leq C^h$, for each $h \in H$, corresponds to a route with cost $\overline{c}(L_1) + \overline{c}(L_2) + \overline{c}_{v(L_1)v(L_2)}$. Actually, since the concatenation phase can be costly, it is not clear that bidirectional search is always superior to the traditional forward search. For example, in the recent work by Martinelli, Pecin, and Poggi [30] the latter alternative was preferred.

3. **Completion Bounds.** In the pricing context, one is only interested in SPPRC solutions corresponding to routes with negative reduced cost. Given a label $L \in \mathscr{L}$, let $CB(L)$ be a completion bound on L, i.e., a lower bound on the cost of all routes that can be obtained as extensions of L. If $CB(L) \geq 0$, then L can be removed from $\mathscr{L}$. For example, labels from a previous run of the dynamic programming that considers only a subset of the resources in H provide valid completion bounds. A good completion bound can reduce considerably the final size of $\mathscr{L}$.

4. ***Decremental State Space Relaxation*** **(DSSR).** This technique was proposed independently by Boland, Dethridge, and Dumitrescu [6] (where it is called State Space Augmenting Algorithm) and by Righini and Salani [38] (where the name DSSR was proposed). It starts by running the label setting algorithm by only considering the capacity resource (i.e., pricing q-routes). The obtained solution is likely to contain cycles. The resources for the vertices that are visited more than once in that solution are then included in the next run of the label setting algorithm. As the algorithm progresses, the set of customers that cannot be revisited is enlarged, until elementary routes are obtained.

3.5.2 ▪ *q*-Routes with *k*-Cycle Elimination

The q-routes with k-cycle elimination are those where multiple visits to a vertex $i \in N$ are only allowed if at least k other costumers are visited between successive visits. Christofides, Mingozzi, and Toth [11] already realized that 2-cycle elimination in the definition of Ω improves significantly the bounds obtainable from the VRP4 relaxation, with a modest impact in the complexity of the pricing subproblem, as explained next. Each pair (i,q) where $i \in N$ and $q \in \{0,1,\ldots,Q\}$ defines a bucket $B(i,q)$. A label $L \in \mathscr{L}$ is associated with $B(i,q)$ if $v(L) = i$ and $T^0(L) = q$. When pricing ordinary q-routes it is clear that the basic dominance rule assures that each bucket contains at most one label in $\mathscr{L}$. However, when performing 2-cycle elimination, a second label may need to be kept in each bucket. In this case, the least cost label L in a bucket does not dominate the second least costly label L' in the same bucket if $pred(L) = j$ and $pred(L') \neq j$ because, as the extension of L to j is forbidden, it is not possible to prove that the extension of L' to j is not part of the optimal solution.

However, when performing k-cycle elimination for $k \geq 3$, deciding which labels need to be kept in each bucket becomes quite intricate. Irnich and Villeneuve [24] propose an Intersection Algorithm for that purpose. Their conjecture (proved only for $k = 3$) is that this algorithm will keep at most $k!$ labels per bucket. Anyway, the average number of labels per bucket actually kept during a run may vary a lot, depending on the instance. Hoshino and de Souza [22] proposed a more efficient implementation where non-dominated labels are identified by deterministic finite automata; there is an automaton for each value of k.

Experiments with q-routes with k-cycle elimination in Fukasawa et al. [19] suggest that it is not worthy to use values of k above four. From this point, the pricing indeed becomes much slower. Unhappily, 4-cycle elimination is not enough to obtain bounds really close to the elementary bounds in some instances.

3.5.3 ▪ *ng*-Routes

A simple alternative to cycle elimination for obtaining partial elementarity in the routes would be to select a subset S of customers to forbid being revisited. The label setting algorithm for the SPPRC would be run defining unitary resources only for those customers. However, this is not likely to produce near-elementary routes if $|S|$ is significantly smaller than $|N|$. Baldacci, Mingozzi, and Roberti [4] devised a better way to impose partial elementarity. Instead of carrying the information that a certain vertex was already visited by a path in all its extensions, a more limited memory mechanism is proposed. It takes advantage of the fact that the cycles that are likely to appear when pricing non-elementary routes, even when they are too long to be efficiently avoided by k-cycle elimination, only use edges with small cost and are confined to relatively small "neighborhoods" in the graph.

For each customer $i \in N$, let $N_i \subseteq N$ be the ng-set of i, defining its neighborhood. This may stand for the $|N_i|$ (this cardinality is decided a priori) closest customers and includes i itself. When extending a label L_1 with $v(L_1) = i$ to a customer j, the visit to i is only remembered in the new label L_2 if i belongs to N_j. In this case, L_2 cannot be extended back to i. However, the extension of L_2 to another vertex k will only carry the information about i in the new label L_3 if i also belongs to N_k. At some point, the visit to i is "forgotten" and further extensions may visit i again, forming a cycle. In other words, an ng-route allows a cycle over a vertex i only if this cycle passes by a vertex v such that $i \notin N_v$.

The modified label setting algorithm for pricing ng-routes works as follows. Let $P = (0, i_1, i_2, \ldots, i_p)$ be a path associated with a label $L(P)$ and having $V(P)$ as the set of visited customers. Define the set of forbidden extensions of P as

$$\Pi(P) = \left\{ i_k \in V(P) \backslash \{i_p\} : i_k \in \bigcap_{s=k+1}^{p} N_{i_s} \right\} \cup \{i_p\}.$$

The label $L(P)$ can be extended to a customer i_{p+1} only when $i_{p+1} \notin \Pi(P)$ and all the resource constraints are satisfied. The extension to customer i_{p+1} creates a new label $L(P')$ corresponding to path $P' = (0, \ldots, i_p, i_{p+1})$ and forbidden extensions $\Pi(P') = (\Pi(P) \cap N_{i_{p+1}}) \cup \{i_{p+1}\}$. The routes that can be obtained by this algorithm are ng-routes. A label $L(P_1)$ can only dominate label $L(P_2)$ if the additional condition $\Pi(P_1) \subseteq \Pi(P_2)$ is satisfied. Assuming that the only resource considered is the capacity, the maximum number of labels in a bucket $B(i, q)$ is limited to $2^{|N_i|-1}$. Therefore, the algorithm remains pseudo-polynomial when the ng-sets have a fixed size.

In order to handle ng-sets with large cardinality, Martinelli, Pecin, and Poggi [30] proposed an adaptation of the DSSR concept. The labeling algorithm starts with the ng-sets relaxed into singletons. If a forbidden cycle over a vertex i is detected in the solution, i is restored in the ng-sets of the vertices in the cycle and the labeling is run again. The labels of the previous run are used for computing completion bounds. This is done until the desired ng-route is found. In that way, ng-sets with size 64, large enough to enforce elementarity, can be efficiently handled on most instances with up to 200 customers.

3.5.4 ▪ Pricing with Non-robust Cuts

The adaptation of the label setting algorithm for handling non-robust cuts requires additional dimensions in the labels, typically one dimension per cut. In the literature (for instance, Jepsen et al. [25]), those dimensions are still called "resources". However, they are not resources in the strict sense defined in Section 3.5.1. First, in the SPPRC definition, the consumption of a resource is monotonically non-decreasing along a path. This is not the case, for example, of the dimensions added for handing SRCs. More importantly, while the exhaustion of a resource forbids certain path extensions, dimensions corresponding to cuts never forbid path extensions. Actually, such a label dimension is only used to reward/penalize certain extensions, determining whether the value of the dual variable associated with a cut should be subtracted from the cost of the new labels.

Assume that the formulation VRP4 was enhanced with (i) the SCCs associated with the customer sets S in a collection $\mathcal{S}$, having dual variables π_S; (ii) the 3SRCs associated with the triplets C in a collection $\mathcal{C}$, having dual variables σ_C. Remark that some of the SCCs can be SDCs. A label L has additional binary dimensions D_S, $S \in \mathcal{S}$, with value 1 if $P(L)$ already visited some vertex in S and 0 otherwise; and binary dimensions D_C, $C \in \mathcal{C}$, indicating the parity of the number of visits made by $P(L)$ to a vertex in C. Whenever a label L is extended to a vertex j belonging to a set S where $D_S(L) = 0$, the value π_S is subtracted from the cost of the new label. As $\pi_S \geq 0$, this means that this particular extension is being rewarded. In a similar way, when L is extended to a vertex j that belongs to a triplet C where $D_C(L) = 1$, σ_C is subtracted from the cost of the new label. As $\sigma_C \leq 0$, this corresponds to a penalization. The dominance rule in the label setting algorithm also needs to be modified. Given labels L and L' with $v(L) = v(L')$ and $T^h(L) \leq T^h(L')$, for $h \in H$, it can be shown that L dominates L' if the following

additional condition is satisfied:

$$\overline{c}(L) \leq \overline{c}(L') - \sum_{S \in \mathscr{S}: D_S(L) > D_S(L')} \pi_S + \sum_{C \in \mathscr{C}: D_C(L) > D_C(L')} \sigma_C. \tag{3.23}$$

The second and third terms in the right-hand side are lower bounds on what the extensions of L' can gain over the extensions of L, by still being able to collect the rewards of visiting some sets in $\mathscr{S}$ already visited by $P(L)$ or by avoiding some penalizations that the extensions of L may incur by revisiting some triplets in $\mathscr{C}$ already visited by $P(L)$ an odd number of times. In general, the presence of too many such non-robust cuts ruins the performance of the labeling algorithm. This happens because the dual variable of each additional cut may interfere with condition (3.23), making it less likely to be satisfied. At some point, an exponential explosion in the number of non-dominated labels is observed.

Anyway, Contardo, Cordeau, and Gendron [13] showed that the addition of non-robust SDCs may be a more efficient way of imposing route elementarity than the traditional definition of resources for avoiding vertex revisitation. In the latter case, if a path P visits any vertex not visited by path P', $L(P)$ can never dominate $L(P')$. On the other hand, when SDCs are used, $L(P)$ can dominate $L(P')$ if $\overline{c}(L(P))$ is sufficiently smaller than $\overline{c}(L(P'))$ to compensate the dual variables corresponding to the vertices not visited in P'. The algorithm in Contardo [12] takes this idea to its logical extreme, only pricing q-routes without 2-cycles. The partial elementarity is obtained by the gradual introduction of non-robust k-CECs and SDCs. However, the algorithm in Contardo and Martinelli [14] obtained better results by pricing ng-routes with ng-sets of size 8 and only using k-CECs and SDCs for removing the larger cycles.

The labels from a previous run of the dynamic programming that does not consider the dimensions corresponding to non-robust cuts may not provide valid completion bounds, unless some adaptations are made. While the dual variables with negative values can be safely ignored, dual variables with positive values must be taken into account. Contardo, Cordeau, and Gendron [13] proposed underestimating the impact of dual variable π_S, $S \in \mathscr{S}$, by subtracting $\pi_S/2$ from the cost of all edges in $\delta(S)$. In order to obtain stronger completion bounds, [12] and [14] actually consider the dimensions corresponding to some of the cuts (those with larger absolute dual variable values) in the dynamic programming run.

Now we explain why the lm-SRCs introduced in [32], when their memory sets are small, can be much better handled by the labeling algorithms than the traditional SRCs. For simplicity, assume that the only non-robust cuts are the 3SRCs associated with pairs (C,M) in a collection $\mathscr{C}$, where C is a triplet and $M \supseteq C$ is a memory set, and having dual variables $\sigma_{C,M}$. As before, the labels have binary dimensions $D_{C,M}$ for each $(C,M) \in \mathscr{C}$. However, $L(D_{C,M})$ now indicates the parity of the number of visits made by $P(L)$ to a vertex in C, but only since $P(L)$ entered M for the last time. By definition, $L(D_{C,M})$ is zero on all labels L where $v(L) \notin M$. This means that only the dual variables $\sigma_{C,M}$ such that $v(L) \in M$ may interfere in the dominance condition:

$$\overline{c}(L) \leq \overline{c}(L') + \sum_{(C,M) \in \mathscr{C}: D_{C,M}(L) > D_{C,M}(L')} \sigma_{C,M}. \tag{3.24}$$

In practice, this means that many more cuts can be added before the explosion in the number of non-dominated labels is observed.

3.5.5 ▪ Column Generation Issues

Column generation is a technique for solving linear programs with a huge number of variables. Of course, there is no magic; its efficiency still depends on "how huge" this number is. Since the number of variables in VRP4 is roughly proportional to $\binom{n}{n/|K|}$, it is quite expected that the number of pricing iterations until convergence depends significantly on the average number of customers per route. This is clearly observed in practice. On instances from the literature where $n/|K|$ is small, convergence of the standard column generation procedure typically occurs in a few dozen iterations. On the other hand, on instances where is $n/|K|$ is large, convergence may require hundreds of iterations. Since those are precisely the instances where each pricing iteration is slower (especially with elementary or near-elementary routes), some kind of dual stabilization (see Du Merle et al. [16] or Pessoa et al. [34]) may help to mitigate the convergence problems.

Anyway, it is always advisable to accelerate the column generation by devising faster pricing heuristics. The relatively expensive exact pricing is only called when the pricing heuristics cannot find routes with negative reduced cost. Such effective heuristics (for example, see Fukasawa et al. [19]) can be obtained by modifying the label setting algorithm. The *scaling* technique consists of running the algorithm with modified demands $q'_i = \lceil q_i/g \rceil$ for $i \in N$ and modified capacity $Q' = \lfloor Q/g \rfloor$ for a chosen factor g. The *sparsification* technique consists of only extending labels by considering a chosen subset of the edges, those that are more likely to appear in routes with negative reduced cost. Finally, the *bucket pruning* technique consists of storing only a chosen (small) number of labels per bucket. This means that many non-dominated labels, those that are less likely to lead to optimal solutions, may be dropped.

3.6 ▪ Branching vs. Route Enumeration

The column and cut generation over the formulation VRP4 may finish with a duality gap, either because it is not possible to find additional violated cuts or because tailing-off, minimal improvements in the lower bound after a number of cut rounds, is detected. In the case of an algorithm that uses non-robust cuts there is another possibility: additional separation would make the pricing excessively expensive. The algorithms in Fukasawa et al. [19] and in Røpke [39] try to close that gap by following the classical BCP paradigm; a branching would be performed, and each child node would be solved by column and cut generation. On the other hand, the algorithms in Baldacci, Christofides, and Mingozzi [3], Baldacci, Mingozzi, and Roberti [4], Contardo [12], and Contardo and Martinelli [14] do not perform branching, at least not directly. Instead, they proceed to enumerate all elementary routes that may belong to an optimal solution, using reduced cost information. A set-partitioning problem with those routes is then solved by the generic Branch-and-Cut algorithm of a MIP solver.

3.6.1 ▪ Branching

Branching directly over the VRP4 variables is non-robust and would complicate the pricing subproblem in the children nodes. On the other hand, it is possible to perform the branching on the edge variables of the formulation VRP2. This may be done either over individual edges $e \in E$, imposing disjunctions like $(x_e \leq 0) \vee (x_e \geq 1)$, or, more generally, over sets $S \subseteq N$, imposing the disjunction $(\sum_{e \in \delta(S)} x_e = 2) \vee (\sum_{e \in \delta(S)} x_e \geq 4)$. In both cases, the branching constraints for the formulation VRP2 can be translated into robust cuts in VRP4.

The BCP in Fukasawa et al. [19] adopted the branching over sets (already used in Lysgaard, Letchford, and Eglese [29]). In order to better choose the branching set, this BCP used strong branching. Each set in a collection containing from 5 to 10 candidate sets was heuristically evaluated by applying a small number of column generation iterations in its children nodes. Remark that the exact evaluation of each candidate, performing full column generation (perhaps also cut generation) in both children nodes, would be too expensive.

The recent work by Røpke [39] showed that is possible to obtain major improvements in the BCP performance by performing a more sophisticated and aggressive strong branching. His extensive experiments were performed both on CVRP and VRPTW instances.

- The simpler branching over individual edges (arcs in the VRPTW case) was used.
- The strong branching procedure starts by performing a quick evaluation of 30 candidate edges, producing a ranking. The best ranked candidate is then fully evaluated and becomes the incumbent winner. Then, other candidates with good ranking are better evaluated, but only while they have a reasonable chance of beating the incumbent winner.
- It is possible to collect statistics along the enumeration tree to help the strong branching. In particular, the previous evaluations of an edge (in different nodes) are good predictors of future evaluations.

This strong branching was the key algorithmic element that allowed his robust BCP to solve the hard instance M-n151-k12, with optimum 1015, starting from the rather modest root lower bound of 1001.5. Another robust BCP with the same strong branching was the first algorithm to solve all the 56 VRPTW Solomon instances with 100 vertices. We remark that a strong branching for a BCP, even with smart accelerating ideas, is a very "pricing-intensive" mechanism (i.e., it requires many calls to the pricing solver).

3.6.2 ▪ Route Enumeration

Baldacci, Christofides, and Mingozzi [3] use a route enumeration–based approach (already used in Baldacci, Bodin, and Mingozzi [2]) in order to close the duality gap after the root node. The key observation is that a route $r \in \Omega$ can only be part of a solution that improves the best known upper bound if (i) its reduced cost (the sum of its edge reduced costs with respect to the current dual solution) is smaller than the gap, and (ii) there is no other route visiting the same set of clients with smaller cost (with respect to the original edge costs). The enumeration of those routes may be performed by a label setting algorithm, similar to those used to price elementary routes, producing a set $R \subseteq \Omega$. A general MIP solver is then used to solve the formulation VRP4 restricted to the set R, using the known upper bound as a cutoff. If the restricted model is infeasible, it is proved that the solution that provided the upper bound is optimal. Otherwise, the optimal solution of the restricted model is the optimal solution of the complete problem.

The efficiency of this approach is related to the cardinality of the set R. In fact, if $|R|$ is too large, the label setting enumerating algorithm will run out of memory. Even assuming that it is possible to enumerate R, there is still the question of solving the restricted VRP4 model. The cardinality of R depends crucially on the duality gap, but there is a second important factor: again, the average number of customers per route. If it is small, say, $n/|K| \leq 8$, then R is not likely to be large, even with a significant gap. It is typical in those cases to generate sets with no more than a few tenths of thousands routes,

the resulting restricted models being solved in minutes or even seconds. The speedup of the enumeration approach with respect to the traditional branching can be large. On the other hand, instances with many customers per route can be more problematic. Successful enumeration on those cases may require small gaps. Baldacci, Mingozzi, and Roberti [4] enhanced the enumeration procedure by considering two dual solutions π_1 and π_2, with associated lower bounds LB_1 and LB_2. A route r cannot belong to any improving solution either if $LB_1 + \bar{c}_1(r) \geq UB$ or if $LB_2 + \bar{c}_2(r) \geq UB$, where UB is the best known upper bound and $\bar{c}_1(r)$ and $\bar{c}_2(r)$ are the route reduced costs with respect to π_1 and π_2, respectively. In their algorithm, the second dual solution corresponds to the final solution at the root node, and the first dual solution is obtained at an earlier stage of the algorithm, so $LB_1 \leq LB_2$. In fact, after π_1 is obtained, the pricing of elementary routes used in their column and cut generation algorithm starts to test the condition $LB_1 + \bar{c}_1(r) < UB$. This restriction on the set of routes may improve the final bound LB_2.

Contardo [12] introduced a different strategy to better profit from the route enumeration. From time to time, it is tested whether the current solution would lead to a set R_1 with less than 5 million routes, still too large to allow the efficient resolution of the restricted VRP4 model by a MIP solver. Instead, the set R_1 is stored in a pool, and the column and cut generation proceeds. However, from this point, the pricing is not solved by a label setting algorithm anymore; it starts to be performed by a straightforward inspection of the routes in the pool. As a result, the separation of non-robust cuts, which was being done in a very controlled way, can now be aggressive. After each improvement of the current lower bound, reduced cost fixing is performed in the pool, decreasing its size. In most instances from the literature, the column and cut generation algorithm terminates with zero gap. If this is not the case, the final set of routes remaining in the pool defines the set R and the corresponding restricted VRP4 model is given to the MIP solver. For example, for solving instance M-n151-k12, a relaxation giving a lower bound of 1009.0 generates a set R_1 with about 4 million routes. In the end of the root node, the lower bound is increased to 1012.5 and the final set R only has about 13,000 routes. This strategy is also used in Contardo and Martinelli [14].

3.6.3 ▪ Hybridizing Branching and Route Enumeration

It is clear that route enumeration is an important element in some of the best performing algorithms for the CVRP, being capable of drastically reducing the running times of some instances. Nevertheless, it is disturbing to consider that route enumeration, no matter how cleverly done, is an inherently exponential space procedure (it would remain exponential even if P=NP) that is bound to fail on larger/harder instances.

However, one does not need to take a radical stance on completely avoiding branching. The hybrid strategy used in Pessoa, Poggi de Aragão, and Uchoa [33] and Pessoa, Uchoa, and Poggi de Aragão [35] performs route enumeration after solving each node. If the limit of 80,000 routes is reached, the enumeration is aborted and the BCP proceeds by traditional branching. Of course, since deeper nodes will have smaller gaps, at some point the enumeration will work. The overall effect may be a substantially reduced enumeration tree. For example, where the traditional branching would need to reach depth 10, that hybrid strategy would not go beyond depth 5.

The BCP in Pecin et al. [32] implements a more sophisticated hybrid strategy. Route enumeration (in Contardo's style) is tried in each node v, allowing quite large sets R_1, in some cases with up to 50 million routes:

- If the enumeration succeeds, node v continues to be solved using the pricing by inspection in the pool of routes. An unlimited number of SRCs and Clique Cuts

(not used before the enumeration) are separated. Fixing by reduced costs eliminates routes from the pool. If the final set R has less than 20,000 routes, the node is finished by an MIP solver. However, there are situations where the final set R is still too large. When this happens, the BCP itself performs a branching. Each child node of v receives a copy of R; the pricing in those nodes will continue to be done by inspection.

- If the enumeration fails, the node performs an aggressive hierarchical strong branching. Actually, the branching effort is dimensioned according to the expected size of the subtree rooted in v.

3.7 ▪ Overview of Computational Results

In order to provide experimental information on the topics discussed in this chapter, we report results over the standard classes of instances used for testing exact methods for the CVRP. Classes A, B, and P were proposed in Augerat et al. [1]. Classes E, F, and M were proposed by Christofides and Eilon [8], by Fisher [18], and by Christofides, Mingozzi, and Toth [9], respectively. All those instances can currently be found at http://branchandcut.org/VRP/data.

We start this section with an analysis of the bounds that are obtained with formulation VRP4 without the addition of extra cuts. Tables 3.1 and 3.2 present the computing times and the percent gap from the proved optimal over a selected set of instances which includes those that, today, can be regarded as the most relevant when analyzing exact approaches for the CVRP. The following variants are considered: (i) q-routes with k-cycle elimination, for $k = 2,3,4,5$, using the code from Fukasawa et al. [19]; and (ii) ng-routes with ng-sets of size 8, 16, 32, and 64, using the code from Martinelli, Pecin, and Poggi [30]. This last code is an unidirectional forward label setting algorithm that uses a kind of DSSR; the ng-sets are increased dynamically. Times correspond to runs on processor Intel Core 2 Duo E7400 2.8GHz. A time limit of 2 hours was set. The rows **Avg. values** are averages, always excluding instances F-n135-k7 and M-n121-k7, where some variants exceeded the time limit. For the columns corresponding to gaps, they are ordinary arithmetic averages. However, for the columns corresponding to times, we report geometrical averages (otherwise a few larger instances would determine the results). Some comments follow:

- Pricing routes without 2-cycles or 3-cycles is fast. This is expected from the theory; there is a proven upper bound of at most 2 and 6 labels per bucket, respectively. On the other hand, the resulting bounds are not close to those that could be obtained by pricing elementary routes. This suggests using k-cycle elimination for larger values of k. However, the approach does not scale well. Elimination of 5-cycles makes the pricing much slower and still does not reach the elementary bounds.

- The ng-route approach performed much better. Using ng-sets of size 8 already gives bounds similar to those obtained by 5-cycle elimination, taking a time comparable to 4-cycle elimination. The relative strength of ng-8 is somewhat counterintuitive, since it is not difficult to construct cycles with five (or even less) reasonably short edges that would not be forbidden by a memory of eight neighbors. Anyway, it seems that the ng-route approach (at least with the new DSSR implementation in [30]) scales very well. Increasing the size of the ng-sets to 16, 32, and 64 does not make the algorithm much slower. The bounds obtained by ng-64 over those instances are identical to the bounds obtained by elementary routes. In fact,

Bounds for the set partitioning formulation without extra cuts

Table 3.1. *Set partitioning formulation - no cuts - k-cycle elimination.*

Instance			2-cycle		3-cycle		4-cycle		5-cycle	
Name	n/k	OPT	T(s)	gap	T(s)	gap	T(s)	gap	T(s)	gap
A-n62-k8	7.6	1288	1.68	5.07	2.90	3.28	6.08	2.97	26.14	2.80
A-n80-k10	7.9	1763	4.37	2.88	4.52	2.28	12.16	2.02	30.34	1.95
B-n50-k8	6.1	1312	0.69	7.20	0.98	6.16	2.87	3.66	13.30	3.47
B-n78-k10	7.7	1221	1.62	7.90	3.05	6.21	4.36	6.00	32.66	5.63
E-n76-k7	10.7	682	3.95	2.74	4.37	2.71	11.88	2.67	57.72	2.66
E-n76-k8	9.4	735	2.71	2.49	3.72	2.40	7.47	2.34	36.26	2.34
E-n76-k10	7.5	830	1.65	2.24	2.04	2.22	3.92	2.20	15.70	2.20
E-n76-k14	5.4	1021	0.64	2.10	0.68	2.03	1.77	1.85	4.75	1.84
E-n101-k8	12.5	815	8.43	3.51	18.69	3.23	20.83	3.15	70.82	3.12
E-n101-k14	7.1	1067	2.39	2.05	3.25	1.88	6.00	1.86	21.76	1.86
F-n135-k7	19.1	1162	1400	6.04	3119	3.82	4757	2.64	-	-
M-n121-k7	17.1	1034	54.07	2.03	66.93	1.35	108.8	1.03	360.3	0.89
M-n151-k12	12.5	1015	32.92	2.32	47.55	2.05	74.93	2.01	194.6	1.94
M-n200-k16	12.5	1274	138.3	2.63	139.5	1.98	230.2	1.87	424.2	1.84
M-n200-k17	11.8	1275	76.62	2.50	94.46	1.87	159.5	1.78	420.4	1.75
P-n70-k10	6.9	827	0.81	2.26	1.25	2.15	2.68	2.05	6.30	2.01
Avg. values:			4.20	3.42	5.74	2.89	11.26	2.60	38.89	2.53

Table 3.2. *Set partitioning formulation - no cuts - ng-routes (ng-64 is equivalent to elementary route pricing in all those instances).*

Instance	ng-8		ng-16		ng-32		ng-64	
Name	T(s)	gap	T(s)	gap	T(s)	gap	T(s)	gap
A-n62-k8	3.26	2.93	3.5	2.58	4.24	2.58	5.09	2.58
A-n80-k10	6.03	1.88	6.12	1.79	7.57	1.78	10.75	1.78
B-n50-k8	1.50	3.47	1.56	3.46	1.82	3.46	1.85	3.46
B-n78-k10	5.50	4.44	5.57	4.38	7.07	4.38	9.24	4.38
E-n76-k7	9.86	2.61	10.95	2.52	11.20	2.41	15.17	2.41
E-n76-k8	6.66	2.34	7.72	2.21	7.25	2.21	10.14	2.21
E-n76-k10	3.80	2.20	4.63	2.18	4.91	2.11	5.56	2.11
E-n76-k14	2.65	1.88	2.56	1.79	2.47	1.79	3.03	1.79
E-n101-k8	23.90	3.14	23.31	2.98	27.57	2.95	35.14	2.95
E-n101-k14	8.29	1.86	9.01	1.74	9.10	1.55	10.39	1.55
F-n135-k7	6462	2.34	-	-	-	-	-	-
M-n121-k7	74.18	0.74	564.1	0.46	-	-	-	-
M-n151-k12	72.70	1.90	83.36	1.81	91.89	1.75	111.1	1.73
M-n200-k16	158.9	1.87	164.7	1.82	206.3	1.78	237.6	1.72
M-n200-k17	201.5	1.76	156.7	1.74	194.9	1.69	234.3	1.65
P-n70-k10	2.82	2.14	3.04	1.98	2.98	1.94	3.67	1.94
Avg. values:	10.24	2.46	10.70	2.35	11.89	2.31	14.60	2.30

setting sufficiently large ng-sets seems to be a very practical way of pricing elementary routes.

- The last comments are not valid for the instances with many customers per route. For example, it is not practical to use large ng-sets on instances F-n135-k7 ($n/|K| = 19.1$) and M-n121-k7 ($n/|K| = 17.1$).

As mentioned before, the bounds obtained by using VRP4, even with elementary routes, are not good enough to be the basis of an efficient CVRP algorithm. Tables 3.3 and 3.4 present times and bounds for VRP4 with the addition of rounded RCCs and comb cuts separated using the heuristics in the CVRPSEP library (see Lysgaard [28]). Those robust cuts typically eliminate more than half of the gaps, which are now small enough to

Bounds for the set partitioning formulation with robust cuts

Table 3.3. *Set partitioning formulation - RCCs + comb cuts - k-cycle elimination.*

Instance			2-cycle		3-cycle		4-cycle		5-cycle	
Name	n/k	OPT	T(s)	gap	T(s)	gap	T(s)	gap	T(s)	gap
A-n62-k8	7.6	1288	11.13	1.10	11.56	0.60	24.69	0.55	111.8	0.55
A-n80-k10	7.9	1763	19.27	0.75	16.68	0.52	47.79	0.43	157.8	0.42
B-n50-k8	6.1	1312	4.52	1.54	6.13	1.30	8.61	0.76	37.32	0.63
B-n78-k10	7.7	1221	17.94	0.69	16.71	0.56	26.02	0.43	171.4	0.35
E-n76-k7	10.7	682	13.52	1.77	17.68	1.73	44.87	1.69	500.2	1.68
E-n76-k8	9.4	735	17.08	1.21	21.96	1.16	39.06	1.15	263.2	1.13
E-n76-k10	7.5	830	7.20	1.56	6.43	1.59	14.00	1.53	104.7	1.52
E-n76-k14	5.4	1021	2.64	1.59	3.46	1.42	6.49	1.33	20.48	1.32
E-n101-k8	12.5	815	53.74	1.38	81.18	1.35	142.1	1.33	827.9	1.29
E-n101-k14	7.1	1067	7.12	1.44	13.19	1.26	24.50	1.24	112.3	1.23
F-n135-k7	19.1	1162	5203	0.21	-	-	-	-	-	-
M-n121-k7	17.1	1034	535.1	0.32	483.0	0.25	935.6	0.21	3302	0.21
M-n151-k12	12.5	1015	125.0	1.72	180.1	1.57	357.7	1.52	1878	1.45
M-n200-k16	12.5	1274	489.0	2.08	592.4	1.73	860.7	1.70	3369	1.67
M-n200-k17	11.8	1275	293.7	1.97	359.2	1.62	559.0	1.58	2358	1.53
P-n70-k10	6.9	827	3.76	1.67	4.26	1.53	11.55	1.48	83.31	1.44
Avg. values:			20.26	1.39	24.61	1.21	47.43	1.13	249.8	1.09

Table 3.4. *Set partitioning formulation - RCCs + comb cuts - ng-routes (ng-64 is equivalent to elementary route pricing in all those instances).*

Instance	ng-8		ng-16		ng-32		ng-64	
Name	T(s)	gap	T(s)	gap	T(s)	gap	T(s)	gap
A-n62-k8	6.74	0.56	7.63	0.46	9.41	0.46	8.21	0.47
A-n80-k10	10.42	0.37	10.24	0.36	10.76	0.35	15.78	0.34
B-n50-k8	3.01	0.65	2.38	0.64	2.67	0.63	3.61	0.63
B-n78-k10	9.54	0.32	9.92	0.30	12.82	0.31	16.01	0.30
E-n76-k7	14.18	1.60	15.05	1.57	15.51	1.48	21.33	1.48
E-n76-k8	12.59	1.15	12.69	1.08	14.53	1.07	16.29	1.05
E-n76-k10	5.22	1.53	6.36	1.52	6.55	1.45	7.61	1.44
E-n76-k14	3.23	1.38	3.73	1.32	3.37	1.32	3.63	1.32
E-n101-k8	47.37	1.29	50.39	1.24	49.31	1.22	61.53	1.23
E-n101-k14	10.01	1.25	9.35	1.21	11.71	1.12	12.30	1.12
F-n135-k7	-	-	-	-	-	-	-	-
M-n121-k7	108.2	0.15	1483	0.13	-	-	-	-
M-n151-k12	107.7	1.44	96.89	1.32	111.7	1.26	131.8	1.23
M-n200-k16	189.5	1.69	189.6	1.66	221.9	1.62	275.2	1.54
M-n200-k17	176.2	1.56	196.6	1.54	209.4	1.51	264.7	1.47
P-n70-k10	4.00	1.53	4.54	1.41	4.35	1.42	6.29	1.40
Avg. values:	15.38	1.10	15.88	1.05	17.35	1.02	21.02	1.01

provide efficient algorithms. Again, results with k-cycle elimination were obtained by the code from Fukasawa et al. [19], and results with ng-paths came from Martinelli, Pecin, and Poggi [30]. While the first algorithm only separates a round of cuts after column generation with exact pricing converges, the more recent algorithm found advantage in trying to separate cuts as soon as the heuristic pricing fails, reducing the number of calls to the more expensive exact pricing. This provides a partial explanation for the fact that the times for the 2-cycle elimination variant are comparable with the times for the much stronger ng-64 variant. Anyway, the previous comments for the formulation VRP4 without cuts are still valid. On most instances, ng-8 obtains bounds comparable to 5-cycle elimination in less time; ng-16, ng-32, and ng-64 are not much slower than ng-8. Moreover, ng-64 is still equivalent to elementary route pricing.

Bounds for the set partitioning formulation with SRCs

Table 3.5. *Set partitioning formulation - RCCs + lm-SRCs - ng-8.*

Instance			1SRC and 3SRCs						+4,5SRCs	
Name	n/k	OPT	T(s)	gap	1SRC	D1	3SRC	D3	T(s)	gap
A-n62-k8	7.6	1288	14.7	0.15	17	8.1	86	10.8	25.6	0.05
A-n80-k10	7.9	1763	16.0	0.09	23	7.3	109	10.3	36.6	0.01
B-n50-k8	6.1	1312	13.2	0.21	22	5.8	53	8.6	31.0	0.17
B-n78-k10	7.7	1221	6.9	0.00	3	5.7	62	6.9	7.5	0.00
E-n76-k7	10.7	682	102.9	0.16	33	11.4	193	13.5	224.0	0.05
E-n76-k8	9.4	735	38.5	0.12	39	9.9	169	11.3	45.3	0.04
E-n76-k10	7.5	830	22.9	0.43	19	5.8	143	8.8	68.6	0.19
E-n76-k14	5.4	1021	6.0	0.62	14	4.5	93	6.1	15.2	0.46
E-n101-k8	12.5	815	155.8	0.00	29	11.7	205	11.2	147.9	0.00
E-n101-k14	7.1	1067	40.1	0.35	35	6.7	175	7.9	120.8	0.27
F-n135-k7	19.1	1162	-	-	-	-	-	-	-	-
M-n121-k7	17.1	1034	82.9	0.05	25	15.4	47	26.0	228.2	0.05
M-n151-k12	12.5	1015	636.6	0.28	78	11.0	340	12.5	996.3	0.14
M-n200-k16	12.5	1274	2791	0.56	122	9.5	546	13.0	8400	0.42
M-n200-k17	11.8	1275	2405	0.45	92	9.6	537	13.0	7903	0.31
P-n70-k10	6.9	827	12.6	0.47	17	6.0	135	8.3	39.1	0.28
Avg. values:			57.1	0.28					118.3	0.17

Table 3.5 is intended to show the large supplementary gap reductions that can be obtained by non-robust SRCs. The results were obtained with the algorithm in Pecin et al. [32] on a single core of an i7-2620M 3.3GHz processor, pricing ng-routes with ng-sets of size 8. The first set of columns corresponds to the separation of RCCs, lm-1SRCs, and lm-3SRCs. As the separation is performed until convergence, the final lower bounds are the same that would be obtained by pricing elementary routes (the lm-1SRCs remove all cycles) and separating RCCs plus traditional 3SRCs. The table also shows additional information for that variant of the algorithm: the number of active lm-1SRCs, their densities (D1) (the average size of the corresponding M sets), the number of active lm-3SRCs, and also their densities (D3). Comparing with Table 3.4, it can be verified that the 3SRCs alone reduced the average gap from 1.01 to 0.28. The relatively small densities of the lm-SRCs explain how the pricing remains tractable even on the instances with 200 vertices. The last two columns in Table 3.5 show results for a variant that also separates lm-4SRCs, lm-5,1SRCs, and lm-5,2SRCs. The average gap decreased to 0.17. We remark that it is not a good BCP strategy to separate lm-SRCs until convergence; it is possible to obtain slightly larger gaps in a fraction of the times reported in Table 3.5.

The final set of results shown in this chapter is aimed at showing the evolution of the recent exact approaches for the CVRP. Tables 3.6, 3.7, and 3.8 summarize their performances over the set of 81 instances, divided by class, used in Fukasawa et al. [19]. As usual in the recent literature where similar tables appear, classes E and M are grouped together. Columns **Opt** indicate the number of instances solved to optimality. Columns **Gap** and **Time** are the average gap in the root node and average times in seconds (over the indicated machines), computed only over the solved instances. Some comments and additional information about those results follow:

- **LLE04** refers to the Branch-and-Cut in Lysgaard, Letchford, and Eglese [29]. That code represents the culmination of a sequence of Branch-and-Cut algorithms (started by Augerat et al. [1]), with increasingly better separation of valid cuts for the formulation VRP2. Analyzing its results in more detail, it can be noticed that the algorithm had particular difficulties solving instances with fewer customers per

Overall results of the more recent exact algorithms

Table 3.6. *Summary of results: LLE*04*, FLL*+06*, BCM*08*.*

		LLE04			FLL+06			BCM08		
Class	NP	Opt	Gap	Time	Opt	Gap	Time	Opt	Gap	Time
A	22	15	2.06	6638	22	0.81	1961	22	0.20	118
B	20	19	0.61	8178	20	0.47	4763	20	0.16	417
E-M	12	3	2.10	39592	9	1.19	126987	8	0.69	1025
F	3	3	0.06	1016	3	0.14	2398			
P	24	16	2.26	11219	24	0.76	2892	22	0.28	187
Total	81	56			78			72		
Machine		Intel Celeron 700MHz			Pentium 4 2.4GHz			Pentium 4 2.6GHz		

Table 3.7. *Summary of results: BMR*11*, Con*12*, CM*14*.*

		BMR11			Con12			CM14		
Class	NP	Opt	Gap	Time	Opt	Gap	Time	Opt	Gap	Time
A	22	22	0.13	30	22	0.07	59	22	0.09	59
B	20	20	0.06	67	20	0.05	89	20	0.08	34
E-M	12	9	0.49	303	10	0.30	2807	10	0.27	1548
F	3	2	0.11	164	2	0.06	3	3	0.03	27722
P	24	24	0.23	85	24	0.13	43	17	0.18	240
Total	81	77			78			79		
Machine		Xeon X7350 2.93GHz			Xeon E5462 2.8GHz			Xeon E5462 2.8GHz		

Table 3.8. *Summary of results: Rop*12*, PPPU*14*.*

		Rop12			PPPU14		
Class	NP	Opt	Gap	Time	Opt	Gap	Time
A	22	22	0.57	53	22	0.03	5.6
B	20	20	0.25	208	20	0.04	6.2
E-M	12	10	0.96	44295	12	0.19	3669
F	3	3	0.25	2163	3	0.00	3679
P	24	24	0.69	280	24	0.07	33
Total	81	79			81		
Machine		Core i7-2620M 2.7GHz			Core i7-2620M 3.3GHz		

vehicle; their root gaps were too large. For example, instances E-n76-k7 and E-n76-k8 could be solved, but E-n76-k10 and E-n76-k14 could not. On the other hand, the Branch-and-Cut had little trouble solving instance F-n135-k7 (already solved in [1]).

- **FLL+06** refers to the BCP in Fukasawa et al. [19]. As shown in Table 3.6, the combination of column and cut generation reduced a lot the root gaps with respect to those by LLE04. The improvement was more significant precisely in the instances with fewer customers per vehicle. Overall, only the three larger instances in the M series could not be solved to optimality. It should be remarked that in some instances (like F-n135-k7), when the algorithm realizes that the column generation in the root node is too slow, it switches to a Branch-and-Cut similar to LLE04.

- **BCM08** refers to the column and cut generation algorithm in Baldacci, Christofides, and Mingozzi [3]. It can be seen that the introduction of SCCs and Clique Cuts (the latter were particularly effective) led to significantly reduced gaps with respect to those in FLL+06. Moreover, the route enumeration dramatically reduced the overall running times for many instances. Those effects were especially acute on the

instances with fewer customers per vehicle. For example, on instance E-n101-k14, the root node of FLL+06 yielded the lower bound of 1053.8, and the BCP completed after exploring 5848 nodes, taking 116,284 seconds. In the same instance, BCM08 obtained a bound of 1064.3 and proceeded to enumerate 76,756 routes (using an upper bound of 1071). The resulting restricted VRP4 model is then sent to a general MIP solver. The overall solving time was only 1230 seconds. On the other hand, the more recent algorithm had difficulties solving some instances with many customers per route. For example, instance E-n101-k8 could not be solved.

- **BMR11** refers to the column and cut generation algorithm in Baldacci, Mingozzi, and Roberti [4]. It can be seen that newly introduced elements improved the gaps and the running times significantly with respect to BCM08. More importantly, the algorithm became much more stable, being able to quickly solve some instances with many customers per route. For example, instance E-n101-k8 took 579 seconds. The only previously solved instance that could not be solved was F-n135-k7.

- **Con12** refers to the column and cut generation algorithm in Contardo [12]. The algorithm follows the path paved by BCM08 and BMR11, combining non-robust cut pricing and route enumeration. However, Con12 sticks to an easier base pricing problem, 2-cycle elimination, while non-robust cuts (including those to enforce partial elementarity) are carefully added. It seems that this approach leaves more room for introducing a larger number of 3SRCs; those cuts are particularly effective on reducing the gaps. The solution of instance M-n151-k12 in 19,699 seconds is an indication of its strength.

- **CM14** refers to the column and cut generation algorithm in Contardo and Martinelli [14]. The algorithm contains a number of improvements over Con12; ng-routes are priced instead of q-routes with 2-cycle elimination. Instance M-n151-k12 could be solved in less than 3 hours. Also, an improved pricing algorithm implementation allowed solving instance F-n135-k7 by column generation for the first time. It took just over 83,000 seconds. It should be remarked that the algorithm was run only on 17 of the 24 instances of series P. As the seven ignored instances are very small (up to 23 vertices) and very easy, we assumed zero gaps and zero running times for computing statistics in Table 3.7.

- **Rop12** refers to the BCP algorithm in Røpke [39]. This algorithm is closer to the BCP in Fukasawa et al. [19]; its root gaps (ng-10 pricing + CVRPSep cuts) are significantly worse than those by BCM08, BMR11, Con12, and CM14. On the other hand, the robust approach allows the fast solution of each node, which combined with an aggressive and effective strong branching yielded results that are comparable with the results of those algorithms. The run that solved instance M-n151-k12 used ng-15 pricing and took 5268 nodes and 417,146 seconds to be finished. The solution of instance F-n135-k7 also required an special parameterization; instead of ng-routes, faster 2-cycle elimination was used.

- **PPPU14** refers to the BCP algorithm in Pecin et al. [32]. This algorithm uses ng-8 pricing and separates RCCs and lm-SRCs (1SRCs, 3SRCs, 4SRCs, and 5SRCs). After enumeration it can also separate Clique Cuts. Its algorithmic engineering combines elements found in many previous works. Instance M-n151-k12 was solved in 212 seconds. Instance M-n200-k17 was solved in 3581 seconds, while M-n200-k16 took 39,869 seconds. The larger time spent in M-n200-k16 is explained by the relatively poor quality of the best known upper bound for that instance, 4 units

above the optimal 1274. Instance F-n135-k7 was solved by the BCP in 5405 seconds. Four larger instances, ranging from 240 to 360 customers, from the set proposed in Golden et al. [21] were solved too.

Currently, LLE04 and PPPU14 are the non-dominated exact CVRP algorithms available, in the sense that there are instances from the literature that are better solved by one of them. Unhappily, the classical benchmark used in the CVRP has only three instances in the range from 150 to 200 vertices. However, extensive experiments with a new benchmark set (Uchoa et al. [42]) support the statement that current algorithms can indeed solve instances with up to 200 vertices in a consistent way. Furthermore, many instances, those with characteristics that are more favorable to BCP algorithms (like an $n/|K|$ ratio smaller than 8), are likely to be solved with up to 300 vertices.

3.8 ▪ Conclusions and Future Research Directions

The progress made in the last decade on exact CVRP algorithms was very expressive. Measuring it by the size where instances can be consistently solved, we observe an increase from 50 to 200 customers between Lysgaard, Letchford, and Eglese [29] and Pecin et al. [32]. The last algorithm was the result of a deliberate effort of testing, improving, and combining ideas proposed by several authors. Some of the newer elements found in CVRP algorithms are likely to be useful on algorithms for solving other VRP variants. For example, the lm-SRCs could be used in any variant that can be modeled by the set partitioning formulation VRP4.

Bibliography

[1] 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.

[2] R. Baldacci, L. Bodin, and A. Mingozzi, *The multiple disposal facilities and multiple inventory locations rollon–rolloff vehicle routing problem*, Computers & Operations Research, 33 (2006), pp. 2667–2702.

[3] R. Baldacci, N. Christofides, and A. Mingozzi, *An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts*, Mathematical Programming, 115 (2008), pp. 351–385.

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

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

[6] N. Boland, J. Dethridge, and I. Dumitrescu, *Accelerated label setting algorithms for the elementary resource constrained shortest path problem*, Operations Research Letters, 34 (2006), pp. 58–68.

[7] A. Chabrier, *Vehicle routing problem with elementary shortest path based column generation*, Computers & Operations Research, 33 (2006), pp. 2972–2990.

[8] N. CHRISTOFIDES AND S. EILON, *An algorithm for the vehicle-dispatching problem*, Operational Research Quarterly, 20 (1969), pp. 309–318.

[9] N. CHRISTOFIDES, A. MINGOZZI, AND P. TOTH, *The vehicle routing problem*, in Combinatorial Optimization, N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, eds., vol. 1, Wiley Interscience, Chichester, 1979, pp. 315–338.

[10] ——, *Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations*, Mathematical Programming, 20 (1981), pp. 255–282.

[11] ——, *State-space relaxation procedures for the computation of bounds to routing problems*, Networks, 11 (1981), pp. 145–164.

[12] C. CONTARDO, *A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints*, Technical Report, Archipel-UQAM 5078, Université du Québec à Montréal, Canada, 2012.

[13] C. CONTARDO, J.-F. CORDEAU, AND B. GENDRON, *A branch-and-cut-and-price algorithm for the capacitated location-routing problem*, Technical report, CIRRELT-2011-44, Université de Montréal, Canada, 2011.

[14] C. CONTARDO AND R. MARTINELLI, *A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints*, Discrete Optimization, 12 (2014), pp. 129–146.

[15] J. DESROSIERS, F. SOUMIS, AND M. DESROCHERS, *Routing with time windows by column generation*, Networks, 14 (1984), pp. 545–565.

[16] O. DU MERLE, D. VILLENEUVE, J. DESROSIERS, AND P. HANSEN, *Stabilized column generation*, Discrete Mathematics, 194 (1999), pp. 229–237.

[17] D. FEILLET, P. DEJAX, M. GENDREAU, AND C. GUEGUEN, *An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems*, Networks, 44 (2004), pp. 216–229.

[18] M. L. FISHER, *Optimal solution of vehicle routing problems using minimum k-trees*, Operations Research, 42 (1994), pp. 626–642.

[19] 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 (2006), pp. 491–511.

[20] M. T. GODINHO, L. GOUVEIA, T. MAGNANTI, P. PESNEAU, AND J. PIRES, *On time-dependent models for unit demand vehicle routing problems*, in International Network Optimization Conference (INOC), 2007.

[21] B. GOLDEN, E. WASIL, J. KELLY, AND I.-M. CHAO, *The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results*, in Fleet Management and Logistics, Springer, Berlin, 1998, pp. 33–56.

[22] E. HOSHINO AND C. DE SOUZA, *Column generation algorithms for the capacitated m-ring-star problem*, in Computing and Combinatorics, Springer, Berlin, 2008, pp. 631–641.

[23] S. IRNICH, G. DESAULNIERS, J. DESROSIERS, AND A. HADJAR, *Path-reduced costs for eliminating arcs in routing and scheduling*, INFORMS Journal on Computing, 22 (2010), pp. 297–313.

[24] 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 (2006), pp. 391–406.

[25] M. JEPSEN, B. PETERSEN, S. SPOORENDONK, AND D. PISINGER, *Subset-row inequalities applied to the vehicle-routing problem with time windows*, Operations Research, 56 (2008), pp. 497–511.

[26] G. LAPORTE AND Y. NOBERT, *A branch and bound algorithm for the capacitated vehicle routing problem*, OR Spectrum, 5 (1983), pp. 77–85.

[27] A. LETCHFORD AND J. J. SALAZAR GONZÁLEZ, *Projection results for vehicle routing*, Mathematical Programming, 105 (2006), pp. 251–274.

[28] J. LYSGAARD, *CVRPSEP: A package of separation routines for the capacitated vehicle routing problem*, Aarhus School of Business, Department of Management Science and Logistics, Aarhus, Denmark, 2003.

[29] J. LYSGAARD, A. LETCHFORD, AND R. EGLESE, *A new branch-and-cut algorithm for the capacitated vehicle routing problem*, Mathematical Programming, 100 (2004), pp. 423–445.

[30] R. MARTINELLI, D. PECIN, AND M. POGGI, *Efficient restricted non-elementary route pricing for routing problems*, Technical Report MCC 11/13, PUC-Rio, Departamento de Informática, Rio de Janeiro, Brasil, 2013.

[31] D. NADDEF AND G. RINALDI, *Branch-and-cut algorithms for the capacitated VRP*, in The Vehicle Routing Problem, P. Toth and D. Vigo, eds., SIAM, 2002, ch. 3, pp. 53–84.

[32] D. PECIN, A. PESSOA, M. POGGI, AND E. UCHOA, *Improved branch-and-cut-and-price for capacitated vehicle routing*, in Integer Programming and Combinatorial Optimization, vol. 8494 of Lecture Notes in Computer Science, Springer, Berlin, 2014, pp. 393–403.

[33] A. PESSOA, M. POGGI DE ARAGÃO, AND E. UCHOA, *Robust branch-cut-and-price algorithms for vehicle routing problems*, in the Vehicle Routing Problem: Latest Advances and New Challenges, B. Golden, S. Raghavan, and E. Wasil, eds., Springer, New York, 2008, pp. 297–325.

[34] A. PESSOA, R. SADYKOV, E. UCHOA, AND F. VANDERBECK, *In-out separation and column generation stabilization by dual price smoothing*, in Symposium on Experimental Algorithms, Springer, Berlin, 2013, pp. 354–365.

[35] A. PESSOA, E. UCHOA, AND M. POGGI DE ARAGÃO, *A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem*, Networks, 54 (2009), pp. 167–177.

[36] M. POGGI DE ARAGÃO AND E. UCHOA, *Integer program reformulation for robust branch-and-cut-and-price*, in Annals of Mathematical Programming in Rio, L. Wolsey, ed., Búzios, Brazil, 2003, pp. 56–61.

[37] G. RIGHINI AND M. SALANI, *Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints*, Discrete Optimization, 3 (2006), pp. 255–273.

[38] ——, *New dynamic programming algorithms for the resource constrained elementary shortest path problem*, Networks, 51 (2008), pp. 155–170.

[39] S. RØPKE, *Branching decisions in branch-and-cut-and-price algorithms for vehicle routing problems*, Presentation in Column Generation 2012, 2012.

[40] A. SUBRAMANIAN, E. UCHOA, A. PESSOA, AND L. S. OCHI, *Branch-cut-and-price for the vehicle routing problem with simultaneous pickup and delivery*, Optimization Letters, 7 (2013), pp. 1569–1581.

[41] E. UCHOA, *Cuts over extended formulations by flow discretization*, in Progress in Combinatorial Optimization, A. Mahjoub, ed., ISTE-Wiley, London, 2011, pp. 255–282.

[42] E. UCHOA, D. PECIN, A. PESSOA, M. POGGI, A. SUBRAMANIAN, AND T. VIDAL, *New benchmark instances for the capacitated vehicle routing problem*, Presentation in Route 2014, 2014.