Les Cahiers du GERAD ISSN: 0711–2440

The Quadratic Capacitated Vehicle Routing Problem

R. Martinelli
C. Contardo

G–2013–74

October 2013

The Quadratic Capacitated Vehicle Routing Problem

Rafael Martinelli

Departmento de Computação
Universidade Federal de Ouro Preto
Ouro Preto/MG, Brasil 35400-000

rafael.martinelli@iceb.ufop.br

Claudio Contardo

GERAD & Département de management et technologie
ESG UQÀM
Montréal (Québec) Canada, H2X 3X2

claudio.contardo@gerad.ca

October 2013

Les Cahiers du GERAD

G–2013–74

Abstract: In this article we introduce the Quadratic Capacitated Vehicle Routing Problem (QCVRP), a combinatorial optimization problem that arises in practical applications in logistics and transportation. The QCVRP extends two other known problems, the Capacitated Vehicle Routing Problem and the Quadratic Symmetric Traveling Salesman Problem, the former by considering a modified cost matrix in which traveling costs are now associated to pairs of two consecutive edges, and the latter by introducing customer demands and vehicle capacities. We present a three-index vehicle-flow formulation for the problem in which variables represent q-edges (pairs of consecutive edges), and strengthen it with several classes of valid inequalities. We present efficient separation routines for the inequalities used in this paper and derive an exact solver based on the branch-and-cut paradigm. We also present an efficient heuristic for finding feasible solutions of the QCVRP quickly, using some state-of-the-art sub-routines from the vehicle routing literature especially adapted for this problem. Parallel implementations of the two algorithms are proposed that take advantage of state-of-the-art multiple-core architectures. We conduct extensive computational experiments to assess the efficiency of the modeling and solution approaches presented, as well as of their parallel implementations. Small- and medium-size problems can be solved to optimality while tight lower and upper bounds are computed for larger – more difficult – problems.

Key Words: Quadratic capacitated vehicle routing problem, angle capacitated vehicle routing problem, capacitated vehicle routing problem with reload costs, branch-and-cut, hybrid metaheuristics.

Acknowledgments: We thank F.D. Fomeni and A.N. Letchford for kindly providing us with the revised manuscript and the C code of their dynamic programming heuristic for the 0-1 QKP. We thank also A. Fischer and C. Helmberg for sending us the instances used in their experiments with the QSTSP. We are also thankful to Compute Canada for having access to its resources in high performance computing. C. Contardo has been partially funded by the ESG UQÀM under the *PAFARC* program and the *Natural Sciences and Engineering Research Council of Canada* (NSERC) under the Discovery Grant program. These supports are gratefully acknowledged.

1. Introduction

This paper introduces the Quadratic Capacitated Vehicle Routing Problem (QCVRP), an extension of two important problems in combinatorial optimization and logistics: the Capacitated Vehicle Routing Problem (CVRP) and the Quadratic Symmetric Traveling Salesman Problem (QSTSP). In the QCVRP, a fleet of K homogeneous vehicles must be routed from a single depot to visit exactly once all customers from a predefined set of customers $V^+ = \{1, \dots, n\}$. For notational convenience, the depot node is labeled as the node 0, and we denote $V = V^+ \cup \{0\}$. With each vehicle is associated a capacity Q, and with each customer $i \in V^+$ is associated a demand $d_i \in \mathbb{Z}^+$. The main difference of the QCVRP with respect to the traditional CVRP is the definition of the edge costs. They are not defined on every edge but rather on every pair of two consecutive edges. In the QSTSP, the cost function follows this last structure but now a single vehicle of infinite capacity is used to visit all customer nodes, instead of a limited fleet of K capacitated vehicles as for the QCVRP. An important difference between the problem introduced in this paper with respect to the QSTSP is the cost associated to edges adjacent to the depot. While in the QSTSP the costs incurred by two edges $\{0, i\}$ and $\{0, j\}$ depends on the sequence $i \to 0 \to j$ (or equivalently $j \to 0 \to i$ since the costs are symmetric), in the QCVRP we consider these two routing costs to be independent. From a strict modeling viewpoint, we can say that the QCVRP generalizes the original CVRP but only extends the QSTSP, as the latter cannot be modeled as a QCVRP in the strict sense.

The QCVRP can be used as modeling framework for several practical applications in vehicle routing and logistics. The Angle-CVRP is a generalization of the CVRP in which, in addition to the traveling costs on edges, we associate penalties to the angles performed between every pair of consecutive edges traversed. This problem arises in the routing of robots by taking into account some energy minimization aspects [1]. In this application, the energy consumption of a robot along its path is affected by the *smoothness* of the path, i.e. sharp turns are harder to perform than smooth turns. The CVRP with reload costs (CVRP-RC) is another generalization of the CVRP in which edges are color-labeled [26, 30, 2]. In addition to the traveling costs associated to edges, another cost is associated to the change in color between two consecutive edges. In practical applications in logistics, a color can be used to associate a particular mode of transportation to an edge. Thus, the change in color represents in fact a change in the mode of transportation, to which we associate a handling cost from unloading and reloading the freight on the vehicles implied.

We can mention two other applications of the QCVRP in logistics that require the costs matrix not to be symmetric. This problem generalizes the QCVRP and we refer to it as the Asymmetric QCVRP (AQCVRP). In the first application, the AQCVRP can be used to model turn penalties. In such setting, a set of vehicles must visit a set of predetermined customers (in a node-routing context) or edges (in an arc-routing context), subject to some additional constraints. Each node in the graph represents a corner in the city, and therefore some turns (normally left turns), are penalized or sometimes simply forbidden, so as to mimic some usual traffic regulations in urban transportation [11, 42, 7]. In the second application, the AQCVRP can be used to model and solve a clustered vehicle routing problem, as follows. Assume that a large-scale vehicle routing problem needs to be solved. One way of decreasing the computational complexity related to its size is to cluster the customer nodes so as to impose that each cluster must be visited by a single vehicle. Moreover, assuming that such vehicle visits all customers into the cluster before leaving it, one may assume the following policy when visiting three clusters in sequence –say clusters C_1, C_2 and C_3–. Let v_{12} be the node in C_1 that is the closest from C_2, and let v_{23} be the node in C_2 that is the closest from C_3. If the order in which the vehicle visits the nodes in C_2 is given by the solution of an Open TSP that starts in v_{12}, visits all the nodes in C_2 and finishes in v_{23}, a quadratic costs structure arises naturally.

The main contributions of this article are mainly four. First, we formally introduce the QCVRP by providing a strict mathematical formulation of the problem based on q-edges. Second, we develop an exact algorithm for the QCVRP based on the branch-and-cut paradigm, and show that the proposed modeling approach and solution method are useful to derive strong lower bounds of the problem in short computing times. Third, we provide a heuristic method based on state-of-the-art sub-routines especially tailored for solving the QCVRP, and capable of finding tight upper bounds of the problem quickly. Fourth, we present parallel implementations of the proposed algorithms that take advantage of current state-of-the-art multiple-core architectures.

The remainder of the article is structured as follows. In Section 2 we present a literature review of related problems and methodologies. In Section 3 we formalize the QCVRP by providing a strict mathematical formulation based on q-edges. In Section 4 we present several classes of valid inequalities for the problem. In Section 5 we present the different separation algorithms used to find violated inequalities. In Section 6 we present the exact algorithm based on the branch-and-cut paradigm to derive lower and upper bounds in an iterative manner. In Section 7 we present a heuristic algorithm for the QCVRP based on some state-of-the-art routines adapted from the vehicle routing literature. In Section 8.2 we provide an extensive computational study on several classes of instances to show the effectiveness of the proposed methodologies. Finally, Section 9 concludes the article and provides a discussion of several potential avenues for future research.

2. Literature review

At the best of our knowledge, the QCVRP has not yet been considered in the literature. However, it is closely related to some other classes of combinatorial optimization problems.

The Symmetric Traveling Salesman Problem (STSP) is one of the most classical problems in combinatorial optimization. In the STSP, a traveling salesman (or equivalently, a vehicle with infinite capacity) must visit a set of vertices and get back to the origin vertex in the minimum possible time. The traveling distances between each pair of vertices are supposed to be symmetric, this is $c_{ij} = c_{ji}$ for every pair of vertices i and j. The STSP was introduced in the seminal work of [18] in which the authors present a compact two-index formulation containing an exponential number of constraints. This integer program is solved using a branch-and-cut method, a new technique at that time. State-of-the-art algorithms for the STSP are based on the work of [18] by considering several new classes of valid inequalities and scalable separation algorithms [40, 33, 3].

The Quadratic Symmetric Traveling Salesman Problem (QSTSP) is a natural extension of the STSP in which the distances depend of every pair of two consecutive edges (namely q-edges). The QSTSP was formally introduced by [22]. The authors introduce a three-index vehicle-flow formulation of the problem which they strengthen using several classes of valid inequalities. They perform a polyhedral study and show that several classes of valid inequalities induce facets of the QSTSP polytope. Practical applications of the QSTSP include the Angle-STSP (A-STSP), a variation of the STSP in which the traveling times on two consecutive edges depend on their angle at the middle vertex [1].

The Capacitated Vehicle Routing Problem (CVRP) is the most classical variation of the STSP, in which a fleet of K identical vehicles (instead of a single vehicle as for the STSP) is used to visit exactly once each node from a predefined set of customer nodes. Every vehicle must start and end its route at a depot node, and cannot exceed its capacity while collecting the demands of the customers visited through the route. The CVRP was introduced by [19] who formally stated the problem. In [35], the authors proposed a compact two-index vehicle-flow formulation for the CVRP strengthened with capacity cuts, and developed the first exact algorithm for the problem based on the branch-and-cut paradigm. The CVRP shares with the STSP several structural properties but they also differ from an algorithmic point of view. Indeed, the capacities on vehicles induce an additional computational complexity that makes it much harder to solve in practice than the STSP. For the reader to have an idea, while the most efficient algorithm for the STSP [3] can solve problems with several thousands of customers to optimality, the most efficient exact algorithms for the CVRP [25, 5, 15] cannot solve problems containing more than 200 customers.

3. Mathematical formulation

Before formulating the QCVRP, let us define some notation. Let $V = \{0, 1, \dots, n\}$ be the set of nodes, and let node 0 be the depot node. Let $V^+ = V \setminus \{0\}$ be the set of customer nodes. With every customer $i \in V^+$ we associate a demand $d_i \in \mathbb{Z}^+$. We are given a fleet of exactly K homogeneous vehicles, each of which has a capacity of Q units of demand. Let E be the set of edges, namely $E = \{\{i, j\} : i, j \in V, i < j\}$, and let E^q be the set of q-edges, namely $E^q = \{(\{i, k\}, j) : i, j, k \in V, i < k, i \neq j, j \neq k, j \neq 0\}$. With every q-edge $e = (\{i, k\}, j) \in E^q$ is associated a traveling cost c_{ijk}. Note that in the definitions of sets E and E^q the symmetry of the network (and of the routing costs) is implicit. Indeed, if $\{i, j\}$ and $(\{i, k\}, j)$ encode edges and q-edges in E and E^q, respectively, then $\{j, i\}$ and $(\{k, i\}, j)$ encode exactly the same two objects. For the single-customer trips represented by routes of the form $0 \to i \to 0$ with $i \in V^+$, we let $2c_{0i}$ be the routing cost associated. For every customer set $S \subseteq V^+$ we define $r(S) = \lceil \sum_{i\in S} d_i/Q\rceil$, which is a lower bound on the number of vehicles needed to visit the customers in S due to the capacity constraints.

Now, let us define the variables of the model. For every $i \in V^+$ we let w_i be a binary variable equal to 1 iff customer i is served using a single-customer route. For every edge $\{i, j\} \in E^+$, with $E^+ = \{e = \{i, j\} \in E : i, j \in V^+, i < j\}$, we let z_{ij} be a binary variable equal to 1 iff customers i and j are the only two customers visited in a route. For every $e = (\{i, k\}, j) \in E^q$ we let y_{ijk} be a binary variable equal to 1 iff the q-edge e is used in a multiple-customer route (this is, a route visiting at least three customers). In addition, we let for every edge $e = \{i, j\} \in E$, x_{ij} be a binary variable equal to 1 iff edge $\{i, j\}$ is used in a multiple-customer route visiting three or more customers. Variables x are not strictly necessary as they can be derived from variables y but are still included in the model. Finally, we define binary variables ξ_{ij} for every edge $\{i, j\} \in E$ equal to 1 iff the edge $\{i, j\}$ is used by any vehicle regardless of the number of customers served in its route. Again, these edge variables are not strictly necessary as they can be derived from the previous ones, but will help in the presentation of the article as they will allow linking some of our results with previous results for the CVRP. Indeed, variables ξ correspond to the usual two-index vehicle-flow variables of the CVRP used, for instance, in [35, 38]. Note that in order to be consequent with the notation and the network symmetry, we may use variables z_{ji}, ξ_{ji}, x_{ji} and y_{kji} to encode the exact same variables as z_{ij}, ξ_{ij}, x_{ij} and y_{ijk}, respectively.

Now, let us define the following additional notation. For every edge subset $F \subseteq E$ we let $x(F) \overset{\text{def}}{=} \sum_{e\in F} x_e$ and $\xi(F) \overset{\text{def}}{=} \sum_{e\in F} \xi_e$. Analogously, for every q-edge subset $F \subseteq E^q$ we define $y(F) \overset{\text{def}}{=} \sum_{e\in F} y_e$. For every $F \subseteq E^+$ we let $z(F) \overset{\text{def}}{=} \sum_{\{i,j\}\in F} z_{ij}$. For every customer subset $U \subseteq V^+$ we let $w(U) \overset{\text{def}}{=} \sum_{i\in U} w_i$. For any two vertices sets $U, T \subset V$, we define $(U : T) \overset{\text{def}}{=} \{e = \{i, j\} \in E : (i \in U \wedge j \in T) \vee (i \in T \wedge j \in U)\}$. Also, for any three subsets $U, T, W \subset V$ we define $(U : T : W) \overset{\text{def}}{=} \{e = (\{i, k\}, j) \in E^q : (i \in U \wedge k \in W) \vee (i \in W \wedge k \in U), j \in T\}$. Now, for every vertex subset $S \subseteq V$ we let $E(S) \overset{\text{def}}{=} (S : S)$ and $\delta(S) \overset{\text{def}}{=} (S : V \setminus S)$. We also define, for every vertex subset $S \subseteq V$, $E^q(S) \overset{\text{def}}{=} (S : S : S)$ in addition to the following quantities:

$$\begin{aligned}
\delta^{ioo}(S) &\overset{\text{def}}{=} (S : V \setminus S : V \setminus S)\\
\delta^{ioi}(S) &\overset{\text{def}}{=} (S : V \setminus S : S)\\
\delta^{iio}(S) &\overset{\text{def}}{=} (S : S : V \setminus S)\\
\delta^{oio}(S) &\overset{\text{def}}{=} (V \setminus S : S : V \setminus S)
\end{aligned}$$

In addition, if $S \subseteq V^+$, we let the sets $\delta^+(S)$, $\delta^{ioo+}(S)$, $\delta^{ioi+}(S)$, $\delta^{iio+}(S)$, $\delta^{oio+}(S)$ be defined as before but without including edges or q-edges linking S to the depot. Note that if any of the sets involved in $(U : T)$ or $(U : T : W)$ is a singleton $\{i\}$, we will denote instead i.

The QCVRP can be formulated as the following integer linear program:

$$\min \quad 2\sum_{i\in V^+} c_{0i}w_i + \sum_{\{i,j\}\in E^+} (c_{0ij} + c_{0ji})z_{ij} + \sum_{(\{i,k\},j)\in E^q} c_{ijk}y_{ijk} \tag{1}$$

subject to

$$\xi(\delta(\{i\})) = 2 \qquad i \in V^+ \tag{2}$$

$$\xi(\delta(\{0\})) = 2K \tag{3}$$

$$\xi(\delta(S)) \geq 2r(S) \qquad S \subseteq V^+, 2 \leq |S| \leq |V^+| - 1 \tag{4}$$

$$y_{0ij} + y_{0ji} + z_{ij} \leq 1 \qquad i, j \in V^+, i < j \tag{5}$$

$$x_{0i} = y(0 : i : V^+ \setminus \{i\}) \qquad i \in V^+ \tag{6}$$

$$x_{ij} = y(i:j:V \setminus \{i,j\}) = y(V \setminus \{i,j\}:i:j) \qquad i,j \in V^+, i<j \qquad (7)$$
$$\xi_{ij} = x_{ij} + z_{ij} \qquad \{i.j\} \in E^+ \qquad (8)$$
$$\xi_{0i} = x_{0i} + 2w_i + z(\delta^+(\{i\})) \qquad i \in V^+ \qquad (9)$$
$$w_j \in \{0,1\} \qquad j \in V^+ \qquad (10)$$
$$z_{ij} \in \{0,1\} \qquad \{i,j\} \in E^+ \qquad (11)$$
$$y_{ijk} \in \{0,1\} \qquad (\{i,k\},j) \in E^q \qquad (12)$$
$$x_{ij} \geq 0 \qquad \{i,j\} \in E \qquad (13)$$
$$\xi_{ij} \geq 0 \qquad \{i,j\} \in E \qquad (14)$$

The objective represented by the expression (1) aims to minimize the total traveling costs. The quadratic nature of the costs is expressed in terms of variables y and z. Constraints (2) are the degree constraints. They impose that every customer be visited exactly once or, equivalently, that two edges are adjacent to it. Constraints (3) are the fleet-size constraints. They impose that exactly K vehicles be routed from the depot. Constraints (4) are the subtour-elimination constraints and capacity cuts. They forbid the appearance of subtours (closed tours not linked to the depot) and of routes exceeding the capacity of the vehicles. Constraints (5) impose that two-customer routes of the form $0 \to i \to j \to 0$ cannot be associated with the y variables but rather with the corresponding z variables. Constraints (6)–(7) are the linking constraints between the edges in E and the q-edges in E^q. Constraints (8)–(9) are the linking constraints between the ξ variables and variables x, w, z. Finally, constraints (10)–(14) are the integrality and non-negativity constraints of the decision variables. Note that variables x and ξ need not be imposed as integers, as this will be a direct consequence of the integrality of variables y, w, z and the linking constraints.

This formulation is an adaptation of the three-index formulation introduced by [22] for the QSTSP, and relies on the particular quadratic structure of the costs (they depend on pairs of two consecutive edges rather than on arbitrary pairs of edges) to derive a linear programming model with a cubic number of variables. As pointed out by the authors, this trade-off between the number of variables and the nature of the objective function (linear or quadratic) is usually positive towards the use of this cubic number of variables with a linear objective, rather than using a quadratic number of variables with a quadratic objective.

4. Valid inequalities

In this section we present several families of inequalities that are shown to be valid for the QCVRP. Some of them are adapted from the valid inequalities presented in [22] for the QSTSP while some others are adapted from existing valid inequalities for the CVRP [4, 38] and the Capacitated Location-Routing Problem (CLRP, [9, 16]). In some cases, our inequalities are lifted or tightened versions of the original ones.

4.1 Triangle inequalities

The triangle inequalities were introduced by [39] for the 0-1 Quadratic Problem (0-1 QP) and adapted by [22] for the special case of the QSTSP. Let $i, j, k \in V^+$, be three different customer nodes. The following triangle inequalities are valid for the QCVRP:

$$y_{ijk} + y_{kij} \leq x_{ij}. \qquad (15)$$

Proposition 1 *The triangle inequalities (15) are valid for the QCVRP.*

Proof. If $x_{ij} = 0$ then all q-edges using edge $\{i, j\}$ are such that the associated y variable is zero. If $x_{ij} = 1$ then for any $k \in V^+ \setminus \{i, j\}$, at most one variable y_{ijk} or y_{kij} can be equal to 1, otherwise there would be a subtour of size three. □

4.2 Small routes inequalities

The small routes inequalities were introduced by [9] for the CLRP under the name of *depot degree constraints*. For the QCVRP we can derive, under certain assumptions, three families of valid inequalities. The first assumption, shared by all three families, is that the number of vehicles is tight with respect to the demands, i.e. that $\lceil d(V^+)/Q \rceil = K$.

Let $S \subseteq V^+, |S| \geq 2$ be a subset of customers such that $d_i + d_j \leq Q$ for every two different customers $i, j \in S$. Let us assume that S and the traveling distances $(c_{0i})_{i \in V^+}, (c_{ijk})_{(\{i,k\},j) \in E^q}$ satisfy the following *triangle property*: for every pair of customers $i, j \in S, i < j$, $2(c_{0i} + c_{0j} \geq c_{0ij} + c_{0ji}$. The following single-customer routes cut is valid for the QCVRP:

$$w(S) \leq 1. \tag{16}$$

Proposition 2 *The single-customer routes cuts* (16) *are valid for the QCVRP.*

Proof. Two customers $i, j \in S$ cannot be simultaneously served by single-customer routes. Indeed, it will always be cheaper to route those two customers using a two-customer route of the form $0 \to i \to j \to 0$. □

Now, let $S \subseteq V^+, |S| \geq 4$ be such that $d_i + d_j + d_k + d_l \leq Q$ for every four different customers in S. Let us assume that S and the traveling distances satisfy the following *pentagon property*: for every four different customers $i, j, k, l \in S$, $c_{0ji} + c_{0kl} \geq c_{ijk} + c_{jkl}$. The following two-customer routes cut is valid for the QCVRP:

$$z(E(S)) \leq 1. \tag{17}$$

Proposition 3 *The two-customer routes cuts* (17) *are valid for the QCVRP.*

Proof. Four customers $i, j, k, l \in S$ cannot be simultaneously served by two different two-customer routes. Indeed, it will always be cheaper to route those four customers using one route servicing them all. □

Finally, let $S \subseteq V^+, |S| \geq 4$ be such that for every four different customers $i, j, k, l \in S$, $d_i + d_j + d_k + d_l \leq Q$. Let us assume that S and the traveling distances satisfy the triangle property, the pentagon property, in addition to the following *square property*: for every three different customers $i, j, k \in S$, $2c_{0i} + c_{0jk} \geq c_{0ij} + c_{ijk}$. The following mixed-customer routes cut is valid for the QCVRP:

$$w(S) + z(E(S)) \leq 1. \tag{18}$$

Proposition 4 *The mixed-customer routes cuts* (18) *are valid for the QCVRP.*

Proof. We already know that any two different customers in S cannot be served by single-customer routes, and that any four different customers in S cannot be served by two different two-customer routes either. In addition, the square property ensures that three different customers cannot be served one by a single-customer route, and the other two by a two-customer route. Indeed, it will always be cheaper to visit those three customers using the same vehicle. □

4.3 Odd-set inequalities

Odd-set inequalities were also introduced for the 0-1 QP by [39] and adapted for the QSTSP by [22]. Let $S \subseteq V^+$ be a subset of customers with $|S| \geq 3$ and *odd*. The following odd-set inequalities are valid for the QCVRP:

$$x(E(S)) - y(E^q(S)) \leq \left\lfloor \frac{|S|}{2} \right\rfloor. \tag{19}$$

Proposition 5 *The odd-set inequalities* (19) *are valid for the QCVRP.*

Proof. If no two consecutive edges in S are used, then the first sum of the left-hand size is bounded from above by $\lfloor |S|/2 \rfloor$. Now, for every two consecutive edges used, the corresponding q-edge must be used. □

4.4 Conflicting edges inequalities

The following inequalities are used to forbid cycles by detecting patterns of edges and q-edges that would induce subtours. They were introduced by [22] for the QSTSP. The first family is valid for every pair of customers $i, j \in V^+$:

$$x_{ij} + y(\delta^{ioi+}(\{i,j\})) \leq 1 - z_{ij} - \max\{w_i, w_j\}. \tag{20}$$

The second family of valid inequalities, in addition to customers $i, j \in V^+$, considers a partition of set $V^+ \setminus \{i,j\}$ into sets S and T satisfying $|S| \geq 1, |T| \geq 3$. The inequality is as follows:

$$x_{ij} + y(i : S : j) + y(T : i : T) \leq 1 - z_{ij} - w_i. \tag{21}$$

The third family of this class assumes the size of set T to be equal to 2, namely $T = \{u, v\}$. In that case, the following inequalities are also valid for the QCVRP:

$$x_{ij} + y(i : S : j) + y_{uiv} + y_{ujv} \leq \min\{1, 2 - 2z_{ij} - w_i - w_j\}. \tag{22}$$

Proposition 6 *The conflicting edges inequalities* (20)–(22) *are valid for the QCVRP.*

Proof. In [22], the authors proved the validity of the conflicting edges inequalities (20)–(22) for the particular case in which the right-hand sides are all equal to one. If now $w_i = 1$ or $w_j = 1$ or $z_{ij} = 1$, then the left-hand side of inequalities (20) must be equal to zero. If $w_i = 1$ or $z_{ij} = 1$, then the left-hand side of inequalities (21) must be equal to zero. Finally, if both w_i and w_j are equal to 1 or if $z_{ij} = 1$, then the left-hand-side of inequalities (22) must be necessarily equal to zero. □

An additional family of valid inequalities can be expressed to impose that at most one variable among x_{0i}, w_i and z_{ij} (for some j) can take a value of 1. They can be expressed as:

$$x_{0i} + z(\delta^+(\{i\})) \leq 1 - w_i. \tag{23}$$

The validity of these inequalities comes from the observation that a customer cannot be simultaneously served by routes of different nature.

Following a similar reasoning, the following family of inequalities is a strengthening of constraints (5) whose purpose is to forbid two-customer routes from using variables y. They are valid for every $i, j \in V^+, i \neq j$ and can be expressed as follows:

$$y_{0ij} + y_{0ji} + z(\delta^+(\{i\})) \leq 1 - w_i. \tag{24}$$

The validity of these inequalities is as follows. If $w_i = 1$ then of course all variables y and z related to i must be equal to zero. In the case $w_i = 0$, then either i is served by a two-customer route, in which case the corresponding z_{ik} variable would be equal to 1, or it is served by a multiple-customer route (visiting at least three customers), in which case i cannot be visited using two q-edges of the form $(\{0,i\}, j)$ and $(\{0,j\}, i)$.

4.5 Lifted capacity cuts

In this section we present two families of valid inequalities that, in the same spirit of the capacity cuts (4), are used to forbid vehicle routes from visiting customers that would not fit into a vehicle due to the capacity restrictions. One of the two families is called lifted capacity cuts. As its name suggests, they represent a lifting of the original capacity cuts (4) and of the y-capacity cuts introduced by [9, 16] for the CLRP. Following the same reasoning used by [22] for the QSTSP, one can observe that the q-edges with both extremities inside of

S and the middle vertex outside of it can be omitted from the cut since they represent vehicles that never left set S. The following inequality is then valid for the QCVRP for every set $S \subseteq V^+$:

$$\xi(\delta(S)) - 2y(\delta^{ioi+}(S)) \geq 2r(S). \tag{25}$$

Lemma 1 *Inequalities (25) are valid for the QCVRP.*

Proof. First, note that without loss of generality we may assume that $2w(S) + 2z(E(S)) + 2z(\delta^+(S)) = 0$. Indeed, for each of these terms equals to one, one can remove the corresponding customers from S and prove the inequality for the remaining ones, because each time that a variable w or z is equal to one, the accumulated demand associated to each route is not greater than Q and thus $r(S)$ cannot decrease by more than one unit. Let us assume first that $r(S) = 1$. If the q-edges leaving from S were only of the form $(\{i,k\}, j), i, k \in S, j \in V^+ \setminus S$, then there would be a subtour. Thus, there must be at least two edges that are not of this form which will make the left-hand side of constraint (25) become at least 2. Let us assume now that $r(S) > 1$. If there were $2k$ q-edges leaving from S, with $k < r(S)$, one could follow these q-edges (thanks to the degree constraints) that would eventually get together to the depot creating at most k routes, one of which at least would violate the capacity constraint, or some of them would eventually meet in some other customer, thus creating a subtour. $\square$

Note that this inequality is valid for every set S and not only for sets of size strictly smaller than $|V^+|/2$ as for the QSTSP since tours must necessarily be connected to the depot.

The next lifting of this last inequality is a generalization of the following observation due to [9] for the CLRP. One can strengthen the capacity cut by not considering some routes visiting one or two customers inside S. More precisely, the following inequality is valid for every set $S \subseteq V^+$, and for every subset $S' \subset S$ such that $r(S) = r(S \setminus S')$:

$$x(\delta(S)) - 2y(\delta^{ioi+}(S)) + 2(w(S \setminus S') + z(E(S \setminus S')) + z(\delta^+(S \setminus S'))) \geq 2r(S). \tag{26}$$

Proposition 7 *Inequalities (26) are valid for the QCVRP.*

Proof. Let $S'' \subseteq S'$ be the set of customers $i \in S'$ that are either served by a single-customer route (in which case variable w_i would be equal to one) or by a two-customer route (in which case the corresponding z_{ij} variable would take the value 1). We will now show that that the left-hand side of inequality (26) is greater than or equal to the left-hand side of inequality (25) evaluated in set $S \setminus S''$. Indeed, the three last terms must satisfy:

i.
$$w(S \setminus S') = w(S \setminus S'') - w(S' \setminus S'') = w(S \setminus S'').$$

The last equality follows from the definition of S''.

ii.
$$\begin{aligned} z(E(S \setminus S')) &= z(E(S \setminus S'')) - z(S \setminus S' : S' \setminus S'') - z(E(S' \setminus S'')) \\ &= z(E(S \setminus S'')) - z(S \setminus S' : S' \setminus S''). \end{aligned}$$

Again, the last equality follows from the definition of set S''.

iii.
$$\begin{aligned} z(\delta^+(S \setminus S')) =& z(\delta^+(S \setminus S'')) + z(S \setminus S' : S' \setminus S'') \\ &- z(S' \setminus S'' : V^+ \setminus S) - z(S' \setminus S'' : S'') \\ =& z(\delta^+(S \setminus S'')) + z(S \setminus S' : S' \setminus S''). \end{aligned}$$

Again, the last equality follows from the definition of set S''.

For the first two expressions we can apply similar reasonings to obtain the following relationships:

iv.
$$x(\delta(S)) = x(\delta(S \setminus S'')).$$

v.
$$y(\delta^{ioi+}(S)) = y(\delta^{ioi+}(S \setminus S'')).$$

These identities follow from the definition of set S'' by realizing that no variable x or y visiting nodes in S'' will take a positive value.

Now, we add all these five identities to obtain that the left-hand side of constraint (26) is equal to the left-hand side of constraint (25) evaluated in $S \setminus S''$. Because $r(S) = r(S \setminus S'')$ the result follows. □

Remark 1 The lifted capacity cuts (26) dominate the y-capacity cuts introduced by [9] because of the tighter left-hand side. They also extend the lifted subtour elimination constraints of [22] valid for the QSTSP.

The second family of capacity cuts takes into account q-edges only. Given that q-edges visit two customers each, one can derive an inequality that does not dominate, nor is dominated by the previous family of valid inequalities. Let $S \subset V^+$ be a customer subset of size $|S| \geq 3$. Let us define $\rho(S) = \min\{|S|-1, r(S)\}$. The following family of q-capacity cuts is valid for the QCVRP:

$$y(E^q(S)) \leq |S| - 1 - \rho(S). \tag{27}$$

Lemma 2 *The q-capacity cuts (27) are valid for the QCVRP.*

Proof. If $r(S) = |S|$ then all customers in S must be visited by different vehicles and thus $y(E^q(S)) = 0$. Therefore, let us assume that $r(S) \leq |S| - 1$. Because there are at least $r(S)$ vehicles servicing S, then S can be partitioned into $n \geq r(S)$ non-empty sets $S_i, i = 1, \ldots, n$ such that

$$|S_i| - y(E^q(S_i)) = \begin{cases} 1 & \text{if } |S_i| = 1 \\ 2 & \text{if } |S_i| \geq 2 \end{cases}$$

and $|S| - y(E^q(S)) = \sum_{1 \leq i \leq n} |S_i| - y(E^q(S_i))$. Let us define $l_1 = |\{i : 1 \leq i \leq n, |S_i| = 1\}|, l_2 = |\{i : 1 \leq i \leq n, |S_i| \geq 2\}|$. Thus, the following identity holds:

$$|S| - y(E^q(S)) - 1 - r(S) = l_1 + 2l_2 - 1 - r(S).$$

If $l_2 = 0$, then $l_1 = |S|$ and one has $l_1 + 2l_2 - r(S) - 1 = |S| - r(S) - 1 \geq 0$. If $l_2 \geq 1$ then $l_1 + 2l_2 - r(S) - 1 = (n - r(S)) + (l_2 - 1) \geq 0$. □

As for the lifted capacity cuts (25)–(26), one can derive a similar lifting to strengthen the left-hand side of the inequality. More precisely, let $S' \subset S$ be such that $\rho(S) = \rho(S \setminus S')$. The following lifted q-capacity cut is valid for the QCVRP:

$$y(E^q(S)) + w(S') + 2z(E(S')) + z(S' : V^+ \setminus S) + y(V \setminus S : S' : V \setminus S) \leq |S| - 1 - \rho(S). \tag{28}$$

Proposition 8 *The lifted q-capacity cuts (28) are valid for the QCVRP.*

Proof. Let $S'' \subseteq S'$ be the subset of customers of S' being served by single-customer routes, two-customer routes or by multiple-customer routes using a q-edge in $(V \setminus S : S' : V \setminus S)$. Let us define $\gamma(S, S') = w(S') + 2z(E(S')) + z(S' : V^+ \setminus S) + y(V \setminus S : S' : V \setminus S)$. One has that

$$\gamma(S, S') = |S''|.$$

Also, because of the definition of S'', one also has that

$$y(E^q(S \setminus S'')) = y(E^q(S)).$$

Using these two identities in addition to the q-capacity cut (27), one can realize that the following identity and inequality also hold:

$$y(E^q(S)) + \gamma(S, S') = y(E^q(S \setminus S'')) + |S''| \leq |S \setminus S''| - 1 - \rho(S \setminus S'').$$

Because $\rho(S \setminus S') \leq \rho(S \setminus S'') \leq \rho(S)$ the result follows. □

4.6 Comb inequalities

The following comb inequalities for the QCVRP are a lifting of the ones proposed by [34] and later generalized by [38]. Unlike the comb inequalities of the STSP [13, 32], the ones for the CVRP take into account the vehicle capacities to derive a stronger cut. Our generalization of the strengthened comb inequalities of [38] makes use of the lifted capacity cuts (25) to derive even stronger cuts. Let $H \subset V^+$ be customer set (namely the *handle*) and sets $\mathcal{T} = \{T_j \subset V : j = 1, \ldots, t\}$ (namely the *teeth*) satisfying:

- $T_j \cap H \neq \emptyset$ for all $j = 1, \ldots, t$.
- $T_j \setminus H \neq \emptyset$ for all $j = 1, \ldots, t$.
- for each $\{i, j\} \subset \{1, \ldots, t\} : T_i \cap T_j \subset H$ or $T_i \cap T_j \cap H = \emptyset$.

For every tooth $T \in \mathcal{T}$ with $0 \notin T$ we define $\sigma(H,T) = y(T \setminus H : V \setminus T : T \setminus H) + y(T \cap H : V \setminus T : T \cap H)$ and $\pi(H,T)$ as:

$$\pi(H,T) \overset{\text{def}}{=} \begin{cases} \xi(\delta(T)) - 2\sigma(H,T) & \text{if } 0 \notin T \\ \xi(\delta(T)) & \text{if } 0 \in T \end{cases} \tag{29}$$

We also define, for every set $U \subseteq V$, $\tilde{r}(U) = r(U)$ if $0 \notin U$ and $\tilde{r}(U) = r(V \setminus U)$ otherwise. Finally, we define the quantity $R(H, \mathcal{T}) = \sum_{j=1}^{t} (\tilde{r}(T_j) + \tilde{r}(T_j \setminus H) + \tilde{r}(T_j \cap H))$. If $R(H, \mathcal{T})$ is *odd*, the following comb inequality is valid for the QCVRP:

$$\xi(\delta(H)) + \sum_{j=1}^{t} \pi(H, T_j) \geq R(H, \mathcal{T}) + 1. \tag{30}$$

Proposition 9 *The comb inequalities* (30) *are valid for the QCVRP.*

Proof. The validity proof for the comb inequalities (30) consists in showing that their left-hand sides can be bounded from below by

$$\frac{1}{2} \sum_{T \in \mathcal{T}} [\xi(\delta(T)) + \xi(\delta(T \setminus H)) + \xi(\delta(T \cap H))] - \sum_{T \in \mathcal{T}, 0 \notin T} \left[y(\delta^{ioi+}(T)) + y(\delta^{ioi+}(T \setminus H)) + y(\delta^{ioi+}(T \cap H)) \right].$$

If the above is true, the result follows from using the capacity constraint for the teeth $T \in \mathcal{T}, 0 \notin T$, and by noticing that $\xi(\delta(T)) = \xi(\delta(V \setminus T))$ if $0 \in T$ to then use the capacity constraints on the sets $V \setminus T$, $V \setminus (T \setminus H)$ and $T \cap H$. The right-hand side of (30) follows from the observation that the left-hand side of the inequality is *even* while $R(H, \mathcal{T})$ is *odd*.

From the proof of the original comb cuts made for the CVRP [34, 38], the following identity holds:

$$2\xi(\delta(H)) + 2\sum_{j=1}^{t}\xi(\delta(T_j)) \geq \sum_{j=1}^{t}\left(\xi(\delta(T_j)) + \xi(\delta(T_j \setminus H)) + \xi(\delta(T_j \cap H))\right).$$

Therefore, we must prove that the following is true:

$$2\sum_{T\in\mathcal{T},0\notin T}\sigma(H,T) \leq \sum_{T\in\mathcal{T},0\notin T}\left[y(\delta^{ioi+}(T)) + y(\delta^{ioi+}(T\setminus H)) + y(\delta^{ioi+}(T\cap H))\right].$$

Indeed, let us develop the terms appearing in the right-hand side of this inequality, for every $T \in \mathcal{T}, 0 \notin T$:

i. $y(\delta^{ioi+}(T)) = y(T\setminus H : H\setminus T : T\setminus H) + y(T\setminus H : V\setminus(H\cup T) : T\setminus H) + y(T\cap H : H\setminus T : T\cap H) + y(T\cap H : V\setminus(H\cup T) : T\cap H) + y(T\setminus H : H\setminus T : T\cap H) + y(T\setminus H : V\setminus(H\cup T) : T\cap H)$.

ii. $y(\delta^{ioi+}(T\setminus H)) = y(T\setminus H : H\setminus T : T\setminus H) + y(T\setminus H : T\cap H : T\setminus H) + y(T\setminus H : V\setminus(H\cup T) : T\setminus H)$.

iii. $y(\delta^{ioi+}(T\cap H)) = y(T\cap H : H\setminus T : T\cap H) + y(T\cap H : T\setminus H : T\cap H) + y(T\cap H : V\setminus H\setminus T : T\cap H)$.

Therefore, for each set $T \in \mathcal{T}, 0 \notin T$, one has:

$$\begin{aligned} y(\delta^{ioi+}(T)) &+ y(\delta^{ioi+}(T\setminus H)) + y(\delta^{ioi+}(T\cap H)) \\ &= 2\sigma(H,T) + y(T\cap H : T\setminus H : T\cap H) \\ &\quad + y(T\setminus H : T\cap H : T\setminus H) + y(T\setminus H : V\setminus T : T\cap H) \\ &\geq 2\sigma(H,T). \end{aligned}$$

By adding all these terms we obtain the desired result. □

Remark 2 The comb inequalities (30) dominate the strengthened comb inequalities of [38] because of the terms $\sigma(H,T)$ with negative coefficients in the left-hand side of the constraint.

4.7 Framed capacity inequalities

The framed capacity inequalities were originally introduced by [4] for the CVRP. Their validity and strength are based upon the following observation. A capacity cut for a set S may be satisfied for a certain fractional solution, but if we are able to extract from that solution some clustering information about the subsets of customers that are visited in sequence, the capacity cut may be strengthened because the specific clustering or ordering may impose the need of a larger number of vehicles than $r(S)$.

Let $T \subseteq V^+$ be a customer subset, and let $\mathcal{S} = (S_j)_{j=1}^t$ be a partition of T, i.e. such that $\bigcup_{j=1}^t S_j = T$, $S_i \cap S_j = \emptyset$ for every $1 \leq i < j \leq t$. Let $BPP(T|\mathcal{S})$ represent the solution of the following bin-packing problem. For every $i = 1,\dots,t$ consider $n_i = \lceil d(S_i)/Q\rceil$ items, each of which has size Q except for the last item that will have size $d(S_i) - (n_i - 1)Q$, and let the bins have capacity Q. The following framed capacity inequality is valid for the QCVRP:

$$\xi(\delta(T)) - 2y(\delta^{ioi+}(T)) + \sum_{j=1}^{t}\left[\xi(\delta(S_j)) - 2y(\delta^{ioi+}(S_j))\right] \geq 2\left(BPP(T|\mathcal{S}) + \sum_{j=1}^{t} r(S_j)\right). \tag{31}$$

The validity proof of these cuts makes use of the following lemma:

Lemma 3 ([4]) *Let $T \subseteq V^+$ and $\mathcal{S} = (S_j)_{j=1}^t$ be a partition of T. If $\lceil d(S_1\cup S_2)/Q\rceil = \lceil d(S_1)/Q\rceil + \lceil d(S_2)/Q\rceil$ then $BPP(T|S_1,S_2,\dots,S_t) \geq BPP(T|S_1\cup S_2, S_3,\dots,S_t)$. Otherwise $BPP(T|S_1,S_2,\dots,S_t) + 1 \geq BPP(T|S_1\cup S_2,S_3,\dots,S_t)$.*

Proof. See [4]. □

Proposition 10 *The framed capacity inequalities* (31) *are valid for the QCVRP.*

Proof. Let us suppose first that the sets S_j satisfy $d(S_j) \leq Q$. Let us consider the bin-packing problem defined above, with objects of sizes $d(S_j)$ for every $j = 1, \dots, t$ and bin size equal to Q. Let us denote the set of objects by K. In this context, We denote by *cut of object k in K* the following operation: remove k (of size $d(k)$) from K and replace it by two smaller objects whose total size is equal to $d(k)$. It is known that after a cut operation, the solution of the BPP is reduced by at most one unit, so after q cut operations the solution of the BPP is reduced by at most q units. In our case, the quantity $\eta = \frac{1}{2}\sum_{j=1}^{t}(\xi(\delta(S_j)) - 2y(\delta^{ioi+}(S_j)) - 2)$ represents the number of cuts that are applied to the set T, and thus $BPP(T|\mathcal{S}) - \eta$ is a lower bound on the number of vehicles needed to serve the demand of T.

Now, in the general case in which the demands of the sets in $\mathcal{S}$ exceed Q, let (ξ, x, y, w, z) be a feasible solution of (1)–(14). For every subset S_j, (ξ, x, y, w, z) defines a partition $S_j^k, k = 1, \dots n_j$ of subsets of S_j such that *i)* $\xi(\delta(S_j^k)) - 2y(\delta^{ioi+}(S_j^k)) = 2$ and *ii)* $n_j = \frac{1}{2}[\xi(\delta(S_j)) - 2y(\delta^{ioi+}(S_j))]$. Because $d(S_j^k) \leq Q$ for every $j = 1, \dots, t$ and $k = 1, \dots, n_j$ we can apply the previous result and then it holds that $\xi(\delta(T)) - 2y(\delta^{ioi+}(T)) \geq 2BPP(T|(S_1^k)_{k=1}^{n_1}, (S_2^k)_{k=1}^{n_2}, \dots, (S_t^k)_{k=1}^{n_t})$. For every $j = 1, \dots, t$ we apply n_j successive contractions of the subsets S_j^k and compute $\alpha(j, l)$ equal to the number of times that $BPP(T|(S_1^k)_{k=1}^{n_1}, (S_2^k)_{k=1}^{n_2}, \dots, (S_t^k)_{k=1}^{n_t})$ decreases by one unit after a contraction. By applying the lemma, we have that $\alpha(j,1) = \lceil d(S_j^1)/Q \rceil + \lceil d(S_j^2)/Q \rceil - \lceil d(S_j^1 \cup S_j^2)/Q \rceil = 2 - \lceil d(S_j^1 \cup S_j^2)/Q \rceil$ and, more generally, $\alpha(j,l) = l + 1 - \lceil d(\bigcup_{k=1}^{l} S_j^k)/Q \rceil$. At the end of all of these successive contractions we will have that $\xi(\delta(T)) - 2y(\delta(^{ioi+}(T)) \geq 2[BPP(S|\mathcal{S}) - \sum_{j=1}^{t}(n_j - \lceil d(S_j)/Q \rceil)]$. □

Remark 3 The framed capacity inequalities (31) dominate the original ones from the CVRP due to the tighter left-hand side.

4.8 Generalized large multistar inequalities

The generalized large multistar inequalities were introduced by several authors for the CVRP [23, 31]. A lifting of the cuts was proposed in [16] by noticing that when the w variables are used, the associated customers can be omitted from the cut to derive a stronger inequality. We propose a similar lifting valid for the QCVRP, that now takes into account also the z variables and the fact that the q-edges $\delta^{ioi+}(S)$ can be omitted from the capacity cut. Let us denote the demand of the depot by d_0 and let it be equal to 0. Let $S \subset V^+$ be a customer subset. The following generalized large multistar inequality is valid for the QCVRP:

$$x(\delta(S)) - 2y(\delta^{ioi+}(S)) + \frac{2}{Q}\left(\sum_{j\in S} d_j w_j + \sum_{\{i,j\}\in E(S)} (d_i + d_j)\, z_{ij} + \sum_{i\in S, j\in V^+\setminus S} d_i z_{ij}\right)$$
$$\geq \frac{2}{Q}\left(d(S) + \sum_{(\{i,k\},j)\in\delta^{ioi+}(S)} d_j y_{ijk} + \sum_{i\in S,(\{i,k\},j)\in\delta^{ioo+}(S)} (d_j + d_k) y_{ijk}\right). \quad (32)$$

Proposition 11 *The generalized large multistar inequalities* (32) *are valid for the QCVRP.*

Proof. Without loss of generality we may assume that

$$g(S) \stackrel{\text{def}}{=} \sum_{j\in S} d_j w_j + \sum_{\{i,j\}\in E(S)} (d_i + d_j)\, z_{ij} + \sum_{i\in S, j\in V^+\setminus S} d_i z_{ij}.$$

is equal to zero. Indeed, if $g(S) > 0$ we can identify a set $S' \subseteq S$ such that $g(S) = \sum_{i\in S'} d_i$. We can subtract $g(S)$ from $d(S)$ in the right-hand side of the constraint and prove the inequality for $S \setminus S'$. Now, the expression

$$f(S) \stackrel{\text{def}}{=} d(S) + \sum_{(\{i,k\},j)\in\delta^{ioi+}(S)} d_j y_{ijk} + \sum_{i\in S,(\{i,k\},j)\in\delta^{ioo+}(S)} (d_j + d_k) y_{ijk},$$

corresponds to the demands associated to customers being visited by the same vehicles that visit S, and thus $f(S)/Q$ is a lower bound on the number of vehicles needed to visit S. The result follows from the same reasoning applied to the lifted capacity constraints (25) to discard from the left-hand side the term $2y(\delta^{ioi+}(S))$. □

The following proposition states that constraints (32) dominate the classical generalized large multistar inequalities from the CVRP:

$$\xi(\delta(S)) \geq 2\left(d(S) + \sum_{i\in S, j\in V^+\setminus S} d_j \xi_{ij}\right). \tag{33}$$

Proposition 12 *The generalized large multistar inequalities (32) dominate constraints (33).*

Proof. Let us define the following quantity:

$$\alpha(S) \overset{\text{def}}{=} \sum_{j\in S}(Q-d_j)w_j + \sum_{\{i,j\}\in E(S)}(Q-d_i-d_j)z_{ij} + \sum_{i\in S, j\in V^+\setminus S}(Q-d_i)z_{ij}.$$

Let us consider constraint (32) and let us add $\frac{2}{Q}\alpha(S) + 2y(\delta^{ioi+}(S))$ at both sides of the inequality. The new left-hand side will become $\xi(\delta(S))$. The new right-hand side will become:

$$\frac{2}{Q}\left(d(S) + \alpha(S) + \sum_{(\{i,k\},j)\in\delta^{ioi+}(S)}(d_j+Q)y_{ijk} + \sum_{i\in S, (\{i,k\},j)\in\delta^{ioo+}(S)}(d_j+d_k)y_{ijk}\right). \tag{34}$$

We can use the fact that $d_j + Q \geq 2d_j$ and that $d_j + d_k \geq d_j$ for every $j,k \in V^+$ to derive the following inequality:

$$\sum_{(\{i,k\},j)\in\delta^{ioi+}(S)}(d_j+Q)y_{ijk} + \sum_{i\in S, (\{i,k\},j)\in\delta^{ioo+}(S)}(d_j+d_k)y_{ijk} \geq \sum_{i\in S, j\in V^+\setminus S} d_j x_{ij}.$$

In addition, we can use the fact that $Q - d_j \geq 0$ for every $j \in S$, $Q - d_i - d_j \geq 0$ for every $\{i,j\} \in E(S)$ such that $z_{ij} > 0$ and that $Q - d_i \geq d_j$ for every $i \in S, j \in V^+ \setminus S$ such that $z_{ij} > 0$ to derive the following inequality involving $\alpha(S)$:

$$\alpha(S) \geq \sum_{i\in S, j\in V^+\setminus S} d_j z_{ij}.$$

If we use these two inequalities we can bound the expression (34) from below by

$$\frac{2}{Q}\left(d(S) + \sum_{i\in S, j\in V^+\setminus S} d_j \xi_{ij}\right).$$

□

4.9 Generalized hypotour inequalities

The hypotour inequalities were introduced by [4] for the CVRP. If one can detect a set of edges F such that any feasible solution of the CVRP uses at least one edge in F, then the inequality $\xi(F) \geq 1$ is valid for the CVRP. In [10], the authors considered several variations of hypotour inequalities, including the so-called 2-edges extended hypotour inequalities. These inequalities were also considered by [38] who introduced a polynomial-time heuristic for their separation based on the solution of the Hungarian algorithm for the assignment problem. We consider two families of hypotour inequalities for the QCVRP, one of which includes the 2-edges extended hypotour inequalities as a particular case, and whose separation can also be done in polynomial time for fixed values of $r(S)$ (the separation will be given in detail in Section 5).

Let $S \subseteq V^+$ be a subset of customers and let us denote $k = r(S)$. Let $H = \{e_l : 1 \leq l \leq 2k\} \subset \delta^{ioo+}(S)$ be a subset of $2k$ different q-edges, with $e_l = (\{i_l, k_l\}, j_l)$ and $j_l, k_l \in V^+ \setminus S$ for every l. We denote $W \overset{\text{def}}{=} S \bigcup_{1 \leq l \leq 2k} \{j_l, k_l\}$. We also define $F \overset{\text{def}}{=} E(V \setminus W) \cup (V \setminus W : \{k_l : 1 \leq l \leq 2k\})$ and let $F_0 \subset F$ be a subset of edges. Let $\widehat{G} = (\widehat{V}, \widehat{E})$ be the following auxiliary graph. The vertex set $\widehat{V}$ is defined as $\widehat{V} = (V \setminus W) \cup \{k_l : 1 \leq l \leq 2k\} \cup \{0'\}$ (with $0'$ being a dummy node), and the edge set $\widehat{E}$ is defined as $\widehat{E} = (F \setminus F_0) \cup \{\{k_l, 0'\} : 1 \leq l \leq 2k\}$. With each edge $e = \{i, j\} \in \widehat{E}$ we associate a weight $\omega_e = d_i + d_j$ (with the convention that $d_0 = d_{0'} = 0$). If we denote by $\{P_l : 1 \leq l \leq 2k\}$ the $2k$ shortest customer-disjoint $00'$-paths in $\widehat{G}$ and by $\{\omega(P_l) : 1 \leq l \leq 2k\}$ their lengths, then if $\sum_{1 \leq l \leq 2k} \omega(P_l) > 2(kQ - d(S) - \sum_{1 \leq l \leq 2k} d_{j_l})$, the following generalized $2k$-q-edges hypotour inequality is valid for the QCVRP:

$$\left(\frac{2k-1}{k}\right)\left[\xi(\delta(S)) - 2y(\delta^{ioi+}(S))\right] + 2x(F_0) \geq 2y(H). \tag{35}$$

Proposition 13 *The generalized $2k$-q-edges hypotour inequalities (35) are valid for the QCVRP.*

Proof. If $y(H) \leq 2k-1$ then the inequality is automatically satisfied from the fact that $\xi(\delta(S)) - 2y(\delta^{ioi+}(S)$ $\geq 2k$. If $\xi(\delta(S)) - 2y(\delta^{ioi+}(S)) \geq 2(k+1)$ the inequality is also automatically satisfied. The only interesting case is then whenever $\xi(\delta(S)) - 2y(\delta^{ioi+}(S)) = 2k$ and $y(H) = 2k$. In that case, there are exactly k vehicles crossing S, each of which uses two q-edges in H to connect S to the depot. If the tours are not allowed to use the edges in F_0 and their minimum accumulated demands exceed kQ, then at least one edge in F_0 must be used. □

If $r(S) = 1$, the generalized $2k$-q-edges hypotour inequalities (35) do not dominate nor contain as a particular case the classic 2-edges extended hypotour inequalities. Indeed, even if the left-hand side of inequalities (35) does not involve q-edges in $\delta^{ioi+}(S)$, their right-hand side may be weaker than that of the 2-edges extended hypotour inequalities. However, we can identify another family of cuts that generalizes and dominates the 2-edges extended hypotour inequalities [10, 38].

Let $S \subseteq V^+$ be a subset of customers and let $k = r(S)$. Let $H \overset{\text{def}}{=} \{e_l = \{u_l, v_l\} : u_l \in S, v_l \in V^+ \setminus S, 1 \leq l \leq 2k\} \subseteq \delta^+(S)$ be a subset of edges in the boundary of S of size $|H| = 2k$. Let $W \overset{\text{def}}{=} S \bigcup_{1 \leq l \leq 2k} \{v_l\}$ and $F \overset{\text{def}}{=} E(V \setminus W) \cup (V \setminus W : \{v_l : 1 \leq l \leq 2k\})$. Also, let $F_0 \subset F$ be a subset of S. Let $\widehat{G} = (\widehat{V}, \widehat{E})$ be the following graph. The vertex set $\widehat{V}$ is defined as $\widehat{V} = (V \setminus W) \cup \{v_l : 1 \leq l \leq 2k\} \cup \{0'\}$, with $0'$ being a dummy node. We let $\widehat{E} = (F \setminus F_0) \cup \{\{v_l, 0'\} : 1 \leq l \leq 2k\}$. With every edge $e = \{u, v\} \in \widehat{E}$ we associate a weight $\omega_e = d_u + d_v$ (we assume that $d_0 = d_{0'} = 0$). If the $2k$ shortest customer-disjoint $00'$-paths $\{P_l : 1 \leq l \leq 2k\}$ are such that $\sum_{1 \leq l \leq 2k} \omega(P_l) > 2(kQ - d(S))$, the following generalized $2k$-edges hypotour inequality is valid for the QCVRP:

$$\left(\frac{2k-1}{k}\right)\xi(\delta(S)) + 2x(F_0) \geq 2\xi(H). \tag{36}$$

Proposition 14 *The generalized $2k$-edges hypotour inequalities (36) are valid for the QCVRP.*

Proof. It uses the exact same arguments as the validity proof of the generalized hypotour inequalities (35). For the sake of brevity we prefer to omit the details. □

Remark 4 If $r(S) = 1$, the associated generalized $2k$-edges hypotour inequalities (36) dominate the classic 2-edges extended hypotour inequalities [10, 38] of the CVRP because of the term $2x(F_0)$ in the left-hand side of the inequality that does not consider single-customer and two-customer routes. Also, if we replace $x(F_0)$ by $\xi(F_0)$ in (36) we obtain a new class of hypotour inequality valid for the classical two-index formulation of the CVRP that extend the classical 2-edges hypotour inequalities used, for instance, in [38].

4.10 Blossom inequalities

The blossom inequalities were introduced by [21] in the context of the 2-matching polytope. Because the 2-matching polytope is as a relaxation of the STSP polytope, they are also valid for the STSP [41]. Surprisingly, they had not been used for routing-like problems until a recent branch-and-cut algorithm for the CLRP [9]. In this article, we introduce the following blossom inequalities. Let $S \subset V$ be a node subset, and let $F \subseteq \delta(S)$ be an edge subset of the boundary of S. If $|F|$ is *odd*, the following inequality is valid for the QCVRP:

$$x(\delta(S) \setminus F) - x(F) + |F| \geq 1. \tag{37}$$

Proposition 15 *The blossom inequalities* (37) *are valid for the QCVRP.*

Proof. if $x(F) < |F|$ the inequality is automatically satisfied. If $x(F) = |F|$ then at least one edge $e \in \delta(S) \setminus F$ must be such that $x_e = 1$ so as to recover the parity of $x(\delta(S))$. □

5. Separation algorithms

Let $QCVRP_L$ be the linear relaxation of (1)–(14). This is, the linear program resulting from replacing the binary conditions (10)–(14) by the corresponding linear conditions. Given a solution $(\overline{\xi}, \overline{x}, \overline{y}, \overline{w}, \overline{z})$ of $QCVRP_L$, a separation algorithm for a family of inequalities $\mathcal{F}$ is a method receiving $(\overline{\xi}, \overline{x}, \overline{y}, \overline{w}, \overline{z})$ as input and returning an inequality valid for $\mathcal{F}$ violated by $(\overline{\xi}, \overline{x}, \overline{y}, \overline{w}, \overline{z})$, if one exists. Note that a separation algorithm may fail in finding a violated valid inequality, in which case we refer to it as *heuristic*. Otherwise, it is said *exact*.

In this section we present separation algorithms, exact and heuristics, for all the classes of valid inequalities introduced in this paper.

First, let us note that the triangle inequalities (15) and the conflicting edges inequalities (20), (22)–(23), can be separated in polynomial time by exhaustive enumeration. This separation is exact.

The small routes inequalities (16)–(18) are not separated dynamically but rather included from the beginning of the algorithm. In particular, we do not add them for every possible subset $S \subseteq V^+$, but we rather consider the sets $V^+_{Q/2} = \{i \in V^+ : d_i \leq Q/2\}$ and $V^+_{Q/4} = \{i \in V^+ : d_i \leq Q/4\}$. We check if $V^+_{Q/2}$ satisfies the triangle property, in which case we add the corresponding inequality (16). Then, we check if the pentagon property holds for the set $V^+_{Q/4}$, in which case we add the corresponding inequality (17). Finally, if the triangle and pentagon properties hold for $V^+_{Q/4}$, we additionally check the square property before adding the corresponding inequality (18).

The separation of the odd-set inequalities (19) is done using an iterative, constructive algorithm. For any set S, let us define

$$\beta(S) := \overline{x}(E(S)) - \overline{y}(E^q(S)) - \lfloor |S|/2 \rfloor.$$

Our algorithm starts by selecting an initial set of customers S satisfying

$$S = arg\,max\{\beta(T) : T \subseteq V^+, |T| = 3\}. \tag{38}$$

At each iteration, we first check if set S defines a violated odd-set inequality. If not, and if $|S| \leq |V^+| - 2$ we enlarge set S by adding the two nodes $u, v \in V^+ \setminus S$ that satisfy

$$\{u, v\} = arg\,max\{\beta(S \cup \{u, v\}) : u, v \in V^+ \setminus S, u < v\}.$$

To compute $\beta(S \cup \{u, v\})$ efficiently, we make use of the following identity:

$$\beta(S \cup \{u,v\}) = \beta(S) - 1 + \overline{x}_{uv} + \sum_{i\in S}(\overline{x}_{iu} + \overline{x}_{iv}) - \sum_{i\in S}(\overline{y}_{iuv} + \overline{y}_{uvi} + \overline{y}_{viu}) \\ - \sum_{i\in S}\sum_{j\in S, j>i}(\overline{y}_{iju} + \overline{y}_{jui} + \overline{y}_{uij} + \overline{y}_{ijv} + \overline{y}_{jvi} + \overline{y}_{vij}).$$

If a violated odd-set inequality is found using this procedure, we try to obtain an additional violated cut, as follows. In the next iteration, we consider as initial set S the one that solves (38) but such that at least one of the three customers does not belong to any of the recently violated inequalities found. This is done in an iterative manner until no more violated odd-set inequalities can be detected.

To separate the conflicting edges inequalities (21) we formulate the separation problem, for each $i, j \in V^+, i \neq j$, as the following minimum-cut problem that is solvable in polynomial time.

Let $\widehat{G} = (\widehat{V}, \widehat{A})$ be an auxiliary digraph, constructed as follows. The node set is composed by two auxiliary nodes s, t and the union of two sets, namely $\widehat{V} = \{s,t\} \cup \widehat{V}_1 \cup \widehat{V}_2$, with $\widehat{V}_1 = V^+ \setminus \{i,j\}$ and $\widehat{V}_2 = \{m = \{k,l\} : k,l \in V^+ \setminus \{i,j\}, k < l\}$. The arc set $\widehat{A}$ is constructed as follows. For each $u \in \widehat{V}_1$ we will consider an arc (u,t) with capacity $\overline{y}_{iuj}$. For every $m = \{k,l\} \in \widehat{V}_2$ we will consider an arc (s,m) with capacity $\overline{y}_{kil}$. Finally, for every $m = \{k,l\}$ we will consider two arcs (m,k) and (m,l) with capacities $+\infty$. By the construction of $\widehat{G}$, a violated inequality (21) exists if and only if for some $i, j \in V^+, i \neq j$, the corresponding maximum-flow in the auxiliary graph has a capacity strictly greater than $1 - \overline{x}_{ij} - \overline{z}_{ij} - \overline{w}_i$. This construction was proposed by [22] but unlike their implementation that uses the total unimodularity of the constraint matrix to solve it as a linear program, we make use of a specialized algorithm for the maximum-flow problem [28].

Now, let us present the heuristic separation algorithm designed for finding the lifted capacity cuts (26). We first use a modified version of the implementation of [38] for the separation of capacity cuts (4). The modification makes use of a parameter $\epsilon > 0$ to find customer subsets S that either violate a capacity cut (4) or, if not, the difference between its right-hand side and left-hand side is of at most ϵ. For every set S found, we consider the lifted capacity cut (25) produced by removing all q-edges leaving and re-entering set S. Let us define $\kappa = d(S) + Q - 1 \pmod{Q}$. If $\kappa = 0$, we make $S' = \emptyset$ and check if S violates inequality (25), in which case we add this cut to (1)–(14). Otherwise (i.e., if $\kappa > 0$), we solve the following 0-1 Quadratic Knapsack Problem (0-1 QKP) to find the set S':

$$\max \quad \sum_{i\in S}(\overline{w}_i + \overline{z}(i : V^+ \setminus S))\mu_i + \sum_{i\in S}\sum_{j\in S\setminus\{i\}} \overline{z}_{ij}\mu_i\mu_j \tag{39}$$

subject to

$$\sum_{i\in S} d_i\mu_i \leq \kappa \tag{40}$$

$$\mu_i \in \{0,1\} \qquad\qquad i \in S. \tag{41}$$

If we denote the optimal solution of (39)–(41) by $\{\overline{\mu_i} : i \in S\}$, we make $S' = \{i \in S : \overline{\mu_i} = 1\}$, which corresponds to the set S' that induces the greatest violation of (26) for the given candidate set S. Unfortunately, solving this 0-1 QKP is $\mathcal{NP}$-hard in the strong sense [12]. However, any feasible solution of (40)–(41) can be used to construct a valid inequality (26). We make use of a dynamic programming heuristic proposed by [24] to find a good solution of problem (39)–(41). The heuristic uses a similar recursion to that of the classical 0-1 (Linear) Knapsack Problem (0-1 KP) but with the observation that the Bellman optimality principle does not hold anymore. The method thus yields lower bounds for the 0-1 QKP. The authors report, however, that their heuristic finds the optimal solutions in a 100% of the problems tested when combined with a simple local search heuristic due to [27].

The separation problem for the q-capacity cuts (27) and their lifted version (28) is done in an analogous fashion as for inequalities (25)–(26). For the same candidate sets S, we first check whether $r(S) \leq |S| - 1$ or not. If $r(s) = |S|$ we then simply check for the violation of the q-capacity cut (27) in which case we add it to problem (1)–(14). If $r(S) \leq |S| - 1$, we first detect a subset $T \subset S$ such that $r(T) \leq |T| - 1$. Ideally, the set T should be as small as possible. We then solve the following 0-1 QKP involving customers in $W = S \setminus T$:

$$\max \quad \sum_{i \in W} (\overline{w}_i + \overline{z}(i : V^+ \setminus S))\mu_i + \sum_{i \in W} \sum_{j \in W \setminus \{i\}} \overline{z}_{ij} \mu_i \mu_j + \sum_{i \in W} \sum_{j,k \in V \setminus S, j \leq k} \overline{y}_{jik} \tag{42}$$

subject to

$$\sum_{i \in W} d_i \mu_i \leq \kappa \tag{43}$$

$$\mu_i \in \{0, 1\} \qquad i \in W. \tag{44}$$

The set S' corresponds to the customers $i \in W$ such that $\mu_i = 1$ in the solution of the above problem. Because $r(T) \leq |T| - 1$ then the set S' is guaranteed to satisfy $\rho(S \setminus S') = \rho(S)$. To select a good set T, we first sort the customers in S in non-decreasing order of $\{\overline{w}_i + \overline{z}(i : V^+ \setminus \{i\}) + \sum_{j,k \in V \setminus S, j \leq k} \overline{y}_{jik} : i \in S\}$. Then, we set $T = \emptyset$ and iteratively enlarge it by adding the next customer in the sorted list. We stop when the resulting set T satisfies the property $r(T) \leq |T| - 1$.

The separation of the comb inequalities (30) and framed capacity inequalities (31) is performed by exploiting the separation algorithms proposed by [38] for the CVRP. The algorithms developed by [38] will return the associated strengthened comb inequalities and framed capacity inequalities corresponding to inequalities (30) and (31) in which the coefficients of variables y are all equal to zero. Each of these cuts is then lifted by considering the appropriate y variables and tested for violation before being added to problem (1)–(14). Note that we have modified the code of [38] to return cuts that may not be violated by at most $\epsilon > 0$ in the hope that they become violated after the liftings.

The separation of the generalized large multistar inequalities (32) is performed in two sequential steps. In the first step, we solve the separation problem for the classic generalized large multistar inequalities for the CVRP (33). The separation problem for these inequalities can be solved in polynomial time using a maximum-flow algorithm [10]. The solution of this maximum-flow problem will return a set S that maximizes the violation of the inequality (33). If the violation is negative, we perform an additional checking to test if the lifted inequality (32) is violated before adding it to problem (1)–(14).

The separation problem for the generalized $2k$-q-edges hypotour inequalities (35) is performed in four sequential steps, as follows. First, we detect candidate sets S by using a shrinking routine available in the CVRPSEP package [37] to find sets S such that $\overline{\xi}(\delta(S))$ is close to $2k = 2r(S)$. Second, for each candidate set S, we consider all possible sets of q-edges $H \subset \delta^{ioo+}(S)$ of cardinality $2k$ such that $2k\overline{y}(H) > (2k-1)[\overline{\xi}(\delta(S)) - 2\overline{y}(\delta^{ioi+}(S))]$. Third, for every candidate set S and q-edge set H, we consider the associated set F and let $F_0 = \{e \in F : \overline{x}_e = 0\}$. We then execute the algorithm of [46] to find the $2k$ customer-disjoint paths $\{P_l : 1 \leq l \leq 2k\}$ with minimum accumulated length in $\widehat{G}$. If their lengths $\{\omega(P_l) : 1 \leq l \leq 2k\}$ are such that $\sum_{1 \leq l \leq 2k} \omega(P_l) > 2(kQ - d(S) - \sum_{1 \leq l \leq 2k} d_{j_l})$ then we have detected a violated valid inequality. Finally, and before adding the newly found inequality, we attempt to reduce the size of the set F_0, as follows. We first impose $F'_0 = \emptyset$, and check if the resulting inequality, with F'_0 instead of F_0, is valid. If not, we enlarge F'_0 by adding the edges $e \in F_0$ used in the shortest paths, and re-check the validity of the new cut. This process stops when the resulting inequality is valid. Note that this separation routine is very sensitive to the value of k as the number of possible sets H may grow as $O(|\delta^{ioo+}(S)|^{2k})$. Because of that, we restrict our search to sets S such that $r(S) = 1, 2, 3$. Moreover, for the construction of the candidate sets H, we discard *a priori* most of the q-edges in $\delta^{ioo+}(S)$ by keeping only those taking the ρ largest values of $\overline{y}$ (we let $\rho = 15$ if $k = 1$, $\rho = 10$ if $k = 2$ and $\rho = 7$ if $k = 3$).

The separation problem for the generalized $2k$-edges hypotour inequalities (36) is also performed in four sequential steps in an analogous way to that of inequalities (35). First, we detect candidate sets S such that $\overline{\xi}(\delta(S))$ is close to $2k = 2r(S)$ using the same shrinking routine as before. Second, for each candidate set S, we consider all possible sets of edges $H \subset \delta^+(S)$ of cardinality $2k$ such that $2k\overline{\xi}(H) > (2k-1)\overline{\xi}(\delta(S))$. Third, for every candidate sets S and H, we build the associated set F and consider the set $F_0 = \{e \in F : \overline{x}_e = 0\}$.

We then execute the algorithm of [46] to find the $2k$ customer-disjoint paths $\{P_l : 1 \leq l \leq 2k\}$ with minimum accumulated length in $\widehat{G}$. If their lengths $\{\omega(P_l) : 1 \leq l \leq 2k\}$ are such that $\sum_{1\leq l\leq 2k} \omega(P_l) > 2(kQ - d(S))$ then we have detected a violated valid cut. In the fourth and last stage of the algorithm, we tighten the set F_0 by initially setting $F_0' = \emptyset$ and by iteratively enlarging it in an analogous manner as done for the inequalities (35). For the same reason as before, we also restrict our search to sets S such that $r(S) = 1, 2, 3$ and use the same values of ρ to restrict the search of candidate sets H.

Remark 5 Our separation of the two classes of hypotour inequalities is more general than the one proposed by [38] for the CVRP. In fact, while their algorithm relies on the solution of an assignment problem, it cannot directly handle sets S with values of $r(S)$ strictly greater than 1. Our separation routines, on the other hand, rely on Suurballe's algorithm [46] for finding $2r(S)$ node-disjoint paths in an auxiliary graph, and can thus handle sets S with arbitrary values of $r(S)$. Note also that the separation algorithms for these two classes of valid inequalities run in polynomial-time for fixed values of $r(S)$.

The separation problem for the blossom inequalities (37) is performed using a polynomial-time algorithm developed by [36]. The algorithm starts by constructing the Gomory-Hu tree of minimum cuts in the graph $G = (V, E)$ [29]. With each edge $e \in E$ we associate a capacity $\omega_e = \min\{1 - \overline{x}_e, \overline{x}_e\}$. For each possible cut $\delta(W)$ in the cut-tree we find the associated edge set $F \subseteq \delta(W)$ containing, in principle, all edges $e \in \delta(W)$ such that $\overline{x}_e \geq 0.5$. If $|F|$ is even, then F is either enlarged or reduced by considering the edge $f \in \delta(W)$ that minimizes the quantity $\max\{1 - \overline{x}_e, \overline{x}_e\} - \min\{1 - \overline{x}_e, \overline{x}_e\}$ to then replace F by $F' = F \triangle \{f\}$ (the operator $\triangle$ denotes the symmetric difference between sets). If the resulting candidate sets W and F (or F') define a violated blossom inequality (37) we add it to problem (1)–(14).

6. The branch-and-cut algorithm

In this section we present in detail the exact solver used to solve program (1)–(14), using the branch-and-cut paradigm. Initially, we consider a relaxation of (1)–(14) in which the capacity cuts (4) and the integrality constraints (10)–(12) are relaxed. The solution of this problem yields a lower bound of the QCVRP that can be used as follows: If the associated fractional solution of this problem $\overline{X} = (\overline{\xi}, \overline{x}, \overline{y}, \overline{w}, \overline{z})$ is integer, then it represents either a feasible solution of the problem, that in the first iteration also corresponds to the optimal solution of (1)–(14), or a solution violating a capacity cut that can be detected by simple inspection by following the edges used in the candidate integer solution. If $\overline{X}$ is not integer, then we have two options: we can either look for valid inequalities that may be violated by $\overline{X}$ and strengthen the problem; or we may decide to partially break the fractionality of the solution by creating two new problems, each of which explores disjoint parts of the feasible space. Each of these subproblems is then solved using the same strategy, in a recursive manner.

There are many possible branch-and-cut algorithms depending on the strategies chosen to add valid inequalities and branch. The algorithm used in this paper is the result of some preliminary experiments, and we provide the details in the next paragraphs.

The separation strategy is as follows. First, we have found beneficial to separate all classes of valid inequalities only at the root node. Further in the branching tree, we only separate capacity cuts (25)–(26) and (28). As for the separation strategy at the root node, we first sort the inequalities in the following order: capacity cuts (25)–(26) and (28); triangle inequalities (15); conflicting edges inequalities (20)–(20); generalized large multistar inequalities (32); odd-set inequalities (19); comb inequalities (30); blossom inequalities (37); framed capacity inequalities (31); hypotour inequalities (35)–(36). We also consider the so-called homogeneous multistar inequalities (HMSI) as presented in [38], which are separated using the routines available in the CVRPSEP package. The HMSI are considered right after the separation of the blossom inequalities. This order is motivated by two main considerations: the strength of the inequalities and the efficiency of the separation algorithms. We have found that this order provides a positive balance between these two criteria. As soon as our separation routines detect a violated capacity cut of any kind, the problem is immediately re-optimized, otherwise all the other separation routines are performed in sequential order. By doing this, we avoid performing an excessive number of re-optimizations, each of which can be expensive given the cubic number of variables. When we are not able to detect any further violated inequalities, we branch.

The branching strategy is as follows. We first try to branch on cutsets. If we cannot detect any cutset of fractional value, we branch on variables x, z and w, in that order. To branch on cutsets, we use the following trick already used in successful implementations of branch-and-cut algorithms for the CLRP [9, 16]. At the root node, each lifted capacity cut (25)–(26) is added as an equality constraint by adding a slack variable s with coefficient of -2 to the left-hand side of the inequality. We allow the solver to branch on these slack variables, and impose a branching priority to branch in these variables first. Preliminaty tests have helped us to restrict this trick to sets S such that $r(S) \leq 2$. Indeed, branching on small cutsets has a larger impact on either the bounds or the feasibility of the resulting children nodes. To choose which variable to select among a series of candidate variables of the same family, we use the strong branching rule implemented in CPLEX.

Note that our implementations of the separation procedures are thread-safe, which allows taking advantage of the parallel implementation of the IP solver of CPLEX. During our computational analyses we assess the efficiency of using this feature when compared to a serial implementation of the proposed branch-and-cut solver.

7. A hybrid metaheuristic

In this section we present a hybrid heuristic for the QCVRP based on the iterative execution of a Randomized Clarke and Wright Savings Algorithm (RCWSA, [14, 6]), intensified by using branching-based local search and a IP-based post-optimization method [44, 17, 43, 45]. The pseudo-code of Algorithm 1 provides a high-level description of the algorithm that better illustrates how the different components are combined. In the pseudo-code, N is a parameter that controls the number of iterations that the RCWSA is performed, that after some preliminary experiments we have fixed to $N = 300$. Also, ν and ρ are the parameters controlling the execution of the branching-based local search, and will be described later in detail, together with the other components of the heuristic.

Algorithm 1 GRASP+BBLS+IP

```
1: $\mathcal{S} \leftarrow \emptyset$
2: for all $i = 1, \ldots, N$ do
3:     $S \leftarrow RCWSA$
4:     $S \leftarrow BB\text{-}LocalSearch(S, 1, 1)$
5:     $\mathcal{S} \leftarrow \mathcal{S} \cup \{S\}$
6: end for
7: $S^* \leftarrow SolutionBlender(\mathcal{S})$
8: $S^* \leftarrow BB\text{-}LocalSearch(S^*, \rho, \nu)$
9: Return $S^*$
```

We also consider a parallel implementation of proposed heuristic at two levels. First, the `for` cycle at line 2 of the pseudo-code is decomposed along the available cores, each of which performs a portion (roughly N/C, with C being the number of available cores) of the iterations of the cycle. Once the `for` cycle is finished, the solution blender heuristic at line 7 of the pseudo-code is also parallelized by considering the parallel IP solver of CPLEX, which decomposes the branch-and-bound tree along the available cores.

7.1 Randomized Clarke and Wright Savings Algorithm (RCWSA)

The Randomized Clarke and Wright Savings Algorithm is based on the classic Clarke and Wright Savings Algoritgm (CWSA, [14]), and uses a parameter $\kappa \in \mathbb{Z}_+$ to diversify the search, thus allowing the obtention of solutions that would otherwise be forbidden for the simpler greedy version. Let us denote a route R as a tuple starting with the depot node, followed by the customer nodes in the sequence in which they are visited, and ending at the depot. In the CWSA, we initially consider a set $\mathcal{R} = \{R = (0, i, 0) : i \in V^+\}$ containing only single-customer routes. At each iteration, we consider all pairs of routes and evaluate the possible *merges* between each pair. A merge between two routes $R_1 = (0, u_1^1, u_2^1, \ldots, u_k^1, 0)$ and $R_2 = (0, u_1^2, u_2^2, \ldots, u_l^2, 0)$ corresponds to disconnecting one extremity of each route and connecting the two free nodes, thus resulting in a larger route servicing the customers served by R_1 and R_2. Note that because of the quadratic costs structure, a feasible merge may result in a route with worse cost that the sum of those of R_1 and R_2. To obtain only feasible routes, we only consider pairs of routes whose accumulated demand does not exceed the vehicle capacity. In the classic CWSA, all possible merges are inspected and among the merges with positive saving, the best one is performed. In the randomized version, we keep the κ best merges and choose one randomly. In this way, we allow sub-optimal merges in the hope that this diversification will allow inspecting other regions of the feasible space. Moreover, among the κ best we also include merges that would result in a deterioration of the solution, as this reduces the number of vehicles used. This is needed as the QCVRP instances considered in our study assume that the fleet size is tight with respect to the total customer demands. The algorithm stops when no further merges can be detected. Note if at least one merge with positive saving is available, the random choice of the merge is restricted to the ones resulting in positive savings. After some preliminary tests, we have fixed the value of κ to 4.

7.2 Branching-based local search

In this section we present what we call branching-based local search, a generalization of classical local paradigm search that uses recursion and non-optimal moves to avoid poor local optima. In classical local search, some greedy procedures based on simple moves are applied to a solution S to inspect a certain neighborhood $\mathcal{N}(S)$, with the hope of finding another solution $S' \in \mathcal{N}(S)$, of lower cost. If no such solution can be found, the method is stopped and the best solution found so far is returned.

Unlike classical local search, branching-based local search may allow moves to solutions of worse quality than the incumbent. As its name suggests, this branching-based local search is based on a partial enumeration, as follows. To fix ideas, let S^* be the incumbent solution, and let $\mathcal{N}(\cdot)$ be the functional that given a solution S, returns $\mathcal{N}(S)$ which is a certain neighborhood of S. In simple words, the functional $\mathcal{N}(\cdot)$ corresponds to the solutions that can be obtained using some simple moves on solution S. For a given solution S (not necessarily the incumbent), we pick the best ρ neighbors S' in $\mathcal{N}(S)$. For each of these solutions, we check if its cost $cost(S')$ is lower than $cost(S^*)$, in which case we update S^*. If not, we apply the same procedure in a recursive fashion to S', up to acertain depth ν which is defined a priori.

Several implementation details must be considered. First, to avoid the curse of dimensionality, the depth of the recursion ν as well as the number of possible branches ρ must be bounded. Second, as soon as an improving solution is detected, the incumbent is updated and the branch pruned. Third, to avoid cycling and the creation of repeated solutions, we globally maintain a tabu list $\mathcal{T}$ containing solutions already generated and inspected. When branching from a solution S, we let $\mathcal{C}$ be the set of the ρ best possible solutions in $\mathcal{N}(S) \setminus \mathcal{T}$.

In our implementation of the branching-based local search performed at line 8 of Algorithm 1, we consider initial values for the parameters ρ and ν of 4 and 16, respectively. Each recursive call of the procedure is done using $\rho_k = \max\{1, \lfloor \rho_{k-1}/2 \rfloor\}$ and $\nu_k = \nu_{k-1} - 1$, where k represents the depth of the recursive call. The pseudo-code in Algorithm 2 illustrates the branching-based local search framework proposed for a certain neighborhood structure $\mathcal{N}(\cdot)$. In Figure 1 we illustrate by means of an example, the branching-based local search framework applied on a solution S_1, with $\nu = 3$, and $\rho = 4$. The label on an arc (S_i, S_j) reflects the difference $cost(S_j) - cost(S_i)$. In other words, a negative label represents an improvement of the solution S_i, while a positive label a deterioration. Starting from S_1, the four best possible solutions that can be obtained by applying local search are S_2, S_3, S_4 and S_5, and none of them offers the possibility of improving S_1. In a classical local search framework, the procedure would stop and return S_1 as the best solution. However, in the proposed framework, it is still possible to improve S_1 by considering, in sequence, S_1, S_4, S_{10} and S_{18}, which leads to an improvement of 1 unit with respect to the cost of S_1.

In this article we consider five local search procedures, four of which are implemented using the branching-based local search framework. We consider swap, 2-opt, crossover, reinsert and split operators, to be detailed

Algorithm 2 *BB-LocalSearch*$(\mathcal{N}, \mathcal{T}, S, S^*, \rho, \nu)$

```
1: if $\nu \leq 0$ then
2:     return
3: end if
4: Build $\mathcal{C}$ containing the $\rho$ best solutions in $\mathcal{N}(S) \setminus \mathcal{T}$.
5: for all $S' \in \mathcal{C}$ do
6:     $\mathcal{T} \leftarrow \mathcal{T} \cup \{S'\}$.
7:     if $cost(S') < cost(S^*)$ then
8:         $S^* \leftarrow S'$.
9:         return
10:    else
11:        $\rho' \leftarrow \max\{1, \lfloor \rho/2 \rfloor\}$.
12:        $\nu' \leftarrow \nu - 1$.
13:        return $BB\text{-}LocalSearch(\mathcal{N}, \mathcal{T}, S', S^*, \rho', \nu')$.
14:    end if
15: end for
```

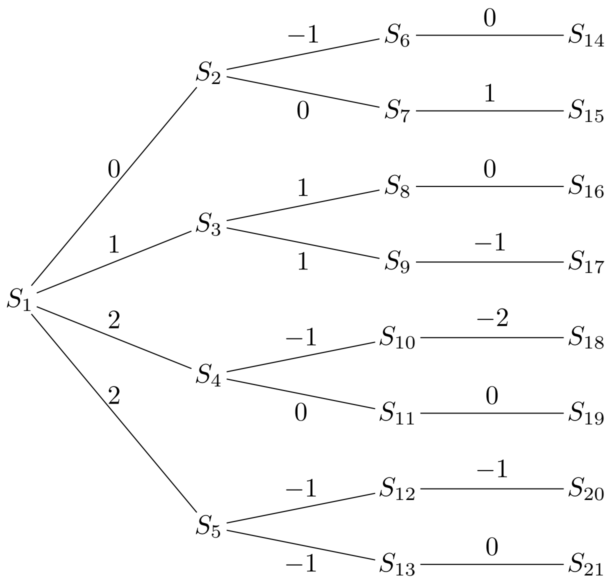

Figure 1: Example of branching-based local search with $\nu = 3$, $\rho = 4$

in the next paragraphs. The split operator is not subject to the branching-based local search because of its higher computational complexity. Also, as the pseudo-code of Algorithm 1 suggests, this branching-based local search is only applied to the solution resulting from the solution blender procedure, while the classical local search procedures corresponding to $\rho = \nu = 1$ are performed inside of the GRASP.

A **SWAP** procedure considers pairs of routes in S, namely R_1 and R_2 (they may be the same). For every customer $u \in R_1$, $v \in R_2$ we test if the routes resulting from exchanging u, v have lower total cost than the original routes, in which case the swap between u and v is performed. The evaluation of a swap move can be done in $O(1)$ time, so the complete procedure has a complexity of $O(|V^+|^2)$.

A **2-OPT** procedure considers every route $R \in S$. For every two different edges e_1, e_2 in R we evaluate the action of deleting both edges e_1 and e_2 and reconnecting the free endpoints differently. If the resulting route has smaller cost than that of R the move is performed. Each move can be evaluated in time $O(1)$, so the complete procedure takes time $O(|V^+|^2)$.

A **CROSSOVER** procedure considers two different routes R_1 and R_2 in S, servicing two customers or more each. For each pair of two consecutive customers nodes in each route, namely u_i, u_{i+1} and v_j, v_{j+1}, we consider the routes resulting from: 1) connecting u_i to v_j and u_{i+1} to v_{j+1}, and 2) connecting u_i to v_{j+1} and u_{i+1} to v_j. Each of these two possible actions can be evaluated in $O(1)$ time, and so the complete crossover procedure runs in time $O(|V^+|^2)$.

A **REINSERT** procedure considers a pair of routes R_1, R_2 (that may be the same) and a customer $v \in R_1$. Customer v is removed rom R_1 and inserted back in R_2, in the best possible position. The evaluation of a reinsert move can be done in $O(1)$ so the complete procedure runs in time $O(|V^+|^2)$.

A **SPLIT** procedure is performed on the complete solution. We start by constructing a giant TSP tour, as follows. We consider one route at a time, in sequence, and reconnect the last customer of the last route already appended to the giant TSP tour with the first or last customer (depending on which one is closer) of the next route to be appended. The giant TSP tour is denoted by $\mathcal{T} = \{u_1, \ldots, u_n, u_{n+1} = u_1\}$, with u_1 being a node arbitrarily chosen to denote the beginning of the giant TSP tour. We consider a digraph $G = (U, A)$ with $U = V^+$ and $A = \{(u_i, u_j) : 1 \leq i < j \leq n+1, \sum_{i \leq t < j} d_t \leq Q\}$. With each arc $a = (u_i, u_j)$ we denote by $\widehat{c}_a$ the cost of the following route. Consider the subtour caracterized by the sequence $(u_i, u_{i+1}, \ldots, u_{j-1}, u_i)$. Each pair of two consecutive nodes in this subtour, namely (u_t, u_{t+1}), could be reconnected to the depot to create a new route. We choose the best possible route that can be obtained by this procedure. This is called also the best rotation as the subtour is rotated until obtaining the best possible route. By running a shortest path algorithm from u_1 to u_{n+1} on the resulting network, one can determine the best possible way of splitting the giant TSP tour on several routes, each of which is determined by the use of an arc in a in the shortest path, that implicitely considers the possible rotations into account. Note that this procedure implictely assumes that the first route begins in node u_1, so the best possible split along the giant TSP toure can be found by running $|V^+|$ shortest path problems, one for every different starting node in the giant TSP tour. This procedure has been adapted from the one proposed by [47] for solving vehicle routing problems with multiple depots and linear costs, which is a generalization of an earlier method proposed by [8]. Our implementation of the split procedure runs in time $O(|V^+|^3)$. Indeed, for every possible rotation of the giant TSP toure (i.e., for every possible starting node u_1), the costs $\widehat{c}_a$ on the arcs can be computed in time $O(|V^+|^2)$ using an iterative algorithm, which is the same complexity as that of running Dijkstra's algorithm [20].

7.3 Solution blender

The solution blender is a heuristic based on the solution of an integer-linear program, and is similar to some post-optimization procedures found in the literature [44, 17, 43, 45]. Note that, as in the QCVRP only the cost matrix is quadratic, and moreover because the cost of a route is independent from the other routes, the QCVRP can be formulated as a linear set-partitioning problem. Let $\mathcal{R}$ denote the set of all feasible routes, and for every route $r \in \mathcal{R}$, we let c_r be the quadratic routing cost associated. Also, for every route $r \in \mathcal{R}$ and customer $i \in V^+$, we let a_{ir} be a binary constant equal to 1 iff route r visits customer i. Finally, for every route $r \in \mathcal{R}$ we let θ_r be a binary variable equal to 1 iff route r is used in the optimal solution. The QCVRP can be formulated as follows:

$$\min \sum_{r \in \mathcal{R}} c_r \theta_r \tag{45}$$

$$\text{subject to} \tag{46}$$

$$\sum_{r \in \mathcal{R}} a_{ir} \theta_r = 1 \qquad i \in V^+ \tag{47}$$

$$\sum_{r \in \mathcal{R}} \theta_r = K \tag{48}$$

$$\theta_r \in \{0, 1\} \qquad r \in \mathcal{R} \tag{49}$$

Unfortunately, solving this problem explicitely may be prohibitive as the size of the set $\mathcal{R}$ grows exponentially with $|V|$. One possible way of tackling this problem is column generation, in which the linear relaxation of this problem is solved for a small subset of $\mathcal{R}$, dynamically enlarged as one detects routes of negative reduced cost. This approach has two main drawbacks. On the one hand, it can be very time consuming as the solution of the pricing subproblem for generating feasible routes is strongly $\mathcal{NP}$-hard. On the other hand, it strongly relies on the quality of the lower bound, as otherwise the reduced costs have no way of guiding the solver to find the good routes. To overcome these issues, several authors have proposed heuristic ways of solving (45)–(49) by considering a predefined set of columns in advance without performing column generation [see, for instance 44, 17, 43, 45]. In this article we follow the same approach. For a given set of solutions $\mathcal{S}$, we consider every solution $S \in \mathcal{S}$. For each route $r \in S$ we add the corresponding variable θ_r to problem (45)–(49) (repeated routes are added only once), and solve it restricted to this set of routes. This procedure will likely combine routes belonging to different solutions (thus the name of *solution blender*) to potentially find another solution of strictly better quality.

7.4 Implementation issues

Our method, although simple, includes several state-of-the-art sub-routines as the improved split procedure or the solution blender, and a novel branching-based local search framework. Its simplicity, however, also has as consequence some drawbacks, mainly related to the feasibility of the solutions found.

In some cases, the RCWSA may not produce a feasible solution due to the tight vehicle capacities, thus producing solutions that exceed the fleet size from which we cannot detect further feasible merges. In this case, we perform the following procedure. We execute a Bin Packing exact algorithm to produce a number of clusters that does not exceed the fleet size. In this clusterization, only the feasibility aspect is taken into account, thus producing solutions of poor quality in most cases. These clusters are improved in a heuristic fashion by performing swaps of customers from any two different clusters. The swap may include two, three or four customers. Note that although the theoretical complexity of this procedure may be extremely high ($\mathcal{O}(n^4)$), the tight capacities allow us to discard most swaps, which results in an efficient procedure in practice. To drive the algorithm towards building compact clusters, we define a fitness function for each cluster equal to the sum of the q-edges of every three different nodes in the cluster. The swap operator uses this fitness function to exchange customers only if this exchange produces a solution of better total fitness.

Another source of conflict to finding feasible solutions in terms of the number of vehicles is the fact that the triangle property does not always hold for the q-edges linked to the depot. As a consequence, the split procedure may tend to increase the number of vehicles even if less vehicles are required to split the giant TSP tour. To overcome this issue, we simply consider an additional – large – fixed cost that we addition to every arc $a \in A$ in the split network, thus driving the algorithm towards minimizing the number of vehicles needed to split the giant TSP tour. Among all such possible splits, the one that minimizes the routing costs will always be found by the algorithm. Even if the split procedure induces a deterioration of the objective function, it is still performed provided that it reduces the number of vehicles used.

8. Computational experience

In this section we present computational experiments to assess the efficiency of the modeling and solution approaches proposed in this article. This section is subdivided in two parts. First, we describe four families of newly generated instances that have been derived from some classical datasets for the CVRP. Second, we present detailed computational results of the two algorithms presented on seven sets of instances, including the newly generated instances plus some QSTSP instances that have been modified slightly to replace one of the customer nodes by a depot, on each instance. The modification consists on considering the first node as the depot.

8.1 Newly generated Angle-CVRP and CVRP-RC instances

We have adapted some classical instances from the CVRP to include angle penalties and reload costs. We consider the sets A and B availables at `http://neo.lcc.uma.es/vrp`.

The conversion of a CVRP instance to an instance of Angle-CVRP is as follows. Let $(\gamma_{ij})_{\{i,j\}\in E}$ be the original routing costs. For every vertex $i \in V$, let p_i be the position of node i on the plane. For every edge

$\{i,j\} \in E$ we let $\overrightarrow{p_ip_j}$ be the vector in the plane with tail in p_i and head in p_j. We also let $\overrightarrow{p_jp_i} = -\overrightarrow{p_ip_j}$. In addition, $< \cdot, \cdot >$ and $\| \cdot \|$ denote the inner product of two vectors and the norm of a vector, respectively. For every q-edge $(\{i,k\},j) \in E^q$, the angle α_{ijk} is defined as

$$\alpha_{ijk} = \arccos\left(\frac{< \overrightarrow{p_jp_i}, \overrightarrow{p_jp_k} >}{\|\overrightarrow{p_jp_i}\|\|\overrightarrow{p_jp_k}\|}\right). \tag{50}$$

We let $\lambda \geq 0$ be a parameter representing the penalty incurred when the angle α_{ijk} is lower than π. The routing cost of a q-edge $(\{i,k\},j) \in E^q$ is computed as follows:

$$c_{ijk} = \begin{cases} \frac{1}{2}\lfloor(2\gamma_{ij} + \gamma_{jk})(1 + \lambda\cos(\alpha_{ijk}/2)) + 0.5\rfloor & \text{if } i = 0, j, k \in V^+. \\ \frac{1}{2}\lfloor(\gamma_{ij} + \gamma_{jk})(1 + \lambda\cos(\alpha_{ijk}/2)) + 0.5\rfloor & \text{if } i, j, k \in V^+. \end{cases} \tag{51}$$

Also, for the single-customer routes, the routing costs $(c_{0i})_{i\in V^+}$ are computed as $2c_{0i} = \lfloor 2\gamma_{0i}(1+\lambda)+0.5\rfloor$ for every $i \in V^+$. Note that when $\lambda = 0$, the routing costs correspond to those of the original CVRP. For our experiments we test three different values of λ, namely $\lambda \in \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}\}$.

To convert a CVRP instance into a CVRP-RC instance, we first consider a random σ-coloring of the edges, namely a function $col : E \longrightarrow \{1\ldots,\sigma\}$, and a penalty α. The routing cost of a q-edge $(\{i,k\},j)$ is modified according to the difference of colors between the edges involved. Namely, if we denote $\mu(i,j,k) = |col(\{i,j\}) - col(\{j,k\})|$, the new routing costs are as follows:

$$c_{ijk} = \begin{cases} \frac{1}{2}\lfloor(2\gamma_{ij} + \gamma_{jk})(1 + \alpha\mu(i,j,k)) + 0.5\rfloor & \text{if } i = 0, j, k \in V^+. \\ \frac{1}{2}\lfloor(\gamma_{ij} + \gamma_{jk})(1 + \alpha\mu(i,j,k)) + 0.5\rfloor & \text{if } i, j, k \in V^+. \end{cases} \tag{52}$$

For the single-customer routes, the routing costs of the edges $\{0,i\}$ are not modified. We have built 9 subsets of instances, each of which uses a different combination of values for σ and α, namely $\sigma \in \{5, 10, 20\}$ and $\alpha \in \{0.1, 0.2, 0.3\}$.

All newly generated instances are available at http://claudio.contardo.org.

8.2 Computational results

We have tested the two algorithms presented in this article on all instances considered in our study. The number of instances is large (1,369 instances in total) and so the results presented are aggregate for each set of instances, the detailed results being available at the authors' websites. The machine used to run the algorithms is an Intel Xeon E5462 with two quad-core CPUs at 2.8 Ghz, with 16 GB of total available RAM and running the Linux Operating System. The algorithms have been coded in C++ and compiled using the GNU g++ compiler version 4.8. We consider sequential and parallel implementations for both algorithms. In particular, the parallel implementations use 4 cores.

In Table 1 we report a summary of the results obtained with the parallel implementation of the branch-and-cut algorithm executed for a maximum time of two hours. Under column *Set* we aggregate the instances of the same set. The next three columns are used to report the summary results when solving the linear relaxation of problem (1)–(14) strengthened by all valid inequalities. Under column labeled gap_0 we report the average gaps obtained at the end of the linear relaxation. This gap is relative to the best solution found by both algorithms. Namely, if the best solution found has value z_{ub} and the lower bound at the linear relaxation is z_{lb} the relative gap is $\frac{z_{ub}-z_{lb}}{z_{ub}} \times 100$. Under columns labeled *cpu* and *wall* we report the average CPU times and wall times (in seconds) spent by the algorithm to solve the linear relaxation. The next five columns are used to report the aggregate results of the branch-and-cut algorithm. In column gap_f we report the average final relative gap with respect to the best solution found. In column gap_h we compute the average gap of the best primal solution found by the exact method relative to the best solution found by the heuristic. Note that the exact method is warm-started with the best solution found with the heuristic and so this value is expected to be smaller than or equal to zero. In column labeled *nodes* we report the average number of nodes inspected by the branch-and-cut solver. In columns *cpu* and *wall* we report the average CPU times and wall times (in seconds) spent by the exact solver. In the final two columns we report the summary results of the exact method. Under column labeled *total* we report the total number of instances included for each set. Finally, under column labeled *opt* we report the number of instances that have been solved to optimality by our exact solver.

Table 1: Parallel branch-and-cut algorithm with 4 cores

Set	Root Node			Branch-and-cut					Summary	
	gap_0 (%)	cpu (s)	wall (s)	gap_f (%)	gap_h (%)	nodes	cpu (s)	wall (s)	total	opt
angle	2.193	81.45	81.48	0.420	-0.063	1734.0	6790.03	2579.12	150	103
rel-5	3.292	152.09	152.12	1.236	-0.053	1616.1	10387.21	3802.49	150	76
rel-10	5.129	134.99	135.01	2.593	-0.090	2224.9	13321.02	4825.01	150	56
rel-20	8.694	113.58	113.60	5.638	-0.186	3092.6	16324.77	5901.36	150	32
winkel	3.649	0.07	0.07	0.000	-0.013	27.9	0.80	0.47	210	210
rel	30.105	0.05	0.05	0.000	-9.826	1142.0	142.34	48.81	349	349
r3	19.118	0.06	0.07	0.000	-8.253	244.8	9.30	3.34	210	210
									1,369	1,036

From this table we can raise several observations. First, instances with angular penalties (sets *angle* and *winkel*) appear to be among the easiest to solve. Indeed, the corresponding gaps at the linear relaxation are in average the tightest. Second, instances with reload costs (*rel-5*, *rel-10*, *rel-20* and *rel*) are much more difficult to solve. Moreover, their difficulty increases with the number of colors. Next, random instances (set *r3*) provide weak lower bounds at the linear relaxation but the exact method is robust enough to solve all of them to optimality in very short times. Finally, we can observe that the heuristic performs very well on all sets of instances except for sets *rel* and *r3* of instances derived from the QSTSP. This suggests that the heuristic does not exploit the specific QSTSP structure properly, and so the need to develop in the future heuristics especially tailored for this class of problems.

In Table 2 we present aggregate results for the parallel heuristic. In this table, column labeled *Set* is used to aggregate the instances belonging to the same family. Under columns *cpu* and *wall* we report the average CPU times and wall times (in seconds) spent by the heuristic. Under column labeled gap_{best} we report the average best gap (in %) obtained when running the heuristic 5 times for each instance. This gap is relative to the best solution found. If the best solution found has value z^* and the best cost found by the heuristic on the five runs is z, the gap is computed as $\frac{z-z^*}{z} \times 100$. Under column gap_{avg} we compute the average gaps with respect to the best solution found. This gap differs from the previous in which it is computed using the average cost produced after five runs of the heuristic rather than using the best one. Under column *opt* we report the number of instances for which the heuristic found the optimal solution proved with the branch-and-cut. Note that this does not include instances for which the exact method did not prove optimality. Under column labeled *best* we report the instances for which the heuristic produced the best solution, i.e. those for which the branch-and-cut solver was not capable of improving. Finally, under column labeled *total* we report the total number of instances for each set.

Table 2: Parallel heuristic with 4 cores

Set	cpu (s)	wall (s)	gap_{best} (%)	gap_{avg} (%)	opt	best	total
ang	20.6	6.4	0.10	0.31	76	110	150
rel-5	17.2	5.7	0.17	0.63	57	98	150
rel-10	24.2	11.5	0.34	1.17	35	85	150
rel-20	47.8	33.8	0.73	2.38	10	68	150
winkel	0.5	0.2	0.06	0.23	199	199	210
rel	0.3	0.1	12.89	20.60	214	214	349
r3	0.5	0.3	10.08	15.72	110	110	210
					701	884	1,369

The observations that we can raise from this table are surprisingly very similar to those raised from the previous table. Indeed, we can observe that once again, instances with angle penalties (sets *ang* and *winkel*) are among the easiest to solve, whereas sets *rel* and *r3* present the opposite behaviour. Moreover, the instances with reload costs are very sensitive to the number of colors, being the problems with many colors harder than those with few colors. With respect to the times, the heuristic is much faster than the branch-and-cut algorithm.

In Table 3 we present an aggregate comparison between the parallel and sequential implementations of both algorithms. For the branch-and-cut method, the maximum wall times are the same for both implementations, so we report the final gap (under columns labeled gap_f) and the number of instances solved to optimality (under columns labeled *opt*) for the sequential and parallel implementations. For the heuristic method, the number of iterations of the GRASP algorithm is the same for the sequential and parallel implementations, so we report the total wall times (under columns labeled *wall*) spent for both of them. From this table we can observe that the parallel implementations of both solvers take advantage of the additional available resources.

Table 3: Sequential vs. parallel implementations of the solution methods

Set	Branch-and-cut				Heuristic		Total
	Sequential		Parallel		Sequential	Parallel	
	gap_f (%)	opt	gap_f (%)	opt	wall (s)	wall (s)	
ang	0.495	102	**0.420**	**103**	15.9	**6.4**	150
rel-5	1.372	73	**1.236**	**76**	13.8	**5.7**	150
rel-10	2.839	50	**2.593**	**56**	18.6	**11.5**	150
rel-20	6.010	26	**5.638**	**32**	39.6	**33.8**	150
winkel	0.000	210	0.000	210	0.3	**0.2**	210
rel	0.287	348	**0.000**	**349**	0.2	**0.1**	349
r3	0.000	210	0.000	210	0.3	0.3	210
		1,019		**1,036**			1,369

9. Concluding remarks

This article introduces the Quadratic Capacitated Vehicle Routing Problem (QCVRP), a combinatorial optimization problem arising in practical applications in logistics and transportation. The contributions of this article are twofold. On the one hand, we propose a mathematical formulation of the QCVRP based on q-edges, and strengthen it with the inclusion of several classes of valid inequalities. We show that these inequalities are not only valid for the QCVRP, but that in many cases they also dominate some known valid inequalities for the CVRP. We have implemented an exact solver based on the branch-and-cut paradigm, and tested it on several randomly generated instances to assess its efficiency. On the other hand, we present an efficient heuristic to find solutions of good quality in very short computing times. The heuristic uses several state-of-the-art techniques developed during the last few years which are complemented with some new contributions as the branching-based local search. We show that the modeling and solution approaches presented in this article are efficient to provide tight lower and upper bounds and optimal solutions of the QCVRP in short to moderate moderate computing times. As of potential avenues of future research, we believe that embedding some of the inequalities introduced in this paper into a column generation-based exact solver would result in a much more robust algorithm. Also, we believe that some new cutting planes could still be derived for the QCVRP. In particular, it would be interesting to derive some other classes of multistar inequalities especially tailored for the QCVRP. Finally, we believe that future heuristics should explore other paradigms as tabu search, variable neighborhood search, adaptive large neighborhood search, iterated local search or hybrid combinations of them to improve the quality of the solutions found.

References

[1] Aggarwal, A., Coppersmith, D., Khanna, S., Motwani, R., Schieber, B.: The angular-metric traveling salesman problem. SIAM Journal on Computing 29, 697–711 (2000)

[2] Amaldi, E., Galbiati, G., Maffioli, F.: On minimum reload cost paths, tours, and flows. Networks 57, 254–260 (2011)

[3] Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study. Princeton University Press, Princeton (2007)

[4] Augerat, P.: Approche polyhédrale du problème de tournées de véhicules. Ph.D. thesis, Institut National Polytechnique de Grenoble, France (1995)

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

[6] Bard, J.F., Huang, L., Jaillet, P., Dror, M.: A decomposition approach to the inventory routing problem with satellite facilities. Transportation Science 32, 189–203 (1998)

[7] Bautista, J., Fernández, E., Pereira, J.: Solving an urban waste collection problem using ants heuristics. Computers and Operations Research 35, 302–309 (2008)

[8] Beasley, J.: Route first - cluster second methods for vehicle routing. Omega 11, 403–408 (1983)

[9] Belenguer, J.M., Benavent, E., Prins, C., Prodhon, C., Wolfler-Calvo, R.: A branch-and-cut algorithm for the capacitated location routing problem. Computers & Operations Research 38, 931–941 (2011)

[10] Blasum, U., Hochstättler, W.: Application of the branch and cut method to the vehicle routing problem. Technical report, Zentrum für Angewandte Informatik Köln, Germany (2002)

[11] Bräysy, O., Martínez, E., Nagata, Y., Soler, D.: The mixed capacitated general routing problem with turn penalties. Expert Systems with Applications 38, 12,954–12,966 (2011)

[12] Caprara, A., Pisinger, D., Toth, P.: Exact solution of the quadratic knapsack problem. INFORMS Journal on Computing 11, 125–137 (1998)

[13] Chvátal, V.: On certain polytopes associated with graphs. Journal of Combinatorial Theory B 18, 138–154 (1975)

[14] Clarke, G., Wright, J.W.: Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12, 568–581 (1964)

[15] Contardo, C.: A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. Tech. Rep. 5078, Archipel-UQÀM (2012)

[16] Contardo, C., Cordeau, J.F., Gendron, B.: A computational comparison of flow formulations for the capacitated location-routing problem. Discrete Optimization (2013). Forthcoming

[17] Contardo, C., Cordeau, J.F., Gendron, B.: A GRASP + ILP-based metaheuristic for the capacitated location-routing problem. Journal of Heuristics (2013). Forthcoming

[18] Dantzig, G.B., Fulkerson, D.R., Johnson, S.M.: Solution of a large-scale traveling salesman problem. Operations Research 2, 393–410 (1954)

[19] Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. Management Science 6, 80–91 (1959)

[20] Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959)

[21] Edmonds, J.: Paths, trees and flowers. Canadian Journal of Mathematics 17, 449–467 (1965)

[22] Fischer, A., Helmberg, C.: The symmetric quadratic traveling salesman problem. Mathematical Programming (2012). DOI 10.1007/s10107-012-0568-1. Forthcoming

[23] Fisher, M.L.: Optimal solution of vehicle routing problems using minimum k-trees. Operations Research 42, 626–642 (1994)

[24] Fomeni, F.D., Letchford, A.N.: A dynamic programming heuristic for the quadratic knapsack problem. INFORMS Journal of Computing (2013). Forthcoming

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

[26] Galbiati, G., Gualandi, S., Maffioli, F.: On minimum reload cost cycle cover. Electronic Notes in Discrete Mathematics 36, 81–88 (2010)

[27] Gallo, G., Hammer, P.L., Simeone, B.: Quadratic knapsack problems. Mathematical Programming Studies 12, 132–149 (1980)

[28] Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum-flow problem. Journal of the Association for Computing Machinery 35, 921–940 (1988)

[29] Gomory, R.E., Hu, T.C.: Multi-terminal network flows. Journal of the SIAM 9, 551–570 (1961)

[30] Gourvès, L., Lyra, A., Martinhon, C., Monnot, J.: The minimum reload s-t path, trail and walk problems. Discrete Applied Mathematics 158, 1404–1417 (2010)

[31] Gouveia, L.: A result on projection for the vehicle routing problem. European Journal of Operational Research 85, 610–624 (1995)

[32] Grötschel, M., Padberg, M.W.: On the symmetric travelling salesman problem I: inequalities. Mathematical Programming 16, 265–280 (1979)

[33] Jünger, M., Thienel, S., Reinelt, G.: Provably good solutions for the traveling salesman problem. Zeitschrift für Operations Research 40, 183–217 (1994)

[34] Laporte, G., Nobert, Y.: Comb inequalities for the vehicle routing problem. Methods of Operations Research 51, 271–276 (1984)

[35] Laporte, G., Nobert, Y., Desrochers, M.: Optimal routing under capacity and distance restrictions. Operations Research 33, 1050–1073 (1985)

[36] Letchford, A.N., Reinelt, G., Theis, D.: A faster exact separation algorithm for blossom inequalities. In: D. Bienstock, G. Nemhauser (eds.) Integer Programming and Combinatorial Optimization, *Lecture Notes in Computer Science*, vol. 3064, pp. 196–205. Springer Berlin Heidelberg (2004)

[37] Lysgaard, J.: CVRPSEP: A package of separation routines for the capacitated vehicle routing problem (2003)

[38] Lysgaard, J., Letchford, A.N., Eglese, R.W.: A new branch-and-cut algorithm for the capacitated vehicle routing problem. Mathematical Programming 100, 423–445 (2004)

[39] Padberg, M.: The boolean quadric polytope: some characteristics, facets and relatives. Mathematical Programming 45, 139–172 (1989)

[40] Padberg, M.W., Hong, S.: On the symmetric travelling salesman problem: a computational study. Mathematical Programming Study 12, 78–107 (1980)

[41] Padberg, M.W., Rinaldi, G.: Facet identification for the symmetric traveling salesman polytope. Mathematical Programming 47, 219–257 (1990)

[42] Perrier, N., Langevin, A., Amaya, C.A.: Vehicle routing for urban snow plowing operations. Transportation Science 42, 44–56 (2008)

[43] Pirkwieser, S., Raidl, G.R.: Variable neighborhood search coupled with ILP-based very large-neighborhood searches for the (periodic) location-routing problem. In: M. Blesa, C. Blum, G. Raidl, A. Roli, M. Sampels (eds.) Hybrid Metaheuristics, *Lecture Notes in Computer Science*, vol. 6373, pp. 174–189 (2010)

[44] Rochat, Y., Taillard, E.D.: Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1, 147–167 (1995)

[45] Subramanian, A., Uchoa, E., Ochi, L.S.: A hybrid algorithm for a class of vehicle routing problems. Computers & Operations Research (2013). DOI 10.1016/j.cor.2013.01.013. Forthcoming

[46] Suurballe, J.W.: Disjoint paths in a network. Networks 4, 125–145 (1974)

[47] Vidal, T., Crainic, T., M., G., Prins, C.: Implicit depot assignments and rotations in vehicle routing heuristics. Tech. rep., CIRRELT 2012–60 (2012)