A Branch & Cut Algorithm for the Windy General Routing Problem and Special Cases

Angel Corberán and Isaac Plana
DEIO, Universitat de València, Dr. Moliner 50, 46100 Burjassot, Valencia, Spain

José M. Sanchis
DMA, Universidad Politécnica de Valencia, Camino de Vera s/n, 46022 Valencia, Spain

In this paper, we present an exact algorithm for the Windy General Routing Problem. This problem generalizes many important Arc Routing Problems and also has some interesting real-life applications. The Branch & Cut method presented here is based on a cutting-plane algorithm that identifies violated inequalities of several classes of facet-inducing inequalities for the corresponding polyhedron. The whole procedure has been tested over different sets of instances and is capable of solving to optimality large-size instances of several routing problems defined on undirected, mixed, and windy graphs.
 NETWORKS, Vol. 49(4), 245–257 2007

Keywords: branch & cut; arc routing; windy general routing problem; windy rural postman problem

1. INTRODUCTION

The Chinese Postman Problem (CPP) consists of finding a shortest tour (closed walk) traversing *all* the links of a given graph G. The CPP was originally proposed by Guan [23] for *undirected* graphs, where all the links are *edges* that can be traversed in both directions with the same cost. This problem is considered to be the first Arc Routing Problem to appear in the literature. The book edited by Dror [18] summarizes the state of the art and real-life applications of the Arc Routing problems. When G is a *directed* graph, where all the links are *arcs* that must be traversed in a given direction, the problem is known as the Directed Chinese Postman Problem (DCPP) and was proposed by Edmonds and Johnson [19]. In the same paper, the Mixed Chinese Postman Problem (MCPP) was introduced. This problem is defined on a *mixed* graph having edges and arcs simultaneously. While the CPP and DCPP can be solved in polynomial time, the MCPP is NP-hard [32] and it generalizes both the CPP and the DCPP. Finally, Minieka [27] proposed the Windy (Chinese) Postman Problem (WPP), which is defined on an undirected graph where the cost of traversing an edge (i, j) in a given direction, c_{ij}, can be different from the cost of traversing it in the opposite direction, c_{ji}. Brucker [5] and Guan [24] showed that the WPP is NP-hard, although it can be solvable in polynomial time if the graph is Eulerian [34], or if the two orientations of every cycle have the same cost [24]. Moreover, in [34] and [22] an integer programming formulation and a polyhedral study of the WPP are presented, as well as a cutting-plane algorithm capable of solving medium-size instances to optimality. The WPP generalizes the undirected CPP when $c_{ij} = c_{ji}$, the DCPP, since each arc (i, j) with cost c can be modeled as an edge with costs $c_{ij} = c$ and $c_{ji} = \infty$, and hence the MCPP.

On the other hand, the CPP has also been generalized in another way. Orloff [29] proposed the Rural Postman Problem (RPP) and the General Routing Problem (GRP). In the RPP, the tour does not have to traverse all the links of the graph but only those in a given subset of required links. In the GRP, the graph contains a subset of required links to be traversed and a subset of required vertices to be visited. The GRP can also be considered as a generalization of the Graphical Traveling Salesman Problem (GTSP), which consists of finding a shortest tour visiting all the vertices of a given graph G at least once. The GTSP was introduced by Cornuèjols et al. [17] and Fleischmann [21] and also studied in [28]. The RPP, GRP, and GTSP were initially defined for undirected graphs, but they can also be formulated on directed (DRPP, DGRP, DGTSP), mixed (MRPP, MGRP, MGTSP), and windy graphs (WRPP, WGRP, WGTSP). Note that the GTSP defined on a directed, mixed, or windy graph is equivalent to the Graphical Asymmetric Traveling Salesman Problem (GATSP), introduced by Chopra and Rinaldi [6].

As mentioned earlier, the CPP, RPP, and GTSP are special cases of the GRP. Furthermore, routing problems on undirected, directed, or mixed graphs can be generalized by the

Received July 2005; accepted August 2006
Correspondence to: A. Corberán; e-mail: angel.corberan@uv.es
Contract grant sponsors: Ministerio de Ciencia y Tecnología of Spain; Generalitat Valenciana
DOI 10.1002/net.20176
Published online in Wiley InterScience (www.interscience.wiley.com).

corresponding problem defined on a windy graph. Thus, the problem we deal with in this paper, the Windy General Routing Problem (WGRP), is the most general problem among those mentioned earlier and includes all the others as special cases: the CPP, RPP, GTSP, GRP, DCPP, DRPP, GATSP, DGRP, MCPP, MRPP, MGRP, WPP, and WRPP. Hence, the theoretical and computational results obtained in this paper can be applied to all these problems. In particular, the exact algorithm for the WGRP we propose here can be used to solve instances of any of these problems.

In addition, the WGRP is also interesting from a practical point of view because it is the optimization model describing some real-life situations. Besides the usual routing applications, consider for instance the case described in [3] of some climbing robots designed to inspect complex three-dimensional structures, such as bridges. They carry a limited battery, so their routes must be carefully designed in order to minimize energy consumption [2]. These robots are remotely controlled and are equipped with TV-cameras to inspect the structure in such a way that any possible damage in the bridge beams, for example, can be detected. All beams must be inspected, so they can be represented by required edges. Some special movements must also be performed by the robots in order to move from the end of one beam to the beginning of another one or to another side of the same beam. These would be modeled by nonrequired edges. And since, for example, the energy consumed by the robot is not the same if the movement is upwards or downwards, the cost of traversing each edge can be different for each direction. So the problem can be formulated as a WRPP.

In the next section, we introduce the notation that will be used in this paper and present an ILP formulation of the WGRP and the polyhedron associated with it. The section ends with a summary of the polyhedral results known in the literature. Section 3 is devoted to the separation algorithms used in the Branch & Cut (B&C) algorithm, which is described in section 4. The computational experiments performed on a wide set of instances are described in section 5, while section 6 presents our conclusions.

2. PROBLEM FORMULATION AND POLYHEDRAL RESULTS

The WGRP consists of finding a minimum cost tour traversing at least once all the required edges, $E_R \subseteq E$, and visiting at least once all the required vertices, $V_R \subseteq V$, of an undirected graph $G = (V, E)$. Associated with each edge $(i,j) \in E$ there are two non negative costs, c_{ij} and c_{ji}, corresponding to the cost of traversing it from i to j and from j to i, respectively.

For the sake of simplicity we will assume throughout the paper that $V_R = V$. This is not a loss of generality since it is easy to transform a WGRP instance not satisfying this assumption into an equivalent one which does (see for example [8] or [20]). This transformation often decreases the number of edges, although may increase it sometimes. The subgraph associated with the required vertices and edges, $G_R = (V, E_R)$, is in general nonconnected. We denote by p the number of its connected components and by $V_1, V_2, \ldots, V_p$, with $V_1 \cup \ldots \cup V_p = V$, their corresponding vertex sets, which are called R-sets. The R-sets play a very important role in the WGRP because they define subgraphs of G that must be connected by the tour. Note, for example, that if $E_R = \emptyset$, the WGRP becomes the GATSP and the R-sets are the vertices of the graph. Hence, the number of R-sets is a significant parameter concerning the difficulty of a given instance.

Given $S \subseteq V$, $\delta(S)$ denotes the set of edges with an end-point in S and the other in $V \backslash S$ and $E(S)$ is the set of edges with both end-points in S. Given $S_1, S_2 \subseteq V$, (S_1, S_2) represents the set of edges with one end-point in S_1 and the other in S_2, while $\delta_R(S)$, $E_R(S)$, $(S_1, S_2)_R$ denote the previous sets referring only to the required edges. A vertex is R-even (R-odd) if it is incident with an even (odd) number of required edges, and a subset of vertices $S \subseteq V$ is called R-even (R-odd) if it contains an even (odd) number of R-odd vertices.

Let us associate two variables x_{ij} and x_{ji} with each edge (i,j), representing the number of times edge (i,j) is traversed from i to j and from j to i, respectively. Given $F \subseteq E$, we denote by $x(F)$ the sum of all the variables associated with the edges in F, $x(F) = \sum_{(i,j)\in F}(x_{ij} + x_{ji})$, and given (S_1, S_2), we denote by $x(S_1 : S_2)$ the sum of the variables associated with the traversal from S_1 to S_2 of the edges in (S_1, S_2),

$$x(S_1 : S_2) = \sum_{i\in S_1,\, j\in S_2} x_{ij}$$

Note that $x(S_1, S_2) = x(S_1 : S_2) + x(S_2 : S_1)$.

The WGRP can be formulated as follows:

$$\text{Min} \sum_{(i,j)\in E} (c_{ij}x_{ij} + c_{ji}x_{ji})$$

$$\text{s.t} \quad x_{ij} + x_{ji} \geq 1, \quad \forall (i,j) \in E_R \tag{1}$$

$$\sum_{(i,j)\in\delta(i)} (x_{ij} - x_{ji}) = 0, \quad \forall i \in V \tag{2}$$

$$x(S : V \setminus S) \geq 1, \quad \forall S = \bigcup_{k\in Q} V_k, \quad Q \subset \{1, \ldots, p\} \tag{3}$$

$$x_{ij}, x_{ji} \geq 0, \quad \forall (i,j) \in E \tag{4}$$

$$x_{ij}, x_{ji} \text{ integer}, \quad \forall (i,j) \in E \tag{5}$$

Conditions (1) guarantee that each required edge is traversed at least once. Conditions (2) and (3) ensure that the vehicle departs from each vertex as many times as it arrives at it and that the route is connected. Note that it is enough to connect the different R-sets. Basically, this formulation is the same as the one proposed by Benavent et al. [3] for the WRPP. The only difference is that here some R-sets V_i can consist of only one vertex.

The polyhedron associated with the feasible solutions to (1) to (5), WGRP(G), has been studied in [12] and [33]. It is an unbounded polyhedron of dimension $2|E| - |V| + 1$ (due to

the equations (2)), for which many classes of facet-inducing inequalities are known. Given that WGRP(G) is not full dimensional, different inequalities can induce the same facet. Such inequalities are called equivalent and, among them, those with fewer non-zero coefficients are more appropriate for computational purposes. The non-negativity inequalities (4), which induce facets of WGRP(G), are handled implicitly by any LP solver. The $|E_R|$ constraints (1), which are also facet-inducing, and the system equations (2) are explicitly included in the LP formulation. Our B&C procedure uses separation algorithms for other classes of facet-defining inequalities, which are briefly described in what follows.

2.1. Connectivity and R-Odd Cut Inequalities

Connectivity inequalities (3) induce facets of WGRP(G) if and only if the subgraphs induced by S and $V \setminus S$ are connected. On the other hand, the R-odd cut inequalities

$$x(\delta(S)) \geq |\delta_R(S)| + 1, \quad \forall \delta(S) \text{ R-odd cutset of } G \qquad (6)$$

are also facet-inducing of WGRP(G) if and only if the subgraphs induced by S and $V \setminus S$ are connected. Equations (2) associated with all the vertices in $S \subset V$ imply that $x(S : V \setminus S) = x(V \setminus S : S)$. Then, inequalities (6) are equivalent to the following ones, which contain fewer non-zero elements:

$$x(S : V \setminus S) \geq \frac{|\delta_R(S)| + 1}{2}, \quad \forall \delta(S) \text{ R-odd cutset of } G \quad (7)$$

2.2. K-C and K-C$_{02}$ Inequalities

Standard K-C inequalities [15] are associated with a partition of the set of nodes V in $K + 1$ subsets $\{M_0, M_1, \ldots, M_{K-1}, M_K\}$, with $K \geq 3$, satisfying that each R-set V_i is contained either in $M_0 \cup M_K$ or in any set M_j, $j = 1, \ldots, K-1$, subgraphs $G(M_i)$ are connected, set (M_0, M_K) contains a positive and even number of required edges and sets (M_i, M_{i+1}) are nonempty, for $i = 0, \ldots, K-1$. The inequality, which can be represented by a graph like the one shown in Figure 1, takes the form:

$$(K-2)\, x(M_0, M_K) + \sum_{\{i,j\} \neq \{0,K\}} |i-j| x(M_i, M_j) \geq 2(K-1) + (K-2)|(M_0, M_K)_R| \quad (8)$$

K-C inequalities can be written in a sparse form as follows. Consider first the case when K is an even number. If $x(M_i : V \setminus M_i)$ is replaced by $x(V \setminus M_i : M_i)$ in (8) for $M_1, M_3, \ldots M_{K-1}$, then an equivalent inequality is obtained which, divided by 2, takes the form:

$$\frac{K-2}{2} x(M_0, M_K) + \sum_{\{i,j\} \neq \{0,K\}} (\alpha_{ij} x(M_i : M_j) + \alpha_{ji}\, x(M_j : M_i)) \geq K - 1 + \frac{K-2}{2} |(M_0, M_K)_R| \quad (9)$$

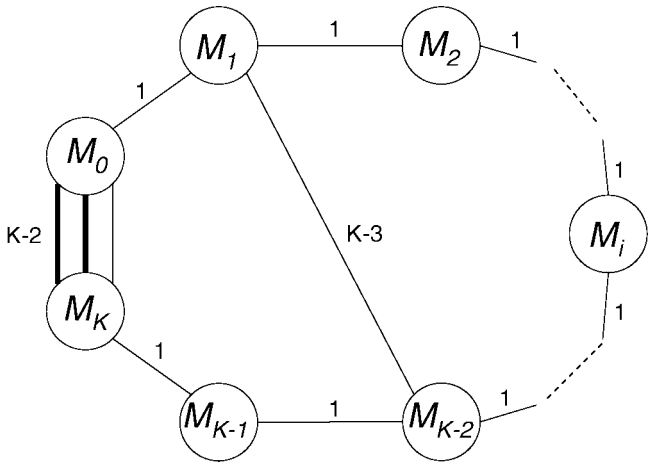

FIG. 1. Graph representing a K-C inequality.

where

$$\alpha_{ij} = \begin{cases} \dfrac{|j-i|-1}{2} & \text{if } i \text{ is odd and } j \text{ is even} \\ \dfrac{|j-i|+1}{2} & \text{if } i \text{ is even and } j \text{ is odd} \\ \dfrac{|j-i|}{2} & \text{otherwise} \end{cases}$$

When i is odd and $j = i+1$ or $j = i-1$, then $\alpha_{ij} = 0$, and the inequality has fewer non-zero coefficients. If K is odd, the original K-C inequality can be transformed in a similar way.

K-C$_{02}$ inequalities [14] are similar to the standard K-C inequalities, since they are also associated with a partition of the set of nodes V satisfying the same conditions, but have different coefficients:

$$(K-1)\, x(M_0, M_K) + \sum_{j \neq K} ((j-1) x(M_0 : M_j) + (j+1)\, x(M_j : M_0)) + \sum_{i,j \neq 0} |i-j|\, x(M_i, M_j) \geq 2(K-1) + (K-1)|(M_0, M_K)_R| \quad (10)$$

Note that the K-C$_{02}$ inequalities are inherently *asymmetric*. Like the standard K-C inequalities, K-C$_{02}$ inequalities also have an equivalent version with fewer nonzero elements. Both families of inequalities induce facets of WGRP(G) if certain conditions are satisfied [12].

2.3. Path-Bridge Inequalities

If we generalize a K-C inequality by allowing a number $p \geq 1$ of paths from M_0 to M_K such that $p + |(M_0, M_K)_R|$ is odd, we obtain a Path-Bridge (PB) inequality [26]. Figure 2 shows a graph representing a PB inequality. A PB inequality is called *regular* when all its paths have the same number of nodes. Details on these inequalities and the conditions under which they are facet-inducing for WGRP(G) are given in [12]. As with the K-C inequalities, equivalent PB inequalities with

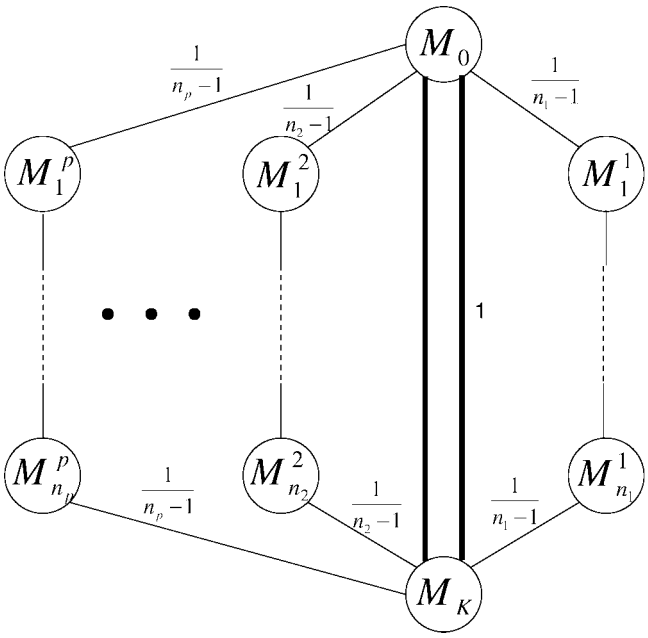

FIG. 2. Graph representing a Path-Bridge inequality.

fewer nonzero elements can be obtained. In [12] it is also shown that other related inequalities called Path-Bridge$_{02}$ are also valid and facet-inducing for WGRP(G). Nevertheless, since some of their coefficients must be obtained by sequential lifting, the PB$_{02}$ inequalities have not been used in the Branch & Cut method and are not presented here in detail.

2.4. Honeycomb Inequalities

Honeycomb (HC) inequalities [16] are also a generalization of K-C inequalities, but in a different way. In this case, the R-sets that were partitioned into M_0 and M_K are now partitioned into $L \geq 2$ subsets. HC inequalities, which are facet-inducing for the WGRP(G), are quite complicated and their details can be found in [12]. An example of a simple HC inequality is shown in Figure 3. With a similar structure, but with different coefficients from the standard HC inequalities, there is another family of facet-inducing inequalities, the HC$_{02}$ inequalities [10], [12]. Versions of the HC and HC$_{02}$ inequalities with fewer nonzero coefficients can also be obtained.

2.5. Zigzag Inequalities

In [13], a new class of facet-inducing inequalities for the WGRP and other Arc Routing Problems is presented. This class contains two different families of inequalities, the even and the odd zigzag inequalities.

Let M_1, M_2, M_3, M_4 be a partition of V such that the subgraphs $G(M_i)$ are connected and $|\delta_R(M_i)|$ is even, $i = 1, 2, 3, 4$. Let α_{ij} be the number of required edges in (M_i, M_j) and suppose that $\alpha_{12} = \alpha_{34} = \alpha_{14} = \alpha_{23} = 0$. The Even Zigzag inequality (see figure 4a) is:

$$x(\delta(M_1 \cup M_2)) + 2x(M_2 : M_1) + 2x(M_4 : M_3) \geq \alpha_{13} + \alpha_{24} + 2 \quad (11)$$

These inequalities can be written also in a sparse form. To illustrate this, consider the Even Zigzag inequality represented in Figure 4a. Owing to the symmetry equation associated with node M_1, the term $x(V \setminus M_1 : M_1)$ in the inequality can be replaced by $x(M_1 : V \setminus M_1)$ to obtain an equivalent inequality. Then, we proceed in a similar way with node M_2 and we obtain an equivalent inequality that divided by 2 has the coefficients shown in Figure 4b.

Basically, when $|\delta_R(M_i)|$ is odd, $i = 1, 2, 3, 4$, we have the Odd Zigzag inequalities, which can look similar to the Even Zigzag inequalities (Fig. 5a), but are in general rather more complicated (Fig. 5b). The inequality associated with the simple Odd Zigzag inequalities (the case in which, again, $\alpha_{12} = \alpha_{34} = \alpha_{14} = \alpha_{23} = 0$) has the same form as that of the even case (11). Since the general Odd Zigzag inequalities have not been used in our B&C algorithm, they are not presented here.

3. SEPARATION ALGORITHMS

In this section, we present the separation algorithms used in the B&C algorithm for the inequalities described in Section 2. Given an LP solution $x^* \in \mathbb{R}^{2|E|}$, we define G^* as the weighted graph induced in G by the edges (i, j) with $w_{ij} = x^*_{ij} + x^*_{ji} > 0$. Given a subset $T \subset E$, we denote $w(T) = \sum_{(i,j)\in T} w_{ij}$.

3.1. Connectivity and R-Odd Cut Separation

Connectivity inequalities can be separated exactly in polynomial time by finding a minimum weight cut in the graph obtained from G^* by shrinking each R-set into a single vertex. Note that when a subset of vertices is shrunk, each set

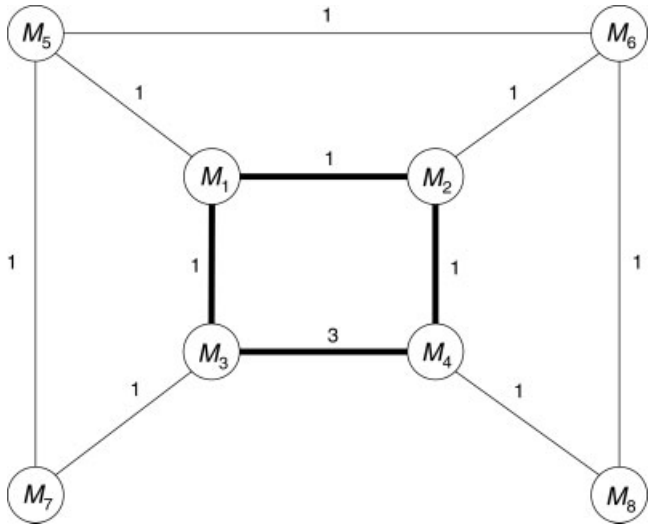

FIG. 3. Graph representing a Honeycomb inequality.

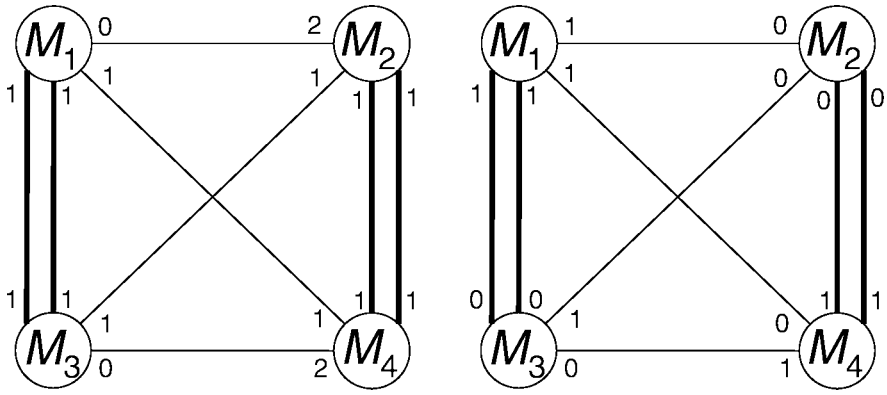

FIG. 4. Graphs representing the same even zigzag inequality in its dense and sparse form.

of parallel resulting edges, if any, are substituted by only one edge whose weight is equal to the sum of all these edge weights. Faster heuristic algorithms, based on computing the connected components induced by the edges of the shrunk graph with weight greater than ϵ, for different values of ϵ, are also used.

The exact separation of R-odd cut inequalities can also be done in polynomial time by means of the Padberg–Rao procedure [30] for finding odd cutsets of minimum weight. Again, faster heuristic algorithms have also been used. They are based on computing the connected components induced by the edges of G^* with weight $w'_{ij} > \epsilon$, for different values of ϵ, where $w'_{ij} = w_{ij} - 1$ if $(i,j) \in E_R$ and $w'_{ij} = w_{ij}$ otherwise.

3.2. K-C, K-C_{02}, PB, HC and HC_{02} Separation

It is not known whether the separation problem for all these classes of inequalities is NP-complete or not. We have implemented heuristic algorithms for the separation of standard K-C, regular PB and standard HC inequalities, which are based on those developed for the undirected GRP (see [9] for the details). The separation algorithms for the K-C_{02} and HC_{02} inequalities have to consider the asymmetry of the solution and thus differ from the previous procedures in some steps. These algorithms are similar to those presented in [10] for the Mixed GRP and details are not shown here. No algorithm for separating violated PB_{02} inequalities has been implemented due to the difficulty of obtaining all the variable coefficients in short computing times. As mentioned before, in these inequalities the coefficients associated with edges which link nodes in different paths have to be computed using sequential lifting.

3.3. Zigzag Inequalities Separation

We have designed a new heuristic separation algorithm for Even and Odd Zigzag inequalities with $\alpha_{12} = \alpha_{34} = \alpha_{14} = \alpha_{23} = 0$. It is based on the idea that solutions x^* violating such an inequality often contain an edge cutset consisting of four non-required edges whose associated variables take the value 0.5, as in the solution depicted in Figure 6a. In this example, solid lines represent required edges whose associated variables take value 1 and dotted lines represent nonrequired edges with value 0.5. We assume that the graph G has at least 2 R-sets (with more than one vertex) and that all the connectivity constraints are satisfied by x^*.

Given x^*, we assign the following capacities to the edges in graph G^*: nonrequired edges with $x^*_{ij} = 0.5$ and $x^*_{ji} = 0$ (or vice versa) are given capacity 1, while all the other edges get infinite capacity. This capacitated graph is called G_{aux} (Fig. 6b).

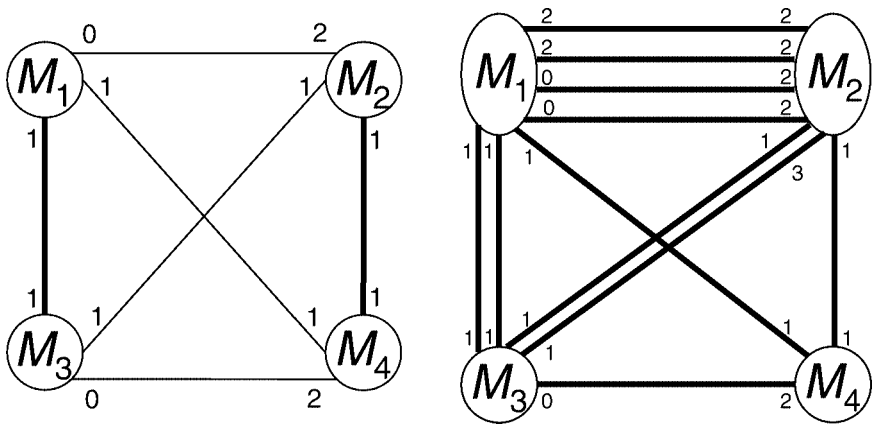

FIG. 5. Graphs representing simple and general Odd Zigzag inequalities.

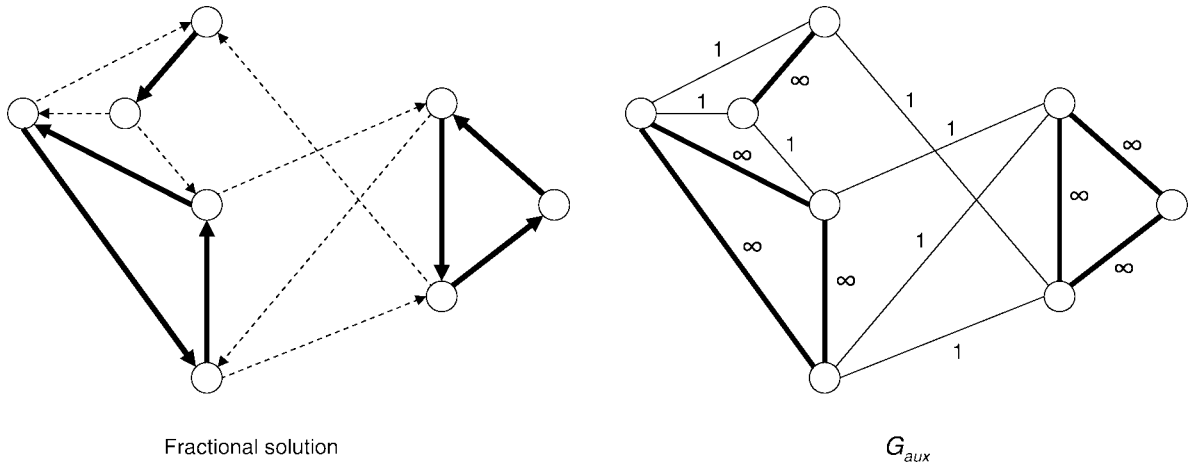

FIG. 6. Fractional solution x^* and capacitated graph G_{aux}

Let (i,j) be an edge with capacity 1 in G_{aux}. A maximum flow from i to j is computed in G_{aux}. The flow cost must be at least four, otherwise there would be a violated connectivity constraint. If the flow value is exactly 4, an edge cutset (V_1, V_2) of capacity 4 is identified (Fig. 7). By construction, this edge cutset consists of four nonrequired edges satisfying $x^*_{ij} = 0.5$ and $x^*_{ji} = 0$ with which we try to find a zigzag. If this is not possible, we try to find another edge cutset of capacity 4 by computing another maximum flow between two vertices connected by another edge of capacity 1.

Let us suppose that, as in the example shown in Figure 8, such a zigzag has been found. Then, we have four subsets with one or two nodes each defining the zigzag "corners", which will be the seeds for M_1, M_2, M_3, and M_4. Now we proceed to partition sets V_1 and V_2 into two subsets each. To do this, the graph $G_1 = (V_1, E_1)$ is constructed, where E_1 contains the edges with two end nodes in V_1 such that $w_{ij} > 0$ if (i,j) is nonrequired and $w_{ij} > 1$ if it is required. In order to get a violated Zigzag inequality, the edges in E_1 should not appear in the edge cutset dividing V_1. We now compute the connected components of G_1. If the nodes defining a "corner" of the zigzag are in the same component C, we consider the partition of V_1 defined by $M_1 = C$ and $M_3 = V_1 \setminus C$. Otherwise, if these nodes belong to two different components C_1 and C_2, we would consider $C = C_1 \cup C_2$. In the case that the other "corner" in V_1 was also in C, the procedure would stop and we would look for another edge cutset (V_1, V_2). By proceeding in a similar way, a partition of $V_2 = M_2 \cup M_4$ is obtained (Fig. 9). If $M_1, \ldots, M_4$ are all R-even (R-odd), we have found a violated Even (Odd) Zigzag inequality. Otherwise different partitions of V_1 and V_2 are considered by joining other components of G_1 and G_2.

In the implementation of our algorithm, when we construct the graph G_{aux}, we assign capacity 1 not only to the nonrequired edges with $x^*_{ij} = 0.5$ and $x^*_{ji} = 0$, but also to those with $x^*_{ij} \leq 0.75$ and $x^*_{ji} = 0$. It is easy to see that in this case the inequality obtained would also be violated.

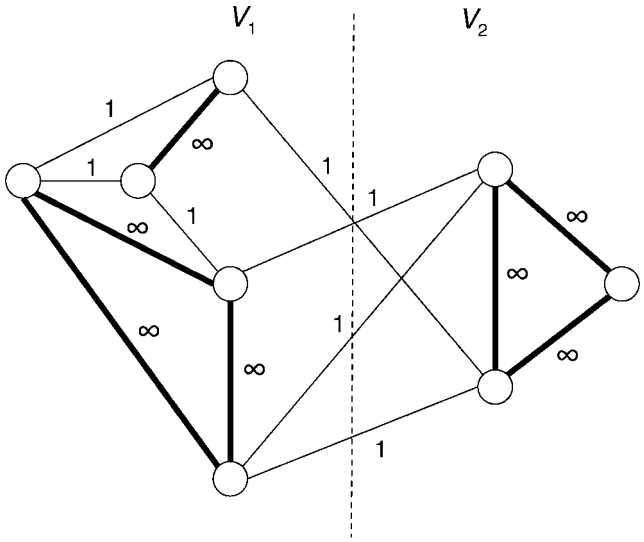

FIG. 7. Minimum capacity cutset.

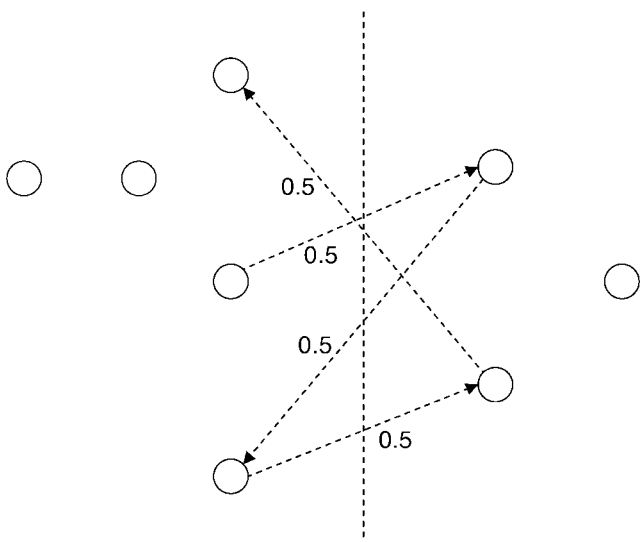

FIG. 8. Zigzag found.

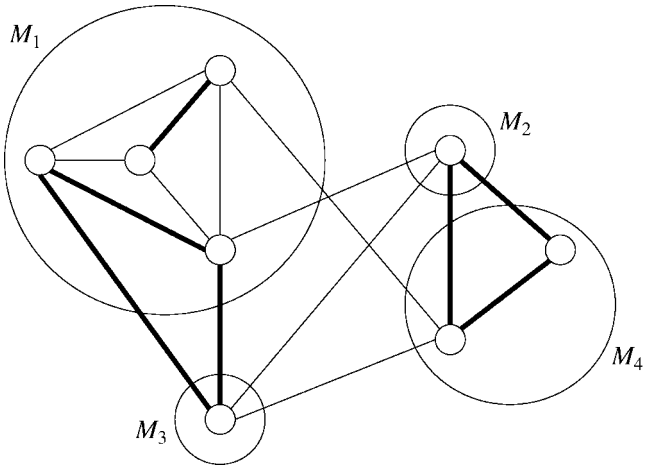

FIG. 9. Partition of V into M_1, M_2, M_3 and M_4.

4. THE BRANCH & CUT METHOD

In this section, we describe the details of our implementation of the B&C method. B&C algorithms were first introduced by Padberg and Rinaldi [31], and have shown to be among the most efficient methods for solving NP-hard problems to optimality. Basically, a B&C algorithm consists of a cutting-plane procedure working at the nodes of a Branch & Bound tree. At each node, facet-defining inequalities that are violated by the current LP solution are identified by the separation algorithms.

4.1. Initial Relaxation and Cutting Plane Algorithm

In what follows, we present the initial LP relaxation that is the input to the LP solver as well as how the separation algorithms are integrated to form the cutting plane algorithm.

Instead of using all the connectivity constraints presented in the formulation of the WGRP (an exponential number), only one connectivity constraint for each R-set is included in the first LP relaxation. Furthermore, some R-odd cut inequalities have been added, specifically those associated with R-odd vertices and with the connected components S of the subgraph of G_R induced by the R-odd vertices, if $|S|$ is odd. Note that even an integer feasible solution for this LP may not be a WGRP feasible solution, since not all the connectivity constraints are guaranteed to be satisfied.

At each iteration of the cutting plane algorithm the exact and heuristic separation algorithms are called in a specific order and a number of violated inequalities are added to the LP relaxation, which is then solved again. From previous experience and based on some tests conducted on a small set of instances, the algorithm has been implemented as follows:

1. R-odd cut and connectivity separation heuristics with $\epsilon = 0$. If no violated inequalities of each class are found, apply the heuristics with $\epsilon = 0.25$ and, if necessary, $\epsilon = 0.5$.
2. Exact connectivity separation if the corresponding heuristics failed.
3. Exact R-odd cut separation if no violated inequalities have been found so far (and only in 1 out of 2 iterations).
4. If the total number of violated inequalities found is less than 10, run heuristic algorithms for separating K-C and K-C_{02} inequalities. If no violated inequalities of any of these classes are found, execute the same heuristics with $\epsilon = 0.25$ and, if necessary, and only at the root node, with $\epsilon = 0.5$.
5. If the total number of violated inequalities found is less than 10, run algorithms for separating HC and HC_{02} inequalities.
6. If the total number of violated inequalities found is less than 10, execute the zigzag heuristic.
7. Heuristic separation for PB inequalities if the total number of violated inequalities found is less than 10.
8. If no violated inequality of any class has been found, and only once at the root node of the B&C tree, run heuristics for K-C, K-C_{02}, HC, and HC_{02} with iterative merging of adjacent R-sets.

The cutting plane procedure is applied at each node of the tree until no new violated inequalities are found or a stopping criterium, called *tailing-off*, is satisfied. In our implementation the cutting plane stops when the increase in the objective function during the last five iterations is less than 0.00005%. At the root node, this percentage has been fixed to 0.00001%.

4.2. Selection of Violated R-Odd Cut Inequalities

For large instances, the number of inequalities found during the exploration of the B&C tree is so big that, in some cases, the optimal solution cannot be found due to memory limitations, and very often computing times increase because of this problem. We have noticed that the exact separation algorithm for R-odd cut inequalities usually finds a large number of violated inequalities that are very similar to each other. Adding so many similar inequalities does not seem to prove of much value to the effectiveness of the algorithm. Therefore, only a small number of the R-odd cut inequalities identified by the exact separation algorithm is added.

Let k_i be the number of edges in the edge cutset associated with a given inequality i. Then, we define the similarity between two inequalities i and j, $\text{sim}(i,j)$, as the number of edges that their associated cutsets have in common divided by $\max\{k_i, k_j\}$. Also, the similarity between an inequality i and a subset of inequalities R, $\text{sim}(i,R)$, is defined as $\max\{\text{sim}(i,j) : j \in R\}$. The selection of the R-odd cut inequalities begins by first choosing the most violated inequality, which is added to a new set R^*. Consider now another violated inequality i. If $\text{sim}(i,R^*) \leq 0.9$, i is incorporated to R^* and the selection procedure continues. The inequalities in R^* are the ones that will be added to the current LP relaxation. From the computational experience, we have observed that $|R^*|$ is about 10%-20% of the total number of R-odd cut inequalities found.

4.3. Initial Upper Bound

To get good upper bounds, several heuristic algorithms have been tested. In particular, the constructive algorithms

H1 and H2 presented in [4], that are modifications of two algorithms described in [3], are used. These algorithms produce good feasible solutions for instances of moderate size. However, in large instances, the gap obtained at the root node is still big and a better upper bound would be helpful. For this purpose, the Scatter Search algorithm described in [4] has been tested, but it is a more time consuming procedure and the gap is not significantly reduced. Therefore, it has not been used here.

4.4. Artificial Upper Bound and Restarting

In some of the large instances, the gap at the root node may be around 3–4%. We have noticed that, in these cases, the lower bound is closer to the optimal value than the upper bound. So, in order to overcome this difficulty, if the gap at the root node is greater than 1.5%, we define an artificial upper bound with a value 1.005 times the lower bound. Obviously, this strategy can also be applied when no initial upper bound is available. For example, in some of the WGRP special cases we report later.

If the B&C ends without finding an optimal solution, the procedure is restarted using a greater artificial upper bound (1.010 times the lower bound). If, again, the B&C ends unsuccessfully, it is restarted for the last time using the true upper bound.

At each iteration, some of the cuts found are stored in order to be used at the next iteration if necessary. Specifically, if x^* denotes the optimal LP solution at the root node, all the cuts $ax \geq b$ such that $ax^* \leq b + 0.1$ are stored.

4.5. Branching Strategies

Several branching strategies have been tested. We first tried branching on constraints because some preliminary experiments showed that it performed better than branching on variables using the standard strategy of Cplex. The idea is to choose a valid inequality $ax \geq b$ such that $b^* = ax^*$ is a non-integer value, where x^* denotes the optimal LP solution at the current node. Then two subproblems are generated by adding the inequalities $ax \leq \lfloor b^* \rfloor$ and $ax \geq \lceil b^* \rceil$. The inequalities we have used for branching are the connectivity and R-odd cut inequalities present at the initial LP relaxation. When ax^* is integer for all these inequalities, we branch on variables. We have tested three different alternatives. The first one is branching on the variable whose fractional value is closer to 0.5 (strategy "Const + Var" in Table 1). The second one consists of allowing Cplex to choose "the best rule based on the problem and its progress" (row "Const + Cplex" in Table 1). Finally, we tested the strong branching strategy [1] implemented in Cplex ("Const + SB" in Table 1). Given that the best results were obtained with the "Const + SB" strategy, we also tried using only the Strong Branching strategy ("SB" in Table 1).

Table 1 shows the results obtained on a test set of 23 instances of large size with the 4 different branching strategies on a Pentium IV at 1.7 GHz machine with 512 MB RAM.

TABLE 1. Testing different branching strategies.

	Optima/total	%	Time (s)	Nodes	Gap(%)
Const + Var	7/23	30.4	30185.8	1025.9	1.544
Const + Cplex	7/23	30.4	31481.2	1041.7	1.718
Const + SB	7/23	30.4	28917.2	740.1	1.261
SB	11/23	47.8	22399.8	608.3	1.089

We set a time limit of 10 h for the B&C method. The first column gives the number of instances solved optimally and its percentage with respect to the total number. The second column shows the average CPU time and the last two columns present the average number of nodes of the B&C tree and the average gap, respectively. The best results, both in terms of time and optimal solutions found, were clearly obtained with the Strong Branching strategy. Therefore, we decided to finally use this strategy in our implementation.

5. COMPUTATIONAL EXPERIMENTS

In this section, we present the computational results obtained on different sets of instances. The B&C procedure has been coded in C/C++ using Cplex 9.0 MIP Solver with Concert Technology 2.0. All the tests were run on a Pentium IV at 2.8 GHz machine with 1GB RAM with a time limit of 10 h.

5.1. Data Instances

The performance of our B&C algorithm has been tested on several sets of instances of different sizes and characteristics. Some of them have already been used in the literature as test sets for other Arc Routing Problems. In what follows we describe the characteristics of these instances and how they were generated. All the test instances can be found in http://www.uv.es/~corberan/instancias.htm.

5.1.1. Instances on Windy Graphs. We first tested the B&C procedure on instances defined on windy graphs. In particular, we have used different sets of WRPP and WGRP instances that are described next. Table 2 shows the characteristics of the instances. It contains the problem type of each set of instances and the average, minimum, and maximum number of nodes, edges and R-sets, respectively.

Some of the WRPP sets were already used in [3] and [4] and were generated from undirected RPP instances taken from the literature. From a given RPP instance, 6 WRPP instances were generated with the same graph but different costs using the two strategies given by Win [34]:

- For each edge (i,j), two integer values $k_1, k_2 \in [-a, a]$ are randomly selected. New costs are defined as $c_{ij} = max\{1, c'_{ij} + k_1\}$ and $c_{ji} = max\{1, c'_{ij} + k_2\}$. Values 5, 8, and 10 for a were used.
- For each edge (i,j), two integer values $k_1, k_2 \in [a, b]$ are randomly selected. New costs are defined as $c_{ij} = k_1, c_{ji} = k_2$. Intervals $[1, 100]$, $[1, 200]$, and $[1, 500]$ were used.

TABLE 2. Characteristics of the windy instances.

		Nodes			Edges			R-sets		
Set	Problem	Aver.	Min	Max	Aver.	Min	Max	Aver.	Min	Max
CHR	WRPP	25.1	7	50	58.9	10	184	4.0	1	7
HG	WRPP	83.4	60	100	149.3	105	180	13.3	4	20
HD	WRPP	84.6	68	100	172.9	141	193	4.0	1	7
ALB	WRPP	116	116	116	174	174	174	14.1	9	22
MAD	WRPP	196	196	196	316	316	316	23	7	33
A500	WRPP	400.8	265	488	1268.2	842	1719	33.9	1	76
A1000	WRPP	848.3	599	988	2521.8	1656	3952	49.3	1	150
B500	WRPP	446.1	318	498	1131.5	630	1537	37.9	1	102
B50	WRPP	673.4	502	750	1706	1013	2288	55.8	1	152
B1000	WRPP	895.1	661	999	2286.7	1297	3073	76.1	1	202
GA500	WGRP	500	500	500	1135	855	1505	99.2	7	311
GB500	WGRP	500	500	500	1209.8	887	1549	93.3	1	281

Set CHR in Table 2 contains 144 WRPP instances generated from the 24 RPP instances proposed by Christofides et al. [7]. Sets HG and HD in Table 2 contain 54 WRPP instances each, that were generated from the 18 largest RPP instances presented in Hertz et al. [25]. Of these instances, 9 correspond to graphs with a grid structure (set HG) and the other 9 to graphs whose vertices have degree 4 (set HD). Finally, sets ALB and MAD were generated by Benavent et al. [3] from the street networks of two Spanish towns (Albaida and Madrigueras). In these instances, each edge is selected as required with probability $p \in \{0.3, 0.5, 0.7\}$. If there is a vertex not incident with any required edge, new required edges are selected until no such vertices remain. For each value of p, 2 instances are generated, thus obtaining 6 RPP instances from each graph. Applying Win's strategies twice for each RPP instance, a total of 72 WRPP instances were obtained for each set ALB and MAD.

In order to obtain larger instances, we have generated two types of random WRPP instances, called types A and B. As before, different undirected RPP instances are initially built, from which the final WRPP ones are generated. Type A instances correspond to pure random graphs, while type B instances are associated with graphs that try to imitate street networks.

We first select $|V|$ points in a square of size $1{,}000 \times 1{,}000$. For the instances of type A, $|E|$ edges are randomly generated as pairs of nodes (i, j) with costs defined by $c_{ij} = \lfloor b_{ij} + 0.5 \rfloor$, where b_{ij} are the Euclidean distances. If the resulting graph is not connected, edges in five different trees spanning the connected components of the graph are also added. At this point we have an undirected graph. Then, each edge is defined as required with probability p. An RPP instance has been generated for each possible combination of the parameters $|V| \in \{500, 1000\}$, $|E| \in \{1.5|V|, 2|V|, 3|V|\}$, and $p \in \{0.25, 0.50, 0.75\}$, producing a total of 18 graphs. Win's first strategy to assign costs has been used with parameter $a \in \{50, 80, 100\}$ to generate three instances from each graph. These 54 random instances are grouped into two sets called A500 and A1000. Numbers 500 and 1,000 refer to the number of nodes of the graphs.

Three sets of instances of type B have been generated in such a way that they are similar to real street networks. To do this, $|V|$ points in the $1{,}000 \times 1{,}000$ square are again randomly chosen. For each vertex v, d edges connecting v to its d closest neighbors are added. The idea is to avoid long edges crossing the graph from side to side that would not appear in real networks. As for the previous sets, if the resulting graph is not connected, edges in 5 different spanning trees are also added. Costs are also generated based on Euclidean distances. If, given an edge (i, j), there is a vertex k such that $c_{ij} \geq 0.98(c_{ik} + c_{kj})$, edge (i, j) is removed to forbid 'almost parallel' edges. To obtain asymmetric costs the first strategy by Win is applied with parameter a set to the 10 or 20% of the average edge cost (c_a). Then, two instances have been generated for each combination of the parameters $|V| \in \{500, 750, 1{,}000\}$, $p \in \{0.25, 0.50, 0.75\}$, $d \in \{3, 4, 5, 6\}$ and $a \in \{0.1c_a, 0.2c_a\}$, obtaining a total of 72 instances, grouped into 3 sets denoted by B500, B750 and B1,000, corresponding to the different values of $|V|$.

Since the graphs associated with the above-mentioned instances can contain vertices which are not incident with required edges, a simplification procedure similar to that presented in [8] or in [20] has been applied. Therefore, the number of vertices of the simplified instances can be less than the initial value fixed for $|V|$.

We have randomly generated WGRP instances by proceeding as above except that we do not simplify the graph, considering all the vertices not incident with required edges as (isolated) required vertices. We have obtained 27 and 24 WGRP instances with 500 vertices of type A and B, respectively, grouped in the sets GA500 and GB500.

5.1.2. Instances on Mixed Graphs. Since the WGRP contains the Mixed General Routing Problem as a special case, we have tested our algorithm in some sets of MGRP and MRPP instances taken from the literature. In [10], 81 MGRP instances were randomly generated from the street networks of Albaida and Madrigueras mentioned before, as well as from a third city called Aldaya, grouped on sets M-Alba, M-Madr, and M-Alda in Table 3. In [11], other sets of MGRP and MRPP instances were generated. The process used to generate these instances is similar to the one described for the windy instances of type B. From the undirected graphs, mixed instances are obtained by transforming edges into arcs using a given probability $q \in \{0.25, 0.5, 0.75\}$. If edge (i, j) is selected to be transformed into an arc, an orientation (i, j) or (j, i) is assigned with probability 0.5. Finally, if the number of strongly connected components is greater than one, some arcs are transformed back into edges to obtain a strongly connected graph. Two sets of 18 MRPP instances each, denoted by MR500 and MR1,000, and two of 18 MGRP instances, called MG500 and MG1,000, were generated.

5.1.3. Instances on Undirected Graphs. We have tested our algorithm on the 47 undirected instances described in [9]. The characteristics of these 40 GRP and 7 GTSP instances are presented in Table 4. The first two sets of instances consist

TABLE 3. Characteristics of the mixed instances.

		Nodes			R-sets		
Set	Problem	Aver.	Min	Max	Aver.	*Min*	Max
MR500	MRPP	449	357	498	35.8	1	102
MR1000	MRPP	899.7	708	999	68.8	1	188
M-Alba	MGRP	116	116	116	25.8	1	70
M-Madr	MGRP	196	196	196	65.0	2	168
M-Alda	MGRP	214	214	214	69.1	2	214
MG500	MGRP	500	500	500	86.8	3	245
MG1000	MGRP	1000	1000	1000	169.2	2	480

	Edges			Arcs		
Sets	Aver.	Min	Max	Aver.	Min	Max
MR500	583	261	987	550.8	210	976
MR1000	1178.1	521	1969	1137.3	470	1984
M-Alba	113.1	84	164	60.9	10	90
M-Madr	192.5	118	298	123.5	18	198
M-Alda	224	224	224	127	1	127
MG500	620.7	311	1021	597.3	277	991
MG1000	1239.9	611	2033	1209.9	532	2031

of 15 GRP instances each, randomly generated from the Albaida and Madrigueras graphs, denoted U-Alba and U-Madr respectively. The set denoted by U-GRP contains 10 GRP instances generated from the Albaida graph by visually selecting the required edges. The set denoted by GTSP contains 7 GTSP instances generated from instances from the TSPLIB.

To check the algorithm on larger instances, six sets with 12 instances each have been generated in a similar way to windy instances of type B. An RPP instance has been generated for each combination of the parameters $|V| \in \{500, 750, 1,000\}$, $p \in \{0.25, 0.50, 0.75\}$ and $d \in \{3, 4, 5, 6\}$. These instances are grouped in the sets denoted by UR500, UR750, and UR1,000 in Table 4. Sets UG500, UG750, and UG1,000 contain the analogous GRP instances.

5.1.4. Instances on Directed Graphs. Since the RPP and the GRP defined on directed graphs are special cases of the WGRP, in order to test our B&C method on these two problems, we have generated six sets with 12 directed instances each in a similar way to the undirected instances described above. First the undirected graphs are generated. Then, one of its two possible directions is assigned to each edge (required or not) with probability 0.5. If the resulting directed graph is not strongly connected, we add some nonrequired arcs joining its strongly connected components. The characteristics of the instances are shown in Table 5.

TABLE 4. Characteristics of the undirected instances.

		Nodes			Edges			R-sets		
Set	Problem	Aver.	Min	Max	Aver.	Min	Max	Aver.	Min	Max
UR500	RPP	446.0	298	499	1128.9	597	1526	35.3	1	99
UR750	RPP	665.7	452	749	1698.41	915	2314	55.7	1	140
UR1000	RPP	886.2	605	1000	2289.9	1122	3083	74.8	1	204
U-Alba	GRP	116	116	116	174	174	174	38.2	11	73
U-Madr	GRP	196	196	196	316	316	316	53.4	7	112
U-GRP	GRP	116	116	116	174	174	174	48.2	11	65
UG500	GRP	500	500	500	1213.1	858	1571	92.2	1	259
UG750	GRP	750	750	750	1826.6	1346	2288	135.0	1	412
UG1000	GRP	1000	1000	1000	2437.2	1765	3105	176.8	1	573
GTSP	GTSP	181.7	150	225	329.9	296	392	181.7	150	225

TABLE 5. Characteristics of the directed instances.

		Nodes			Arcs			R-sets		
Set	Problem	Aver.	Min	Max	Aver.	Min	Max	Aver.	Min	Max
DR500	DRPP	444.6	309	500	1245.1	747	1540	38.0	1	97
DR750	DRPP	667.6	435	750	1877.9	1105	2356	50.2	1	140
DR1000	DRPP	888.3	617	999	2517.7	1563	3139	73.7	1	213
DG500	DGRP	500	500	500	1347.9	1166	1571	92.5	1	286
DG750	DGRP	750	750	750	2033.4	1788	2276	138.9	2	422
DG1000	DGRP	1000	1000	1000	2731.2	2342	3177	184.2	4	553

5.2. Computational Results

The computational results obtained on the above sets of instances are reported in Tables 6–9. In all the tables, the first column shows the name of the set of instances tested. The number of optimal solutions obtained for each set and the total number of instances is given in column 2. For the solved instances (within the time limit of 10 h), columns 3 and 4 present the average computing time (in seconds) and the number of B&C nodes explored. Last column shows the average gap for the unsolved instances.

As can be seen in Table 6, our algorithm solved all the WRPP instances of small and medium size. It was also capable of solving to optimality all the large WRPP and WGRP instances of type A (generated completely at random, A500, A1,000, and GA500). On the other hand, instances of type B (generated trying to imitate real networks) proved to be more difficult and, in some cases, the algorithm could not find the optimal solution within the time limit of 10 h, although the

TABLE 6. Results on windy instances.

	Optima/total	Time (s)	B&C Nodes	Gap (%)
CHR	144/144	0.2	0.2	–
HG	54/54	1.8	1.1	–
HD	54/54	0.6	1.8	–
ALB	72/72	0.8	1.5	–
MAD	72/72	3.0	3.4	–
A500	27/27	12.2	5.6	–
A1000	27/27	657.2	82.4	–
B500	24/24	75.5	24.8	–
B750	23/24	1451.6	1111.2	0.402
B1000	18/24	1241.4	82.8	1.951
GA500	27/27	2377.9	105.2	–
GB500	19/24	1428.0	135.9	2.960

TABLE 7. Results on mixed instances.

	Optima/total	Time (s)	B&C Nodes	Gap (%)
MR500	18/18	13.2	6.2	–
MR1000	18/18	1187.7	79.0	–
M-Alba	25/25	0.9	6.1	–
M-Madr	25/25	1412.4	718.3	–
M-Alda	31/31	30.6	28.6	–
MG500	17/18	2243.6	84.3	0.390
MG1000	14/18	2722.8	30.2	0.376 (*)

final gap is small. The results shown in the Table also confirm that, as expected, WGRP instances are harder than WRPP ones, since they have a greater number of R-sets.

Table 7 reports the computational results obtained on the MRPP and MGRP sets of instances. Again, it can be noticed that MRPP instances are easier than MGRP ones. The performance of the B&C procedure is very good on these types of instances, especially considering that the heuristics for the WGRP can not be applied on mixed graphs and therefore an initial upper bound was not available. All the instances were solved to optimality except for five very large MGRP instances. The algorithm was able to find a feasible solution for two of these instances, one in set MG500 and another one in set MG1,000. The gap marked with a '$*$' in the Table refers to the only instance in set MG1,000 for which a feasible solution was found. Note that the instances in set MG1,000 have 1,000 vertices and, on average, 169 R-sets and 2,400 links, 1,200 of which are required.

In [10], computational experiments with a cutting-plane algorithm for the MGRP are presented. This procedure, which invokes the Branch & Bound option of Cplex when the final solution of the cutting-plane algorithm is fractional, gives good computational results on the Albaida, Madrigueras and Aldaya instances. However, three instances from the Albaida test set, 5 from the Madrigueras one and 8 from the Aldaya one could not be solved. It can be observed in Table 7 that all these instances have been solved by our B&C algorithm, as well as other MRPP and MGRP instances of larger size and greater difficulty.

On the other hand, comparing the results given in Tables 6 and 7 for pairs of sets with similar characteristics, such as B500 and MR500, B1,000 and MR1,000, and GB500 and MG500, it can be concluded that ARP instances defined on windy graphs are harder than those defined on mixed graphs. For example, we can see that all the instances in the set MR1,000 were solved to optimality, while this was not possible for 6 instances in the B1,000 set.

TABLE 8. Results on undirected instances.

	Optima/total	Time (s)	B&C Nodes	Gap (%)
UR500	12/12	26.6	8.8	–
UR750	12/12	2873.3	183.3	–
UR1000	12/12	3033.7	185.8	–
U-Alba	15/15	2.3	11.0	–
U-Madr	15/15	27.3	35.1	–
U-GRP	10/10	42.2	59.6	–
UG500	11/12	1028.8	74.4	0.402
UG750	10/12	2386.7	29.6	2.436
UG1000	9/12	4295.9	47.0	3.733
GTSP	7/7	3103.1	2135.6	–

TABLE 9. Results on directed instances.

	Optima/total	Time (s)	B&C Nodes	Gap (%)
DR500	12/12	1.5	0.1	–
DR750	12/12	3.5	0.0	–
DR1000	12/12	30.8	1.3	–
DG500	12/12	712.3	82.0	–
DG750	11/12	252.3	1.8	0.374
DG1000	11/12	1010.4	2.4	Unknown

Table 8 shows the results obtained with our B&C algorithm on the instances defined on undirected graphs. In spite of not being a specific algorithm for solving instances with symmetric costs and considering that the upper bound obtained with the WGRP heuristics are not good, the results obtained are quite satisfactory. To our knowledge, these are

TABLE 10. Number of violated inequalities.

	Conn	R-odd	K-C	K-C_{02}	HC	HC_{02}	PB	ZZ	Total
CHR	0.9	5.9	2.6	2.0	0.2	0.2	0.0	0.1	12.0
HG	2.2	32.7	8.5	5.5	1.6	0.6	0.3	0.6	51.9
HD	5.7	48.6	21.9	9.2	3.7	1.7	0.8	1.2	92.7
ALB	6.4	60.8	17.2	10.1	2.8	1.9	0.6	1.1	100.9
MAD	5.7	150.5	33.3	13.6	4.6	1.4	0.6	1.3	211.1
A500	2.7	146.2	7.4	5.1	0.3	0.1	0.9	0.7	163.6
A1000	1.4	582.7	8.9	12.6	0.3	0.0	2.5	0.4	608.7
B500	20.5	867.0	86.2	37.9	29.4	7.9	1.8	1.1	1051.8
B750	32.1	2082.3	152.1	65.3	34.3	2.5	7.8	8.2	2384.6
B1000	49.1	3350.7	234.6	76.6	27.6	3.6	42.9	6.3	3791.5
GA500	25.7	326.8	26.9	7.4	0.6	0.0	6.6	0.4	394.4
GB500	75.0	1428.0	188.8	65.3	48.4	3.5	46.1	4.0	1859.1
MR500	7.3	284.1	11.4	9.1	1.3	0.2	0.9	0.2	314.6
MR1000	15.6	1489.3	84.6	47.7	12.6	0.6	2.3	1.1	1653.7
M-Alba	19.6	29.1	8.1	2.7	2.0	1.1	0.8	0.0	63.4
M-Madr	69.6	131.5	145.2	7.8	4.5	1.3	37.3	1.0	398.2
M-Alda	33.1	116.1	27.8	11.2	4.7	0.7	5.5	1.0	200.0
MG500	36.0	898.1	109.8	37.1	21.6	1.6	8.9	0.9	1114.1
MG1000	48.1	866.3	118.7	54.4	13.7	0.8	9.3	0.4	1111.7
UR500	20.2	580.7	65.1	40.8	15.3	6.7	1.5	3.0	733.1
UR750	34.0	1713.6	143.8	66.1	23.4	4.6	8.3	5.2	1999.0
UR1000	44.8	1964.8	173.5	61.8	19.0	1.0	20.8	14.7	2300.3
U-Alba	21.9	83.3	18.8	11.5	3.8	3.6	3.1	0.1	146.1
U-Madr	25.4	208.2	55.7	18.3	11.3	3.5	4.5	3.8	330.6
U-GRP	35.3	55.3	181.3	59.0	96.5	50.1	1.7	1.0	480.2
UG500	63.5	1485.0	163.3	66.5	32.1	6.7	31.7	7.5	1856.2
UG750	90.9	1511.4	172.7	84.2	43.6	3.7	16.7	4.6	1927.7
UG1000	108.1	2022.0	232.0	102.6	50.8	6.3	12.8	3.6	2538.1
GTSP	372.7	0	0	0	0	0	291.6	0	664.3
DR500	10.5	0	3.6	0.1	1.2	0.3	0.2	0.0	15.8
DR750	8.9	0	1.8	0.0	0.6	0.0	0.0	0.0	11.3
DR1000	19.7	0	6.3	0.0	0.3	0.1	0.6	0.1	27.0
DG500	47.1	0	36.3	0.2	25.0	0.3	2.0	0.3	111.2
DG750	57.5	0	10.3	0.2	8.7	1.5	3.1	0.3	81.5
DG1000	64.7	0	24.1	0.1	14.7	1.1	1.3	0.0	105.9

the biggest GRP and RPP instances reported in the literature. Even the GTSP instances, which are pure node routing problems, have been solved with our algorithm, including the TSPLIB instance ts225, which was deliberately constructed to be difficult for cutting-plane algorithms.

The results obtained on directed instances, shown in Table 9, are very good, specially considering that an initial upper bound was not available. Note that DRPP instances with up to 1,000 vertices, 3,100 arcs and 213 R-sets are solved in less than 1 min on average, most of them at the root node. Also, most of the DGRP instances are solved to optimality in moderate computing times, some of them having up to 550 R-sets. We do not know of any other exact method published in the literature capable of solving such large instances.

Table 10 shows the average number of violated inequalities of each type found by the separation algorithms for each set of instances. Note that some families of inequalities are not facet-inducing for some types of problems, e.g. the R-odd and Odd Zigzag inequalities on directed graphs (that are dominated by the symmetry equations). In these cases, the corresponding separation algorithm has not been used. Looking at this table, it can be seen that violated R-odd cut and K-C inequalities are more frequent than the other classes of inequalities. The violated inequalities of each class found depends heavily on the characteristics of the instances, such as whether there are isolated required vertices or not, the number of R-sets or the asymmetry of the costs. Note also that the number of violated inequalities found is much bigger for the instances generated imitating real networks than for those generated completely at random.

6. CONCLUSIONS

In this study we have presented a B&C algorithm for the WGRP which is also capable of efficiently solving other well-known routing problems such as the RPP and the GRP defined on undirected, directed and mixed graphs. All of them are important Arc Routing Problems with many real-life applications.

We have implemented separation algorithms for several families of facet-defining inequalities, including a new separation algorithm for the Zigzag inequalities. We have tried different techniques to improve the performance of the B&C, such as the use of less dense versions of facet-inducing inequalities, heuristic algorithms to get upper bounds, artificial upper bounds, and selection of violated inequalities.

This exact algorithm has been tested on a wide set of instances of different Arc Routing Problems Routing Problems and has been able to solve instances up to 1,000 nodes, 4,000 links, and 500 R-sets. In spite of not being a specific algorithm for solving instances with symmetric costs, our algorithm has performed well on instances defined on undirected graphs. Also its behavior has been very good on directed and mixed instances. We think that these results confirm the usefulness of the polyhedral approach to solving difficult combinatorial optimization problems.

REFERENCES

[1] D. Applegate, R.E. Bixby, V. Chvátal, and W. Cook, Finding cuts in the TSP, Technical Report, DIMACS 95–05, 1995.

[2] C. Balaguer, A. Giménez, J.M. Pastor, V.M. Padrón, and M. Abderrahim, A climbing autonomous robot for inspection applications in 3D complex environments, Robotica 18 (2000), 287–297.

[3] E. Benavent, A. Carrotta, A. Corberán, J.M. Sanchis, and D. Vigo, Lower bounds and heuristics for the windy rural postman problem, Eur J Operational Res 176 (2007), 855–869.

[4] E. Benavent, A. Corberán, E. Piñana, I. Plana, and J.M. Sanchis, New heuristics for the windy rural postman problem, Comput Operations Res 32 (2005), 3111–3128.

[5] P. Brucker, The Chinese postman problem for mixed graphs, Proceedings of this International Workshop. Lecture Notes in Computer Science 100 (1981), 354–366.

[6] S. Chopra and G. Rinaldi, The graphical asymmetric traveling salesman polyhedron: Symmetric inequalities, SIAM J Discr Math 9 (1996), 602–624.

[7] N. Christofides, V. Campos, A. Corberán, and E. Mota, An algorithm for the rural postman problem, Imperial College, London, 1981.

[8] N. Christofides, V. Campos, A. Corberán, and E. Mota, An algorithm for the rural postman problem on a directed graph, Math Program Study 26 (1986), 155–166.

[9] A. Corberán, A. Letchford, and J.M. Sanchis, A cutting plane algorithm for the general routing problem, Math Program 90 (2001), 291–316.

[10] A. Corberán, G. Mejía, and J.M. Sanchis, New results on the mixed general routing problem, Operations Res 53 (2005), 363–376.

[11] A. Corberán, E. Mota, and J.M. Sanchis, A comparison of two different formulations for arc routing problems on mixed graphs, Comput & Operations Res 33 (2006), 3384–3402.

[12] A. Corberán, I. Plana, and J.M. Sanchis, The windy general routing polyhedron: A global view of many known arc routing polyhedra, Technical Report TR06-2005 (http://www.uv.es/sestio/TechRep/tr06-05.pdf), Department of Statistics and O.R., University of Valencia, Spain.

[13] A. Corberán, I. Plana, and J.M. Sanchis, Zigzag inequalities: A new class of facet-inducing inequalities for arc routing problems, Math Program 108 (2006), 79–96.

[14] A. Corberán, A. Romero, and J.M. Sanchis, The mixed general routing problem polyhedron, Math Program 96 (2003), 103–137.

[15] A. Corberán and J.M. Sanchis, A polyhedral approach to the rural postman problem, Eur J Operational Res 79 (1994), 95–114.

[16] A. Corberán and J.M. Sanchis, The general routing problem: Facets from the RPP and GTSP polyhedra, Eur J Operational Res 108 (1998), 538–550.

[17] G. Cornuèjols, J. Fonlupt, and D. Naddef, The traveling salesman problem on a graph and some related integer polyhedra, Math Program 33 (1985), 1–27.

[18] M. Dror (Editor), Arc routing: Theory, solutions and applications, Kluwer, New York, 2000.

[19] J. Edmonds and E. Johnson, Matching, euler tours and the chinese postman problem, Math Program 5 (1973), 88–124.

[20] H.A. Eiselt, M. Gendreau, and G. Laporte, Arc-routing problems, Part 2: The rural postman problem, Operations Res 43 (1995), 399–414.

[21] B. Fleischmann, A cutting-plane procedure for the traveling salesman problem on a road network, Eur J Operational Res 21 (1985), 307–317.

[22] M. Grötschel and Z. Win, A cutting plane algorithm for the windy postman problem, Math Program 55 (1992), 339–358.

[23] M. Guan, Graphic Programming using odd and even points, Chinese Math 1 (1962), 273–277.

[24] M. Guan, On the windy postman problem, Discr Appl Math 9 (1984), 41–46.

[25] A. Hertz, G. Laporte, and P. Nanchen, Improvement procedures for the undirected rural postman problem, INFORMS J Computing 11 (1999), 53–62.

[26] A. Letchford, New inequalities for the general routing problem, Eur J Operational Res 96 (1997), 317–322.

[27] E. Minieka, The Chinese postman problem for mixed networks, Manage Sci 25 (1979), 643–648.

[28] D. Naddef and G. Rinaldi, The symmetric traveling salesman polytope and its graphical relaxation: Composition of valid inequalities, Math Program 51 (1991), 359–400.

[29] C. Orloff, A fundamental problem in vehicle routing, Networks 27 (1974), 95–108.

[30] M. Padberg and M. Rao, Odd minimum cut-sets and b-matchings, Math Operations Res 7 (1982), 67–80.

[31] M. Padberg and G. Rinaldi, Optimization of a 532 city symmetric traveling salesman problem by branch and cut, Operations Res Lett 6 (1987), 1–7.

[32] C. Papadimitriou, On the complexity of edge traversing, J ACM 23 (1976), 544–554.

[33] I. Plana, The windy general routing problem, Ph.D. Thesis, University of Valencia, Spain, 2005.

[34] Z. Win, Contributions to routing problems, Ph.D. Thesis, University of Augsburg, Germany, 1987.