A Branch & Cut Algorithm for the Windy General Routing Problem

Angel Corberán[1]*, **Isaac Plana**[1] **and José M. Sanchis**[2]

[1] Dept. d'Estadística i Investigació Operativa, Universitat de València, Spain
[2] Dept. de Matemática Aplicada, Universidad Politécnica de Valencia, Spain

July 26, 2005

Abstract

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.

Key Words: 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 ([22]) 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. 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 & Johnson ([18]). 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 ([31]) and it generalizes both the CPP and the DCPP. Finally, Minieka ([26]) 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 [23] showed that the WPP is NP-hard, although it can be solvable in polynomial time if the graph is even ([33]), or if the two orientations of every cycle have the same cost ([23]). Moreover, in [33] and [21] an ILP 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

*corresponding author: angel.corberan@uv.es

$c_{ij} = c_{ji}$, 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 ([28]) 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 ([20]) and also studied in [27]. The RPP, GRP and GTSP were initially defined for undirected graphs, but they can also be formulated on directed (DRPP, DGRP, GATSP), mixed (MRPP, MGRP, GATSP) and 'windy' graphs (WRPP, WGRP, GATSP). Note that the GTSP defined either 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 above, 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 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 above 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 3-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 non-required 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 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 the conclusions.

2 Problem formulation and polyhedral results

The Windy General Routing Problem 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 [19]). The subgraph associated with the required vertices and edges, $G_R = (V, E_R)$, is in general non-connected. We denote by p the number of its connected components (R-components) and by $V_1, V_2, \ldots, V_p$, with $V_1 \cup \ldots \cup V_p = V$, their corresponding vertex sets (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 edge set with an end-point in S and the other in $V \setminus S$ 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) will represent the set of edges with one end-point in S_1 and the other in S_2, while $\delta_R(S)$, $E_R(S)$, $(S_1, S_2)_R$ will 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),

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

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

The WGRP can be formulated as follows:

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

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

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

$$\sum_{i\in S,\ j\notin S} x_{ij} \geq 1, \quad \forall S = \bigcup_{k\in Q} V_k, \quad Q \subset \{1, \ldots, p\} \tag{3}$$

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

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

Conditions (1) guarantee that each required edge will be 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 (4), WGRP(G), has been studied in [12] and [32]. It is an unbounded polyhedron of dimension $2|E| - |V| + 1$, 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) will be 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.

Connectivity and R-odd cut inequalities

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

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

are also facet-inducing of WGRP(G) 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:

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

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 $\{M_0, M_1, \dots, M_{K-1}, M_K\}$, 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, $j = 1, \dots, K-1$, 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 non-empty, for $i = 0, \dots, K-1$. The inequality, which can be represented by a graph like the one shown in figure 1, takes the form:

$$\sum_{(i,j)\in(M_0,M_K)} (K-2)\ (x_{ij} + x_{ji}) + \sum_{(i,j)\in(M_i,M_j)\neq(M_0,M_K)} |i-j|\ (x_{ij} + x_{ji})\ \geq$$
$$\geq\ 2(K-1)\ +\ (K-2)(|(M_0, M_K)_R|) \tag{8}$$

K-C inequalities can be written in a simpler way 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 $M_1, M_3, \dots M_{K-1}$, then an equivalent inequality is obtained which, divided by 2, takes the form:

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

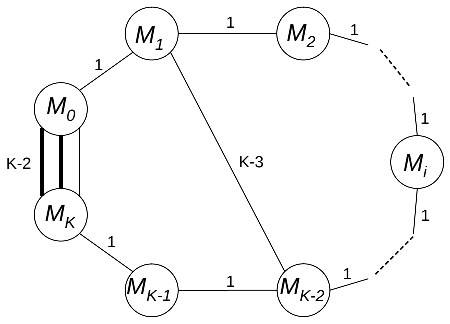


Figure 1: Graph representing a K-C inequality

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

Note that, 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:

$$\sum_{(i,j)\in(M_0,M_K)} (K-1)\ (x_{ij} + x_{ji}) \ + \sum_{(i,j)\in(M_0,M_j),j\neq K} ((j-1)\ x_{ij} + (j+1)\ x_{ji}) \ +$$
$$+ \sum_{(i,j)\in(M_i,M_j),i,j\neq 0} |i-j|\ (x_{ij} + x_{ji}) \ \geq \ 2(K-1) + (K-1)(|(M_0, M_K)_R|) \qquad (10)$$

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

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 ([25]). Figure 2 shows a graph representing a Path-Bridge inequality. A PB inequality is called *regular* when all its paths have the same number of nodes.

As with the K-C inequalities, equivalent Path-Bridge inequalities with fewer non-zero elements can be obtained. Details on these inequalities and the conditions under which they are facet-inducing for WGRP(G) are given in [12]. In that paper 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.

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

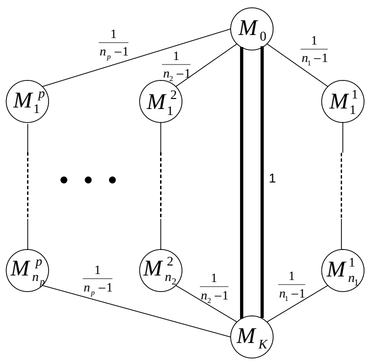


Figure 2: Graph representing a Path-Bridge inequality

into $L \geq 2$ subsets. Honeycomb 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 Honeycomb inequality is shown in figure 3. With a similar structure, but with different coefficients from the standard Honeycomb inequalities, there is another family of facet-inducing inequalities, the Honeycomb$_{02}$ inequalities ([10], [12]). Versions of the Honeycomb and Honeycomb$_{02}$ inequalities with fewer non-zero coefficients can also be obtained.

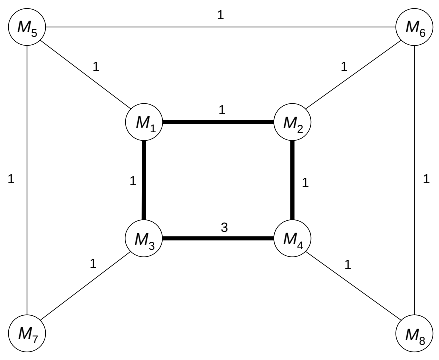


Figure 3: Graph representing a Honeycomb inequality

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 4) is:

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

Basically, when $|\delta_R(M_i)|$ is odd, $i = 1, \ldots, 4$, we have the Odd Zigzag inequalities, which can look similar to the Even Zigzag inequalities (see figure 5a), but are in general rather more complicated (see figure 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 Branch & Cut algorithm, they are not presented here.

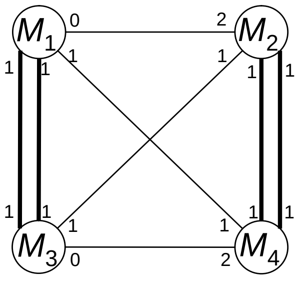
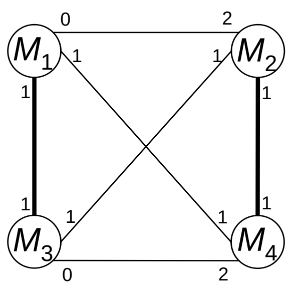
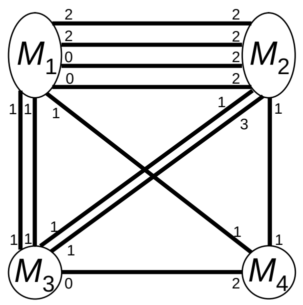


Figure 4: Graph representing an Even Zigzag inequality



(a)



(b)

Figure 5: Graphs representing simple and general Odd Zigzag inequalities

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, all the parallel resulting edges 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 ([29]) 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 wether 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 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 non-required 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^*.

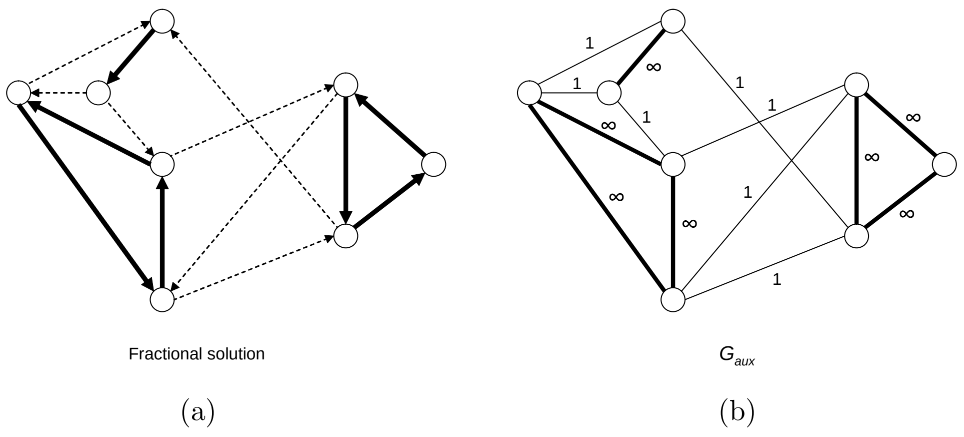


Figure 6: Fractional solution x^* and capacitated graph G_{aux}

Given x^*, we assign the following capacities to the edges in graph G^*: non-required edges with $x^*_{ij} = 0.5$ and $x^*_{ji} = 0$ (or viceversa) will be given capacity 1, while all the other edges get infinite capacity. This capacitated graph will be called G_{aux} (see figure 6b).

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 will be at least four, otherwise there would be a violated connectivity constraint. If the flow value is exactly four, an edge cutset (V_1, V_2) of capacity four is identified (see figure 7). By construction, this edge cutset consists of four non-required edges satisfying $x^*_{ij} = 0.5$ and $x^*_{ji} = 0$ with which we will 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

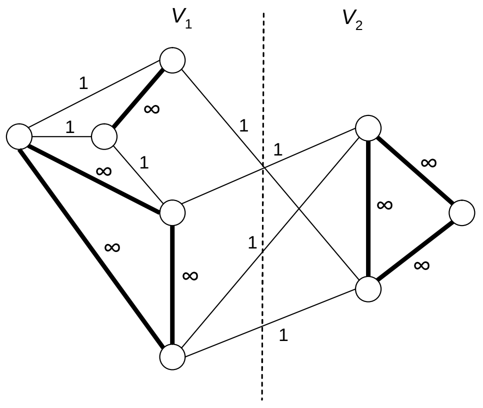


Figure 7: Minimum capacity cutset

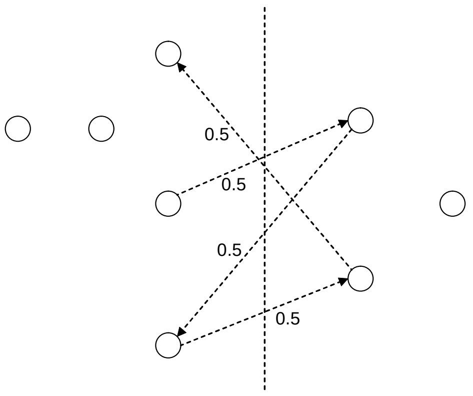


Figure 8: Zigzag found

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 non-required 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 (see figure 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.

Note that in the construction of graph G_{aux} it is possible to assign capacity 1 not only to the non-required 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.

4 The Branch & Cut method

In this section we describe the details on our implementation of the Branch & Cut method. Branch & Cut (B&C) algorithms were first introduced by Padberg and Rinaldi ([30]), and have

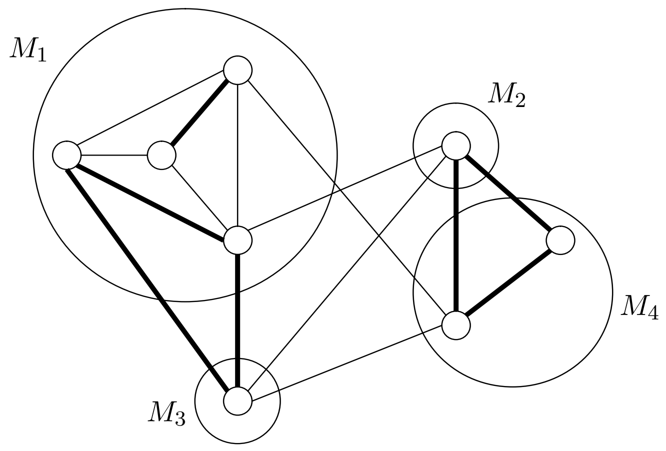


Figure 9: Partition of V into M_1, M_2, M_3 and M_4

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 will be 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 to the connected components S of the graph defined 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. The ordering applied is the following:

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 8.
8. If no violated inequality of any class has been found, and only at the root node of the B&C tree, run heuristics for K-C and K-C_{02} with iterative merging of adjacent R-components.

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 5 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 will be 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, $sim(i,j)$, 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, $sim(i,R)$, is defined as $\max\{sim(i,j) : j \in R\}$. 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 $sim(i,R^*) \leq 0.9$, 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

In order to get good upper bounds, several heuristic algorithms have been used. First, the constructive algorithms presented in [3] and [4] are invoked. Usually these algorithms produce quite good feasible solutions for instances of moderate size. However, in larger instances, the gap obtained at the root node is too big and a better upper bound is needed. For this purpose, the Scatter Search algorithm described in [4] is applied. This is a more time consuming procedure that will only be used when the gap at the root node is larger than 1%.

4.4 Artificial upper bound and restarting

Even with the upper bound provided by the Scatter Search algorithm, the gap is still big in some of the larger instances. 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.

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. We set a time limit of 10 hours 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.

	Optima/total	%	Time (sec)	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%

Table 1: Testing different branching strategies

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 1.7GHz machine with 512MB RAM with a time limit of 10 hours.

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.

		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
C750	WRPP	673.4	502	750	1706	1013	2288	55.8	1	152
D1000	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

Table 2: Characteristics of the 'windy' instances

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 ([33]):

- 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 $[1, 100]$, $[1, 200]$ y $[1, 500]$ were used.

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. ([24]). 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, 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. As before, different undirected RPP instances are initially built, from which the final WRPP ones are generated. The first set corresponds to 'pure' random graphs, while the second one will be associated with graphs that try to imitate street networks.

We first select $|V|$ points in a square of size 1000×1000. For the first type of instances, $|E|$ 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 5 different trees spanning the connected components of the graph are also added. 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 3 instances from each graph. These 54 random instances are grouped into 2 sets called A500 and A1000. Numbers 500 and 1000 refer to the number of nodes of the graphs.

Three subsets of new instances have been generated in such a way that they are similar to real street networks. To do this, $|V|$ points in the 1000×1000 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. The first strategy by Win is applied to obtain asymmetric costs. Two instances have been generated for each combination of the parameters $|V| \in \{500, 750, 1000\}$, $p \in \{0.25, 0.50, 0.75\}$, $d \in [3, 6]$ and $a \in \{10, 20\}$, obtaining a total of 72 instances, grouped into 3 sets denoted by B500, C750 and D1000, corresponding to the different values of $|V|$.

Since the graphs associated with the above instances can contain vertices which are not incident with required edges, a simplification procedure similar to that presented in [8] or in [19] 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 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 sets B500, C750 and D1000 of WRPP instances. If all the vertices are considered as required vertices, we obtain GRP instances. Otherwise, to obtain RPP instances, the simplification procedure mentioned above is applied. In both cases, from the undirected graphs, mixed instances are obtained by transforming edges into arcs using a given probability $p \in \{0.25, 0.5, 0.75\}$. Two sets of 18 MRPP instances, denoted by MRB500 and MRD1000, and 2 of 18 MGRP instances, called MGB500 and MGD1000, were generated.

		Nodes			R-sets		
Set	Problem	Aver.	Min	Max	Aver.	Min	Max
MRB500	MRPP	449	357	498	35.8	1	102
MRD1000	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
MGB500	MGRP	500	500	500	86.8	3	245
MGD1000	MGRP	1000	1000	1000	169.2	2	480
		Edges			Arcs		
Sets		Aver.	Min	Max	Aver.	Min	Max
MRB500		583	261	987	550.8	210	976
MRD1000		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
MGB500		620.7	311	1021	597.3	277	991
MGD1000		1239.9	611	2033	1209.9	532	2031

Table 3: Characteristics of the ‘mixed’ instances

5.1.3 Instances on undirected graphs

We have also tested our algorithm on 47 instances defined on undirected graphs. These 40 GRP and 7 GTSP instances are described in [9] and their characteristics are presented in table 4. The first 2 sets of instances consist 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.

		Nodes			Edges			R-sets		
Set	Problem	Aver.	Min	Max	Aver.	Min	Max	Aver.	Min	Max
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
GTSP	GTSP	181.7	150	225	329.9	296	392	181.7	150	225

Table 4: Characteristics of the ‘undirected’ instances

5.2 Computational results

The computational results obtained on the above sets of instances are reported in tables 5, 6 and 7. 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. Column 3 gives the percentage of optima with respect to the total number of instances. The average computing time (in seconds), number of B&C nodes explored and the gap for each of the sets are presented in the last three columns.

	Optima/total	%	Time (sec)	B&C Nodes	Gap
CHR	144 / 144	100%	0.2	0.2	0%
HG	54 / 54	100%	1.5	1.3	0%
HD	54 / 54	100%	1.4	1.3	0%
ALB	72 / 72	100%	2.3	1.5	0%
MAD	72 / 72	100%	9.8	2.7	0%
A500	27 / 27	100%	39.5	6.5	0%
A1000	27 / 27	100%	607.3	38.2	0%
B500	24 / 24	100%	162.9	23.7	0%
C750	21 / 24	87.5%	5183.9	143.9	0.027%
D1000	20 / 24	83.3%	8972.9	282.0	0.410%
GA500	27 / 27	100%	3279.9	57.1	0%
GB500	19 / 24	79.2%	8463.0	193.4	0.607%

Table 5: Results on ‘windy’ instances

As can be seen in table 5, 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 generated completely at random (A500, A1000 and GA500). On the other hand, instances generated trying to imitate real networks proved to be more difficult and, in some cases, the algorithm could not find the optimal solution, although the final gap is very small. These results also confirm that, as expected, WGRP instances are harder than WRPP ones, since they have a greater number of R-sets.

	Optima/total	%	Time (sec)	B&C Nodes	Gap
MRB500	18 / 18	100%	18.7	4.8	0%
MRD1000	18 / 18	100%	1106.8	40.7	0%
M-Alba	25 / 25	100%	1.7	4.6	0%
M-Madr	25 / 25	100%	508.5	101.1	0%
M-Alda	31 / 31	100%	58.8	25.2	0%
MGB500	18 / 18	100%	2721.4	59.7	0%
MGD1000	13 / 18	72.2%	10839.3	21.8	0.057%

Table 6: Results on ‘mixed’ instances

Table 6 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 could not be applied in this case and therefore an initial upper bound was not available. All the instances were solved to optimality except for 5 very large MGRP instances. Note that the instances in set MGD1000 have 1000 vertices and, on average, 169 R-sets and 2400 links, 1200 of which are required. For 3 of the unsolved instances, the algorithm was not able to find a feasible solution, therefore these instances were not considered at the calculation of the average gap shown in the table. However, their computation time (10 hours) and the number of nodes of their B&C trees have indeed been included in the corresponding averages.

In [10], computational experiments with a cutting-plane algorithm for the MGRP are presented. This procedure, which invokes the Branch and 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, 3 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 6 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, from the results given in tables 5 and 6, it can be concluded that ARP instances defined on windy graphs are harder than those defined on mixed graphs. For example, if we compare the results obtained for the sets GB500 and MGB500, which have similar characteristics, we can see that all the instances in the set MGB500 were solved to optimality, while this was not possible for 5 instances in the GB500 set.

	Optima/total	%	Time (sec)	B&C Nodes	Gap
U-Alba	15 / 15	100%	4.7	3.5	0%
U-Madr	15 / 15	100%	60.8	26.7	0%
U-GRP	10 / 10	100%	53.7	38.9	0%
GTSP	6 / 7	85.7%	5886.0	359.0	0.744%

Table 7: Results on ‘undirected’ instances

Finally, table 7 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, the results obtained with the B&C procedure on the GRP instances are quite good, although, as expected, its performance is not so good on the GTSP instances. Note that, since they are pure Node Routing Problems, they need more specific techniques. Moreover, as noted in [9], the only unsolved GTSP instance is based on the TSPLIB instance ts225, which was deliberately constructed to be difficult for cutting-plane algorithms.

Table 8 shows the average number of violated inequalities of each type found by the separation algorithms for each set of instances. Violated R-odd cut and K-C inequalities are more frequent than the other classes of inequalities. In some sense, it seems that K-C inequalities play the role of the parity constraints, which do not exist for general (non-binary) variables. The violated inequalities of each class found depends heavily on the characteristics of the instances, such as wether 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 paper we have presented a B&C algorithm to solve the Windy General Routing Problem, which is an important Arc Routing Problem that has many applications and contains many other well-known routing problems as special cases.

We have implemented separation algorithms for several families of facet-defining inequalities. Most of these algorithms are based on the corresponding algorithms previously designed in the context of other routing problems. A new separation algorithm for the Zigzag inequalities has been presented. 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.

	Conn	R-odd	K-C	K-C_{02}	HC	HC_{02}	PB	ZZ	Total
CHR	0.9	5.8	2.7	2.0	0.3	0.3	0.0	0.1	12.0
HG	2.2	30.6	8.2	5.3	1.6	0.6	0.3	0.6	49.3
HD	5.7	48.0	20.9	9.1	3.8	1.6	0.7	1.3	91.1
ALB	6.6	65.3	18.8	10.8	3.4	2.2	0.7	1.3	109.2
MAD	5.8	150.7	33.8	13.8	4.7	1.4	0.7	1.1	211.9
A500	2.9	211.9	9.7	5.6	0.3	0.1	1.3	0.8	232.6
A1000	1.5	600.9	10.6	14.0	0.3	0	2.0	0.4	629.8
B500	22.3	1085.5	93.3	42.8	31.8	8.3	2.9	2.2	1289.0
C750	33.6	2222.7	162.5	65.2	42.1	2.6	6.3	5.6	2540.7
D1000	47.3	2935.6	226.5	79.9	27.1	3.3	23.6	5.0	3348.3
GA500	25.1	294.3	27.5	7.8	0.7	0.0	5.6	0.1	361.3
GB500	71.0	1479.8	187.2	66.2	41.0	3.3	23.4	2.6	1874.5
MRB500	7.0	20.7	12.3	8.2	0.9	0.1	0.9	0.3	280.4
MRD1000	14.4	1110.4	75.5	35.5	10.4	0.3	3.6	1.4	1251.5
M-Alba	19.8	34.5	8.4	3.5	2.4	1.4	1.1	0	71.1
M-Madr	42.3	83.6	61.2	5.0	4.4	1.0	8.7	0.3	206.4
M-Alda	31.6	125.3	29.9	11.2	4.8	0.9	5.4	0.3	209.4
MGB500	33.2	654.2	97.0	34.8	16.5	1.3	5.8	0.7	843.6
MGD1000	46.1	875.1	101.1	48.9	12.2	1.2	9.4	1.8	1095.7
U-Alba	21.2	75.9	17.3	11.2	3.8	3.6	2.1	0.1	135.1
U-Madr	27.2	222.8	68.0	16.5	12.9	4.4	3.7	2.3	357.7
U-GRP	35.6	55.6	172.0	56.0	85.8	41.7	1.9	1.0	449.6
GTSP	305.1	0	0	0	0	0	115.7	0	420.8

Table 8: Number of violated inequalities

This exact algorithm has been tested on a wide set of WRPP and WGRP instances and has been able to solve instances up to 1000 nodes, 4000 edges and 300 R-sets. Since the WGRP generalizes many other problems, our algorithm can also be used to solve instances of such problems. Particularly we have extended the tests to instances of other routing problems defined on mixed and undirected graphs that have already been used in the literature. In spite of not being specifically designed for this kind of problems, our algorithm has performed quite well on instances of the MRPP, MGRP, GRP and GTSP. We think that these results are very good and confirm the usefulness of the polyhedral approach to solving difficult combinatorial optimization problems.

Acknowledgments: The authors wish to thank the Ministerio de Ciencia y Tecnología of Spain (project TIC2003-05982-C05-01) and the Generalitat Valenciana (Ref: GRUPOS03/189) for their support.

References

[1] D. Applegate, R.E. Bixby, V. Chvátal & W. Cook (1995): "Finding cuts in the TSP". Technical report, DIMACS 95-05.

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

[3] E. Benavent, A. Carrotta, A. Corberán, J.M. Sanchis & D. Vigo (2003): "Lower Bounds and Heuristics for the Windy Rural Postman Problem". Technical Report TR03-2003. Department of Statistics and OR, University of Valencia (Spain). Submitted to *EJOR*.

[4] E. Benavent, A. Corberán, E. Piñana, I. Plana & J.M. Sanchis (2005): "New Heuristics for the Windy Rural Postman Problem".*Computers & Operations Research* 32, 3111-3128.

[5] P. Brucker (1981): "The Chinese Postman Problem for mixed graphs". *Proc. Int. Workshop. Lecture Notes in Computer Science* 100, 354-366.

[6] S. Chopra & G. Rinaldi (1996): "The Graphical Asymmetric Traveling Salesman Polyhedron: Symmetric Inequalities". *SIAM J. Discrete Math.* 9, 602-624.

[7] N. Christofides, V. Campos, A. Corberán & E. Mota (1981): "An Algorithm for the Rural Postman Problem". *Report IC.OR. 81.5.* Imperial College, London.

[8] N. Christofides, V. Campos, A. Corberán & E. Mota (1986): "An algorithm for the Rural Postman Problem on a directed graph". *Mathematical Programming Study* 26, 155-166.

[9] A. Corberán, A. Letchford & J.M. Sanchis (2001): "A Cutting Plane Algorithm for the General Routing Problem". *Mathematical Programming* 90, 291-316.

[10] A. Corberán, G. Mejía & J.M. Sanchis (2005): "New Results on the Mixed General Routing Problem". *Operations Research* 53, 363-376.

[11] A. Corberán, E. Mota & J.M. Sanchis (2005): "A Comparison of Two Different Formulations for Arc Routing Problems on Mixed Graphs". To appear in *Computers & Operations Research.*

[12] A. Corberán, I. Plana & J.M. Sanchis (2005): "On the Windy General Routing Polyhedron". In preparation.

[13] A. Corberán, I. Plana & J.M. Sanchis (2005): "Zigzag inequalities: A new class of facet-inducing inequalities for Arc Routing Problems". To appear in *Mathematical Programming.*

[14] A. Corberán, A. Romero & J.M. Sanchis (2003): "The Mixed General Routing Problem Polyhedron". *Mathematical Programming* 96, 103-137.

[15] A. Corberán & J.M. Sanchis (1994): "A polyhedral approach to the Rural Postman Problem". *European Journal of Operational Research* 79, 95-114.

[16] A. Corberán & J.M. Sanchis (1998):"The General Routing Problem: facets from the RPP and GTSP polyhedra". *European Journal of Operational Research* 108, 538-550.

[17] G. Cornuèjols, J. Fonlupt & D. Naddef (1985): "The traveling salesman problem on a graph and some related integer polyhedra". *Mathematical Programming* 33, 1-27.

[18] J. Edmonds & E. Johnson (1973):"Matching, Euler Tours and the Chinese Postman problem". *Mathematical Programming* 5, 88-124.

[19] H.A. Eiselt, M. Gendreau & G. Laporte (1995): "Arc-Routing Problems, Part 2: the Rural Postman Problem". *Operations Research* 43, 399-414.

[20] B. Fleischmann (1985): "A cutting-plane procedure for the traveling salesman problem on a road network". *European Journal of Operational Research* 21, 307-317.

[21] M. Grötschel & Z. Win (1992): "A Cutting Plane Algorithm for the Windy Postman Problem". *Mathematical Programming* 55, 339-358.

[22] M. Guan (1962): "Graphic Programming using odd and even points". *Chinese Mathematics* 1, 273-277.

[23] M. Guan (1984): "On the Windy Postman Problem". *Discrete Applied Mathematics*, 9, 41-46.

[24] A. Hertz, G. Laporte & P. Nanchen (1999): "Improvement procedures for the undirected Rural Postman Problem". *INFORMS Journal on Computing* 11, 53-62.

[25] A. Letchford (1997): "New inequalities for the General Routing Problem". *European Journal of Operational Research* 96, 317-322.

[26] E. Minieka (1979): "The Chinese Postman Problem for Mixed Networks". *Management Science* 25, 643-648.

[27] D. Naddef & G. Rinaldi (1991): "The Symmetric Traveling Salesman Polytope and its Graphical Relaxation: Composition of Valid Inequalities". *Mathematical Programming* 51, 359-400.

[28] C. Orloff (1974):"A fundamental problem in vehicle routing". *Networks* 27, 95-108.

[29] M. Padberg & M. Rao (1982): "Odd minimum cut-sets and b-matchings". *Mathematics of Operations Research* 7, 67-80.

[30] M. Padberg & G. Rinaldi (1987): "Optimization of a 532 City Symmetric Traveling Salesman Problem by Branch and Cut". *Operations Research Letters* 6, 1-7.

[31] C. Papadimitriou (1976):"On the complexity of edge traversing". *Journal of the ACM* 23, 544-554.

[32] I. Plana (2005): "The Windy General Routing Problem". *PhD Dissertation*, University of Valencia, Spain.

[33] Z. Win (1987): "Contributions to Routing Problems". *PhD Dissertation*, University of Augsburg, Germany.