A Branch-and-Price algorithm for the windy rural postman problem
Hasan-Murat Afsar, Nicolas Jozefowiez, Pierre Lopez

To cite this version:

HAL Id: hal-00676778
https://hal.archives-ouvertes.fr/hal-00676778
Submitted on 6 Mar 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Abstract. In this paper, we propose an exact solution method for the Windy Rural Postman Problem (WRPP). The motivation to study this problem comes from some real-life applications, such as garbage collecting in a predefined sector with hills, where the traversing or the servicing speed can change following the direction.

We present a Dantzig-Wolfe decomposition and a branch-and-price algorithm to solve the WRPP. To the best of our knowledge, Dantzig-Wolfe decomposition has never been used to solve that problem. The numerical results show that optimal solutions are found in a very reasonable amount of time on instances with up to 100 nodes and 180 edges.

Keywords: Branch-and-Price, Windy Rural Postman Problem

1. Introduction

The purpose of this paper is to investigate the use of a Dantzig-Wolfe decomposition and a branch-and-price algorithm to solve the Windy Rural Postman Problem (WRPP). The WRPP is an asymmetric variant of the Rural Postman Problem (RPP) [2]. The RPP is an arc routing problem that arises when a subset of the arcs must be visited. It is a general case of the Chinese Postman Problem (CPP) in which a circuit that visits all the arcs at least once must be found [2,14]. In the WRPP, the cost of traversing an edge depends on the traveling direction.
The problem can be described as follows: Given an undirected graph $G = (V, E \cup E_R)$, a cost function is associated with every edge/direction ($c_{ij}$ and $c_{ji}$ are the costs of traversing edge $[i, j]$ in either directions) and $E_R$ is the subset of required edges (i.e., needing service), which are supposed to be visited at least once. The WRPP then consists in finding a minimum cost circuit traversing every edge of $E_R$ at least once. Note that any edge can be visited many times without serving, but each required edge is served only once.

First introduced by Orlof [15], the RPP has been shown to be NP-hard by Lenstra and Rinnooy Kan, in the general case [12]. As the RPP is a special case of the WRPP, the WRPP is also NP-hard. Christofides et al. [3] proposed a branch-and-price method for the RPP. Branch-and-cut methods were also considered by Sanchis [16], Corberán and Sanchis [6], Letchford [13], and more recently by Ghiani and Laporte [8]. Benavent et al. [2] were inspired by a property obtained by Win [17] for the windy version of the CPP to solve the WRPP. Win showed that if the graph is Eulerian, the windy CPP is polynomial. A feasible solution for the WCPP is also feasible for the WRPP, since the windy CPP is a special case of the WRPP. By using a stochastic method to duplicate the edges or changing their costs in a given range, Benavent et al. then present a multi-start scatter search algorithm. They also detail a cutting-plane procedure to exactly solve or to obtain lower bounds of the WRPP [1].

In our work, during a field analysis for a waste collection project, the WRPP was used as a key component in order to find an optimal rotation of a truck. Our main contribution is the study of the WRPP by means of a Dantzig-Wolfe decomposition and its solution by means of a branch-and-price algorithm. The proposed algorithm is shown to be competitive with a state-of-the-art branch-and-cut algorithm from the literature [1] and appears as a good stepping stone for further improvement.

The paper is organized as follows. Section 2 introduces mathematical models obtained as results of the Dantzig-Wolfe decomposition and a mixed-integer model for the sub-problem. Section 3 details the branch-and-price algorithm and algorithms to solve the sub-problem. Computational results and conclusions follow in Sections 4 and 5, respectively.

## 2. Mathematical Models

Starting from the edge-based model proposed by Benavent et al. [1], we propose a Dantzig-Wolfe decomposition into a master problem and a sub-problem. We also present a lower bound based on the Lagrangian relaxation of required edge service constraints.

### 2.1. Edge-based model

Benavent et al. [1] give an extension to the WRPP of the mixed integer programming (MIP) model proposed by Grötschel and Win [9] for the windy postman problem. The integer decision variable $u_{ij}$ counts the number of times the edge
(i, j) is traversed from i to j. The set of the incident edges of vertex i is denoted by \( \delta(i) \). The sub-graph \( G_R \) induced by \( E_R \) is not necessarily connected. We denote by \( V_1, V_2, ..., V_p \) the connected components of \( G_R \); they are called R-connected components in [5].

Model 1

\[
C_{WRPP}^{*(1)} \triangleq \min \sum_{(i,j) \in E} (c_{ij}u_{ij} + c_{ji}u_{ji})
\]

s.t.:

\[
\begin{align*}
& u_{ij} + u_{ji} \geq 1 \quad \forall (i, j) \in E_R \\
& \sum_{(i,j) \in \delta(i)} (u_{ij} - u_{ji}) = 0 \quad \forall i \in V \\
& \sum_{(i,j) \in S, j \in V \setminus S} u_{ij} \geq 1 \quad \forall S = \bigcup_{k \in Q} V_k, Q \subset \{1, ..., p\} \\
& u_{ij}, u_{ji} \in \mathbb{N} \quad \forall (i, j) \in E
\end{align*}
\]

In Model 1, the objective is to minimize the total travel cost defined by Equation (1). Constraints (2) impose that every required edge is visited at least once. Constraints (3) are the flow conservation constraints. The connectivity between connected components is ensured by constraints (4); there must be at least one edge between any combination of connected components and the rest of the nodes. Note that constraints (4) are binding constraints and they significantly complicate the solution of the problem. Dantzig and Wolfe [7] propose a special reformulation and decomposition of the problem into a master and a sub-problem. Constraints (3) and (4) are considered by the sub-problem. The master problem takes into account only the set covering constraints (2).

2.2. Mathematical Model of the Restricted Master Problem

The master problem (MP) obtained by the Dantzig-Wolfe decomposition is a path-based model in which only the constraints related to the necessity of visiting all the required arcs are kept. Now, each variable represents a feasible closed-walk, i.e., a circuit passing at least once through each edge of \( E_R \). We will denote by \( P \) the set of all the feasible closed-walks. The length of a closed-walk \( p \in P \) will be denoted by \( c_p \). In spirit with the column generation approach we have adopted, we will only consider at a time a subset \( P' \subset P \). The restricted master problem (RMP) is formulated as follows. For each \( p \in P' \), let \( \lambda_p \geq 0 \) equal to 1 if and only if \( p \) is used. For each edge \( (i, j) \in E_R \), and \( \gamma_p^i \) is equal to 1 if and only if the closed-walk \( p \in P' \) serves \((i, j)\) by traveling from \( i \) to \( j \). Then, the restricted
master problem (RMP) is:

$$C^*_RMP \triangleq \min \sum_{p \in P'} c_p \lambda_p$$ \hspace{1cm} (6)

s.t.:

$$\sum_{p \in P'} \lambda_p = 1$$ \hspace{1cm} (7)

$$\sum_{p \in P'} (\gamma_{ij}^p + \gamma_{ji}^p) \lambda_p = 1 \hspace{0.5cm} \forall [i,j] \in E_R$$ \hspace{1cm} (8)

$$\lambda_p \geq 0 \hspace{0.5cm} \forall p \in P'.$$ \hspace{1cm} (9)

The objective function (6) minimizes the total cost of the chosen closed-walks. Only one closed-walk should be chosen (constraint (7)). Constraints (8) guarantee that every required edge in $E_R$ is used only once and in a unique direction.

2.3. Mixed Integer Model of the Sub-Problem

First, we must underline that, even if the graph is asymmetric and the cost of an edge depends on the travel direction, an edge has a unique dual variable independent from the direction. This is the main reason why we use edge notation instead of arc notation.

In the sub-problem, we are looking for closed-walks with negative reduced costs. Let $\pi_0$ and $\pi_{ij}$ be the dual variables associated to the constraints (7) and (8), respectively. We define the reduced cost $\bar{c}_p$ of a closed-walk $p$ as follows:

$$\bar{c}_p = c_p - \sum_{[i,j] \in E} (\gamma_{ij}^p + \gamma_{ji}^p) \pi_{ij} - \pi_0$$

We introduce a parameter $\beta_{ij}^p$ which counts the number of travels through edge $[i,j]$, from $i$ to $j$, with or without service, on a closed-walk $p$. Hence $\beta_{ij}^p - \gamma_{ij}^p$ is the number of times the vehicle traverses the edge $[i,j]$ without servicing. The cost of a closed-walk $p$ can be rewritten as:

$$c_p = \sum_{[i,j] \in E} (\beta_{ij}^p c_{ij} + \beta_{ji}^p c_{ji}).$$
For a given closed-walk \( p \), \( \pi_0 \) is constant. Thus, the reduced cost can be reformulated as:

\[
\bar{c}_p = \sum_{[i,j] \in E} (\beta^{ij}_p c_{ij} + \beta^{ji}_p c_{ji}) - \sum_{[i,j] \in E_R} (\gamma^{ij}_p + \gamma^{ji}_p) \pi_{ij} - \pi_0
\]

\[
= \sum_{[i,j] \in E_R} [\gamma^{ij}_p (c_{ij} - \pi_{ij}) + \gamma^{ji}_p (c_{ji} - \pi_{ij})] + \sum_{[i,j] \in E_R} [(\beta^{ij}_p - \gamma^{ij}_p) c_{ij} + (\beta^{ji}_p - \gamma^{ji}_p) c_{ji}]
\]

\[
+ \sum_{[i,j] \in E_{NR}} (\beta^{ij}_p c_{ij} + \beta^{ji}_p c_{ji}) - \pi_0
\]

To formulate the sub-problem, we replace the parameters \( \gamma^{ij}_p \) and \( \beta^{ij}_p - \gamma^{ij}_p \) by the binary variables \( y_{ij} \) which indicate whether edge \([i, j]\) is served on this closed-walk, and the integer variables \( z_{ij} \) which count the traversing of \([i, j]\) without servicing. The integer variable \( x_{ij} \) is the total number of traversing. An artificial variable \( f_{ij} \) for each \([i,j] \in E\) forces the walks to be connected. As previously, the set of the incident edges of vertex \( i \) is denoted by \( \delta(i) \). Each edge has two costs (distances) following the directions \( c_{ij} \) and \( c_{ji} \). A large number \( M \) is used as well in the formulation of the sub-problem.

Model 2

\[ C^{*,(2)}_{WRPP} \triangleq \min \sum_{[i,j] \in E_R} \{ y_{ij}(c_{ij} - \pi_{ij}) + y_{ji}(c_{ji} - \pi_{ij}) \} + \sum_{[i,j] \in E} (z_{ij}c_{ij} + z_{ji}c_{ji}) \] (10)

s.t.:

\[
y_{ij} + y_{ji} \leq 1 \quad \forall [i,j] \in E_R
\]

(11)

\[
y_{ij} + z_{ij} = x_{ij} \quad \forall [i,j] \in E
\]

(12)

\[
y_{ji} + z_{ji} = x_{ji} \quad \forall [i,j] \in E
\]

(13)

\[
\sum_{[i,j] \in E_R} (x_{ij} - x_{ji}) = 0 \quad \forall i \in V
\]

(14)

\[
f_{ij} \leq Mx_{ij} \quad \forall [i,j] \in E
\]

(15)

\[
f_{ji} \leq Mx_{ji} \quad \forall [i,j] \in E
\]

(16)

\[
\sum_{[i,j] \in \delta(i)} (f_{ij} - f_{ji}) = 1 \quad \forall i \in V \setminus \{s\}
\]

(17)

\[
\sum_{(s,j) \in \delta(s)} f_{sj} \geq 1
\]

(18)

\[
x_{ij}, x_{ji} \in \mathbb{R}, \; z_{ij}, z_{ji} \in \mathbb{Z} \quad \forall [i,j] \in E
\]

\[
y_{ij}, y_{ji} \in \{0, 1\}, \; f_{ij}, f_{ji} \in \mathbb{R} \quad \forall [i,j] \in E
\]

(19)

The objective function (10) minimizes the reduced cost of the closed-walk. As \( \pi_0 \) is constant at a given iteration for all the closed-walks, it is dropped from the
objective function which minimizes now the difference between the total distance and the sum of the dual values. Constraints (11) allow serving a required edge at most once, only in one direction. Total times of traveling through edges in one or other direction are defined by constraints (12) and (13). Every time the postman comes into a vertex, he must go out of it (flow conservation, constraints (14)). Constraints (15) and (16) put in relation flow of type $x$ and flow of type $f$: If there is a flow on an edge in one direction, then traversing counter direction should be greater than zero. There is a consumption of the flow at each vertex, except the source $s$ (constraints (17) and (18)); thanks to these constraints, the walk is connected. If an edge is not used in the optimal solution, corresponding counting and flow variables are equal to zero but there is always at least a couple of incident edges for all vertices. As the variables $y_{ij}$ and $z_{ij}$ are defined as binary and integer, respectively, $x_{ij}$ and $f_{ij}$ can be defined as real numbers (constraints (19)).

2.4. **LAGRANGIAN BOUND OF THE MASTER PROBLEM**

Lagrangian relaxation is a technique that works by removing hard constraints and putting them into the objective function, assigned with weights called Lagrangian multipliers. Each weight represents a penalty on the objective function if the particular constraint is not satisfied. We obtain the lower bound of the MP by pushing the constraints (8) to the objective function (6) with Lagrangian multipliers ($\pi_{ij}$):

$$\theta(\pi) = \min_{p \in P} \left\{ c_p + \sum_{[i,j] \in E} \pi_{ij} \left(1 - (\gamma_{ij}^p + \gamma_{ji}^p)\right)\right\}. \quad (20)$$

We denote by $\pi = (\pi_{ij})_{[i,j] \in E}$ the vector of dual values associated with constraints (8).

By reformulating equation (20), we obtain:

$$\theta(\pi) = \min_{p \in P} \left\{ \bar{c}_p + \sum_{e \in E_R} \pi_{ij} + \pi_0 \right\}. \quad (21)$$

The expression $\sum_{e \in E_R} \pi_{ij} + \pi_0$ is constant for a given Lagrangian vector $\pi$. Therefore, every time the minimum reduced cost ($\bar{c}_p^* = \min_{p \in P} \bar{c}_p$) is found, i.e., the sub-problem is solved to optimality, the Lagrangian lower bound is easily calculated. Having a valid lower bound of the MP can be useful to trigger an early stop of the column generation procedure. At each iteration, we can find out the gap between the best feasible solution found so far and the best Lagrangian lower bound ($\theta^*$), which can be written as follows:

$$\theta^* = \max_{\pi} \theta(\pi) = \max_{\pi} \left\{ \bar{c}_p^* + \sum_{e \in E_R} \pi_{ij} + \pi_0 \right\}. \quad (22)$$
At each iteration, if the sub-problem is solved to optimality and the minimum reduced cost calculated, the best Lagrangian lower bound ($\theta^*$) is updated if the new Lagrangian bound is better than the actual best Lagrangian lower bound. This update is in constant time and the lower bound is used as an indicator. It must be underlined that, when the column generation procedure terminates, the optimal solution of the linear relaxation of the restricted master problem is equal to the Lagrangian lower bound.

3. Column Generation Heuristics for the WRPP

A branch-and-price algorithm is proposed to solve the WRPP. The column generation heuristic is described in Section 3.1 and different algorithms to solve the sub-problem are detailed in Sections 3.2 and 3.3. The branching strategy is given in Section 3.4.

3.1. Column Generation Heuristic

The implementation of the standard column generation procedure for the WRPP is described as follows: Starting from an initial set $P'$ of closed-walks, which is initialized by the insertion heuristic described in Section 3.3, the RMP is solved. Then, we search for at most $K$ closed-walks with negative reduced cost. We limit the number of the closed-walks with reduced cost to prevent adding too many columns during the first iterations, where the dual variables are not yet stabilized. Negative reduced cost columns are first searched by means of a shortest path based heuristic (SPBH) presented in Section 3.2. If the algorithm is unable to find a negative reduced cost closed-walk $p$, we use the insertion heuristic detailed in Section 3.3. If this method also fails, we solve to optimality the MIP described in Section 2.3 by means of an MIP solver. Generated feasible closed-walks, which have a positive reduced cost, are kept in a pool to be used as candidate columns to be inserted in the RMP. We use the Lagrangian relaxation of the sub-problem to stop the search if the gap is closed with the best closed-walk found so far. The procedure comes to end when there is no more variable with negative reduced cost to add to the MP.

3.2. Shortest Path Based Approach

We search for a shortest elementary path on $G = (V,E)$. Each required edge $[i,j]$ is weighted by $c_{ij} - \pi_{ij}$ (or $c_{ji} - \pi_{ij}$, according to the direction). Naturally the weight of a non required edge is only its distance. The shortest path based approach will minimize the function $f(i,p)$ over a set of required edges not yet serviced. This function $f(i,p)$ is the reduced cost of the partial (and open) walk $p$ when the dual value $\pi_0$ is dropped. To extend a partial walk $p$ ending at node $k$ by a non-incident required edge $[i,j]$, we have:

$$f^*(j, p \cup [k, i] \cup [i, j]) = \min_{\bar{p} \in P \setminus p} f^*(k, p) + c_{ki} + c_{ij} - \pi_{ij}$$
where 

\[ f^*(s, \emptyset) = 0 \text{ and } f^*(s, p) = \bar{c}_p + \pi_0. \]

Figure 1 illustrates such extension of a partial walk \( p \).

\[ f^*(s, \emptyset) = 0 \text{ and } f^*(s, p) = \bar{c}_p + \pi_0. \]

\[ f^*(j, p \cup [k, j]) = \min_{j, i \not\in p} f^*(k, p) + c_{kj} - \pi_{kj}. \]

Dijkstra’s label fixing algorithm (Algorithm 1) is used to find the shortest path. Each label \( l_v \) is initialized to infinity, except the label of the source node; at each iteration, the label with the least cost is fixed and extended to the other nodes.

An alternative to this method is to weight the arcs by only dual values and to search for the longest elementary path.

**Algorithm 1** Shortest path based heuristic

\[
p = \{s\} ; \quad S_T = V \\
\text{for all } v \in S_T \text{ do} \\
\quad l_v = \infty \\
\text{end for} \\
\quad l_s = 0 \\
\text{while } S_T \neq \emptyset \text{ do} \\
\quad \text{Find } u = \arg \min_{v \in S_t} l_v \\
\quad S_T = S_T - \{u\} \\
\quad \text{for all } v \in S_T \text{ do} \\
\quad \quad \text{if } f(v, p \cup \{v\}) \leq l_v \text{ then} \\
\quad \quad \quad p = p \cup \{v\} \\
\quad \quad \quad l_v = f(v, p) \\
\quad \quad \text{end if} \\
\quad \text{end for} \\
\text{end while} \\
\text{Find } p = \arg \min_{v \in V} \{f(v, p) + c_{vs}\}
\]
The shortest path based method does not necessarily visit all the required edges. The following insertion algorithm is used to give a complete and feasible solution to the WRPP.

### 3.3. Insertion Heuristic

The required edges are listed in a decreasing order of their dual values. Then every edge in the list is inserted at the best position. Since the cost of an edge depends on the direction, during the insertion two directions of the edge are compared for every possible insertion point. The best insertion position is the one that causes the least increase in the reduced cost. Let us assume that edge \([i,j]\) is inserted, in the position \(\alpha\), between edges \([k,l]\) and \([m,n]\). Reduced cost increase \((\Delta(\bar{c}))\) is calculated as follows:

\[
\Delta(\bar{c}(\alpha)) = \min(c_{li} + c_{ij} + c_{jm} - \pi_{ij}, c_{lj} + c_{ji} + c_{im} - \pi_{ij}).
\]

If there is not an edge between nodes \(l\) and \(i\), \(c_{li}\) is the distance of the shortest path between them. After inserting all required edges, the closed-walk is locally improved by reversing or swapping edges.

The insertion algorithm (2) starts with a partial walk \(p\) which has only the source node and at each iteration, the edge with the highest dual value, which is not visited by \(p\), is inserted into the best position \((\alpha)\).

#### Algorithm 2 Insertion heuristic

1. Sort edges in a decreasing order of dual values \((L_T)\)
2. \(p = \{s\}\)
3. for \(\forall e \in L_T\) do
   1. Find \(\min \alpha \Delta(\bar{c})\)
   2. Insert \(e\) in the position \(\alpha\) to \(p\)
4. end for

### 3.4. Branching Strategy

Column generation is applied at every node of the tree generated by the branch-and-price algorithm. We branch when the flow over a non-required edge is not an integer, i.e., \(z_{ij} = \xi\) where \(\xi \not\in \mathbb{N}\). In that case two branches are created, with \(z_{ij} \geq \lfloor \xi \rfloor\) and \(z_{ij} \leq \lfloor \xi \rfloor\). These two constraints are easily added to the MIP of the sub-problem. On the other hand, such a constraint is almost impossible to verify with the insertion heuristic and the shortest path based approach. That is why, these two heuristic approaches are only used at the root node of the branch-and-price tree.
4. Experimental Results

We tested our algorithm on the instances of Christofides et al. [4] and the instances generated from Hertz et al. (1999) [11]. The RPP instances of Hertz et al. [10] were transformed in WRPP instances by Benavent et al. in [2]. The instances of Christofides et al. are small-size instances, whereas modified Hertz et al. instances are large.

The proposed method is written in the programming language JAVA. CPLEX 11.1 is used as to solve LP and MIP. Experiments are conducted on an Intel dual core 3.0 GHz with 3 GB of RAM.

Each data set consists of 6 instances and the detail of each data set is given in Table 1. The table gives the number of nodes $|V|$, of required edges $|E_R|$, of unrequired edges $|E_{NR}|$, and of all edges $|E|$). The column $|CC|$ represents the number of connected components of the sub-graph induced by required edges. On the last two columns, we give the average resolution mean time of each instance family obtained by our branch-and-price algorithm and by the branch-and-cut method of Benavent et al. [1] 1.

Almost all of the instances except three are solved to optimality by the proposed branch-and-price algorithm. These three instances are in HG5, HG9, and C21. The sub-problem for these instances cannot be solved in a reasonable amount of time. Therefore even a Lagrangian lower bound cannot be computed to give a gap and they are excluded from the calculation of the average solution times of HG5, HG9, and C21.

It is interesting to see that the data set with the greatest execution time does not correspond to the largest instances. Actually, five out of six instances in each set need an execution time of less than 5 seconds. One instance in each set takes more than 200 seconds. It should be noted that the execution time of the method varies almost 50 times on two different instances that have exactly the same number of nodes and same connections. The only difference between two instances of the same family is the cost of the edges.

One of the best methods found in the literature on WRPPs is the scatter search of Benavent et al. [2]. The best and worst mean gaps between their solution and the optimal solution, as well as the mean execution time following the parameters are given in Table 2 on the instances of Christofides et al. [4] (C) and the instances generated from Hertz et al. [11] (HG).

Our exact method obtains the optimal solution, in less than 5 seconds on the instances up to 180 edges except one instance which takes a little more than 200 seconds. All the instances are solved on the root node of the branch-and-price tree.

Another interesting point is that, the branch-and-price algorithm is much less sensitive to the number of connected components ($|CC|$) than the number of required edges.

---

1 on a Pentium III, clocked at 1.0 GHz
### Table 1. Data properties and execution mean times

| Instance set | $|V|$ | $|E_R|$ | $|E_{NR}|$ | $|E|$ | $|CC|$ | Time (sec.) | Time (sec.) |
|--------------|------|--------|--------|------|-------|-----------|-------------|
| HG1          | 60   | 41     | 64     | 105  | 20    | 2.0       | –           |
| HG2          | 68   | 49     | 70     | 119  | 21    | 4.0       | –           |
| HG3          | 64   | 44     | 72     | 116  | 21    | 5.0       | –           |
| HG4          | 82   | 73     | 75     | 148  | 14    | 3.0       | –           |
| HG5          | 92   | 77     | 88     | 165  | 19    | 50.3      | –           |
| HG6          | 91   | 82     | 80     | 162  | 12    | 3.0       | –           |
| HG7          | 97   | 113    | 61     | 174  | 5     | 4.7       | –           |
| HG8          | 100  | 107    | 73     | 180  | 10    | 6.0       | –           |
| HG9          | 97   | 109    | 66     | 175  | 7     | 4.9       | –           |
| C01          | 11   | 7      | 6      | 13   | 5     | 0.2       | 2.2         |
| C02          | 14   | 12     | 21     | 33   | 5     | 0.4       | 2.5         |
| C03          | 28   | 26     | 31     | 57   | 5     | 0.5       | 3.4         |
| C04          | 17   | 22     | 13     | 35   | 4     | 0.4       | 2.5         |
| C05          | 20   | 16     | 19     | 35   | 6     | 0.3       | 3.1         |
| C06          | 24   | 20     | 26     | 46   | 8     | 0.4       | 3.5         |
| C07          | 23   | 24     | 23     | 47   | 4     | 0.3       | 2.0         |
| C08          | 17   | 24     | 16     | 40   | 3     | 0.3       | 2.0         |
| C09          | 14   | 14     | 12     | 26   | 4     | 0.3       | 2.7         |
| C10          | 12   | 10     | 10     | 20   | 5     | 0.2       | 2.5         |
| C11          | 9    | 7      | 7      | 14   | 4     | 0.2       | 1.7         |
| C12          | 7    | 5      | 5      | 18   | 4     | 0.2       | 1.8         |
| C13          | 7    | 4      | 6      | 10   | 4     | 0.1       | 1.4         |
| C14          | 28   | 31     | 48     | 79   | 7     | 1.0       | 3.8         |
| C15          | 26   | 19     | 18     | 37   | 9     | 0.4       | 3.1         |
| C16          | 31   | 34     | 60     | 94   | 8     | 1.0       | 4.6         |
| C17          | 19   | 17     | 27     | 44   | 6     | 0.4       | 2.1         |
| C18          | 23   | 16     | 21     | 37   | 9     | 0.5       | 2.5         |
| C19          | 33   | 29     | 25     | 54   | 8     | 1.0       | 5.5         |
| C20          | 50   | 63     | 35     | 98   | 8     | 5.4       | 5.6         |
| C21          | 49   | 67     | 43     | 110  | 7     | 3.4       | 6.5         |
| C22          | 50   | 74     | 110    | 184  | 7     | 6.4       | 5.2         |
| C23          | 50   | 78     | 80     | 158  | 7     | 4.0       | 6.6         |
| C24          | 41   | 55     | 70     | 125  | 8     | 1.9       | 3.6         |

### Table 2. Scatter search performance of Benavent et al.

<table>
<thead>
<tr>
<th>Type of instance set</th>
<th>Best mean gap (%)</th>
<th>Worst mean gap (%)</th>
<th>Mean execution time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>0.27%</td>
<td>0.43%</td>
<td>0.68</td>
</tr>
<tr>
<td>HG</td>
<td>0.69%</td>
<td>1.11%</td>
<td>2.09</td>
</tr>
</tbody>
</table>
5. Conclusions

This paper proposes a Branch-and-Price algorithm for solving the Windy Rural Postman Problem. We find the optimal solution for all the instances tested except three. Another advantage is that the column generation procedure is very fast, for the instances up to 100 nodes and 180 edges of which more than 100 are required. An interesting future work would be to understand why the proposed method cannot give any result for three instances by observing in detail the features of these instances. Discovering the structure of the difficult instances can help us to identify and to modify other instances which could be transformed into easy ones. This can lead us to solve other and larger WRPP instances, which we cannot solve in a reasonable amount of time, until now.

Another idea is to find valid inequalities for the sub-problem and accelerate the solution of the Mixed Integer Program. The resulting Branch-and-Cut-and-Price algorithm would be more efficient.

Acknowledgement

This research was partly supported by Région Midi-Pyrénées and the authors gratefully acknowledge the support of this institution. Thanks are also due to the Editor and to the referee for his valuable comments.

References


