



CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

The Rural Postman Problem with Time Windows

Ingrid Marcela Monroy-Licht
Ciro-Alberto Amaya
André Langevin

November 2013

CIRRELT-2013-69

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palais-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

The Rural Postman Problem with Time Windows

Ingrid Marcela Monroy-Licht¹, Ciro-Alberto Amaya², André Langevin^{1,*}

¹ Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, C.P. 6079, succursale Centre-ville, Montréal, Canada H3C 3A7

² Departamento de Ingeniería Industrial, Universidad de Los Andes, Edificio Mario Laserna Cra 1 Este No 19A - 40 Bogotá, Colombia

Abstract. In this article we consider the Rural Postman Problem with Time Windows (RPPTW) for the undirected case. The problem occurs in the monitoring of roads for black-ice detection. We give different formulations and compare them on sets of instances adapted from the literature. We propose a cutting plane algorithm to solve the problem based on valid inequalities for the Traveling Salesman Problem with Time Windows and the Precedence Constrained Traveling Salesman Problem and we tested it on a set of real-life networks. Computational results show that our algorithm is able to solve to optimality instances with up to 104 required edges. At the end of the article we extend the formulations for the undirected case to the directed case.

Keywords. Rural postman problem, time windows, cutting plane algorithms, monitoring of roads.

Acknowledgements. This work was partly supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). This support is gratefully acknowledged.

Results and views expressed in this publication are the sole responsibility of the authors and do not necessarily reflect those of CIRRELT.

Les résultats et opinions contenus dans cette publication ne reflètent pas nécessairement la position du CIRRELT et n'engagent pas sa responsabilité.

* Corresponding author: Andre.Langevin@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2013

© Copyright Monroy-Licht, Amaya, Langevin and CIRRELT, 2013

1. Introduction

The Rural Postman Problem with Time Windows (RPPTW) involves finding the minimum-cost tour that goes through a set of required edges in a network. A vehicle leaves the depot, visits the required edges, and returns to the depot. A release time and a due time are given for each required edge. The tour is feasible if the visits are carried out during the defined time windows. Waiting times are allowed, i.e., the vehicle may arrive at any required edge earlier than its release time, but the service cannot start until the time window “opens.” Costs of service are associated with required edges and traversal costs with non-required edges. The vehicle may go through a required edge more than once, and the cost is normally lower when it does not service the edge.

The real-world application underlying this study is the monitoring of roads for black-ice detection. This activity is carried out by the Ministry of Transport in the province of Quebec from mid-October to mid-December. The goal is to check the state of roads and take measures to prevent accidents. During this period, black ice on roads is almost invisible to the users; timely detection avoids pedestrian falls or automobile accidents.

Currently, a patrol must cover a network and generate reports about the state of the roads. The available information refers to short-term weather forecast and a characterization of roads with high likelihood of black-ice formation; for example it is known that bridges and roads located near rivers are particularly susceptible to ice formation when certain meteorological conditions occur. The patrol determines the roads to be checked weighting their risk level and the meteorological conditions in the area where they are located. The weather forecast induces the time windows for monitoring the road segments over a large region; so the patrol should check the state of road segments of the network during established time intervals. Normally the patrol has enough time to visit all the roads defined previously in the schedule.

The RPPTW reduces to the Rural Postman Problem (RPP) when the time-window constraints are not taken into account, so it is NP-hard. Little attention has been paid to the RPPTW. To the best of our knowledge the works of Norbert and Picard (1994) and Kang and Han (1998) are the only two.

Norbert et al. (1994) introduce a heuristic algorithm for the RPPTW. In their problem the required arcs are of two types: arcs that must be visited during the morning and arcs that may be visited all day long. They propose a heuristic method based on the solution of two rural path problems and on the computation of appropriate penalties. Numerical results are not published. Kang and Han (1998) consider the problem as a multiobjective optimization problem because they allow arrival at the required arcs after the due times, which incurs a cost penalty. The objective is to reduce the total traveling cost and total penalty. The authors present a genetic algorithm and compare three crossover operators.

A similar problem, the RPP with deadline classes, has been studied by Letchford and Eglese (1998). They consider a single-vehicle arc routing problem in which the required edges are partitioned into a number of classes according to priorities, each class having its own deadline. An optimization algorithm is presented based on the use of valid inequalities as cutting planes. They tested the algorithm on a set of instances for the RPP from Corberán and Sanchis (1994) and found optimal solutions for all cases up to 67 required edges.

Our work addresses the undirected version of the problem. We assume that the costs of service are equal to the traversal costs, but our approaches could be easily modified if this is not the case. Our main is to present the problem for a real-life application and several ways to model it. Three formulations are presented: one where the decision variables explicitly express the number of times an edge is traversed and two based on graphs equivalent to the original one. We then explore the third formulation, obtained when we transform the original problem to a Traveling Salesman Problem with Time Windows and side constraints.

For the Traveling Salesman Problem (TSP), polyhedral approaches have been extremely successful (Fischetti and Toth, 1997, Jünger et al., 1995, Padberg and Rinaldi, 1991). Ascheuer et al. (2001) solve the Asymmetric TSP with Time Windows (ATSP-TW) by a branch-and-cut method; they solve in a satisfactory way real-world instances of the control of a stacker crane in a warehouse. We have chosen to use the polyhedral approach. The RPPTW is formulated as an integer linear program that is solved by a cutting plane algorithm.

The paper is organized as follows. Section 2 presents three different models for the undirected case. In Section 3 we summarize the valid inequalities that we use as cutting planes in our algorithm. We briefly describe the solution algorithm in Section 4. Section 5 outlines the computational experiments. In Section 6 an extension of the formulations is given for the directed version of the problem, and Section 7 provides concluding remarks.

2. Undirected RPPTW

We propose three different formulations for the problem. The first considers the classical point of view of formulations of arc routing problems, i.e., an edge can be traversed more than once in any feasible solution, and defines decision variables that explicitly express the number of times an edge is traversed. The other two formulations are based on transformed graphs. The first transformation considers the required edges as nodes and joins them by means of arcs that represent the shortest paths among them in the original graph. The second transformation considers the nodes incident to the required edges and connects them with an arc that again corresponds to their shortest paths in the original graph.

2.1. Model on the edges

This formulation is based on the work presented by Gueguen (1999). He proposes a MIP for the Capacitated Arc Routing Problem with Time Windows (CARP-TW). Apparently this is the only formulation on edges for the CARP-TW, i.e., based on the classical point of view; however, the author does not present numerical results. Basically, we modify Gueguen's formulation by adding a duplicate of each required edge to keep track of the direction in which the vehicle travels along these edges. This is necessary to guarantee conservation of flow on the nodes of the network.

Consider a graph $G(V, E)$ where V is the set of vertices and E is the set of edges. The required edges are the subset $E_R \subset E$. Let A be the set of edges that includes E , a duplicate i° of each required edge $i \in E_R$, and an artificial edge " e_0 " that represents the depot. The duplicate edges have the same cost and time windows as the originals. Let P be the set of pairs of edges (i, i°) such that i, i° correspond to the same required edge (the order i, i° is determined arbitrarily), and R the set that contains all $i \in E_R$ and

their duplicates. Additionally, \hat{c}_i is the traversal cost of edge i , \hat{T}_i is the traversal time of edge i , $[\hat{a}_i \ \hat{b}_i]$ is the time window for edge i ; and for the edge " e_0 " we set $\hat{a}_{e_0} = 0$. M is a large integer value and δ_i is the set of vertices incident to edge i . $m = |E_R| + 1$ is the maximum number of times an edge can be traversed when time windows are considered (Gueguen, 1999). The decision variables defined hereafter allow us to keep track of the number of times (each time corresponds to a "copy" of an edge) that the vehicle traverses each edge.

Let the decision variables $x_{ijkl} = 1$ if copy l of edge j is traversed immediately after copy k of edge i , and 0 otherwise; and let t_{ik} be the time to start traversing copy k of edge i . The formulation on the edges is as follows:

$$\min Z = \sum_{i \in A} \sum_{\substack{j \in A \\ \delta_j \cap \delta_i \neq \emptyset, j \neq i}} \sum_{k=1}^m \sum_{l=1}^m \hat{c}_i x_{ijkl} \quad (2.1)$$

subject to:

$$\sum_{\substack{j \in A \\ \delta_j \cap \delta_i \neq \emptyset, j \neq i}} \sum_{l=1}^m x_{ij1l} + \sum_{\substack{j \in A \\ \delta_j \cap \delta_{i^\circ} \neq \emptyset, j \neq i^\circ}} \sum_{l=1}^m x_{i^\circ j1l} = 1 \quad \forall (i, i^\circ) \in P \quad (2.2)$$

$$\sum_{\substack{j \in A \\ \delta_j \cap \delta_{e_0} \neq \emptyset, j \neq e_0}} \sum_{l=1}^m x_{e_0 j1l} \geq 1 \quad (2.3)$$

$$\sum_{\substack{i \in A \\ \delta_i \cap \delta_j \neq \emptyset, i \neq j}} \sum_{k=1}^m x_{ijkl} = \sum_{\substack{i \in A \\ \delta_i \cap \delta_j \neq \emptyset, i \neq j}} \sum_{k=1}^m x_{jilk} \quad \forall j \in A, l = 1, \dots, m \quad (2.4)$$

$$\sum_{\substack{j \in A \\ \delta_j \cap \delta_i \neq \emptyset, i \neq j, j \neq i^\circ}} \sum_{k=1}^m \sum_{l=1}^m x_{ijkl} + \sum_{\substack{j \in A \\ \delta_j \cap \delta_{i^\circ} \neq \emptyset, j \neq i^\circ, j \neq i}} \sum_{k=1}^m \sum_{l=1}^m x_{i^\circ jkl} \geq 1 \quad \forall (i, i^\circ) \in P \quad (2.5)$$

$$t_{ik} + \hat{T}_i \leq t_{jl} + M(1 - x_{ijkl}) \quad \forall i \in A, j \in A | \delta_i \cap \delta_j \neq \emptyset, i \neq j, j \neq e_0, k, l = 1, \dots, m \quad (2.6)$$

$$t_{i1} \geq \hat{a}_i \quad \forall i \in R \cup \{e_0\} \quad (2.7)$$

$$t_{i1} \leq \hat{b}_i \quad \forall i \in R \quad (2.8)$$

$$\sum_{l=1}^m x_{ijkl} \leq 1 \quad \forall i \in A, j \in A | \delta_i^{(-)} = \delta_j^{(+)}, i \neq j, k = 1, \dots, m \quad (2.9)$$

$$x_{ijkl} \in \{0,1\} \quad \forall i \in A, j \in A | \delta_i \cap \delta_j \neq \emptyset, i \neq j, k, l = 1, \dots, m \quad (2.10)$$

$$t_{ik} \in \mathbb{R}^+ \quad \forall i \in A, k = 1, \dots, m \quad (2.11)$$

The objective is to minimize the total traversal cost. Services are ensured by constraints (2.2). Constraint (2.3) forces the tour to start at the depot. Flow conservation is ensured by constraints (2.4). Constraints (2.5) avoid solutions with subcycles between any pair of edges in P that represents the same edge in A . Inequalities (2.6), (2.7), and (2.8) are the time-window constraints. Constraints (2.9), not in Gueguen (1999), allow us to reduce the number of equivalent solutions. Finally, constraints (2.10) and (2.11) define the decision variables.

This model is intractable for large networks because the number of copies of each edge could be, in the worst case, equal to the cardinality of the set of required edges plus one. Our tests have shown that for large networks the number of times the same edge is traversed never exceeds five, so we set $m = 5$ in our implementation.

2.2. Model on the required edges

Since the model on the edges (Section 2.1) is not practical, we propose an equivalent formulation. Let the original problem be defined on the graph $G = (V, E)$ as in Section 2.1. In this graph two required edges can be connected successively on a route in four ways, depending on the traversal direction of the two edges. This leads us to define the problem on a new graph $G_0 = (N, A_0)$.

For each required edge i in G we define two nodes i, i° in G_0 , where $i, i^\circ \in N$ represent the two possible directions in which edge i in G can be traversed. Each arc of A_0 connects a node $i \in N$ with a node $j \in N$ if they do not correspond to the same edge in G . The cost of the arc that connects node i to node j in G_0 is equal to the cost of the edge represented by i plus the length of the shortest path in G from the final node of the edge represented by i to the initial node of the edge represented by j , according to the traversal directions.

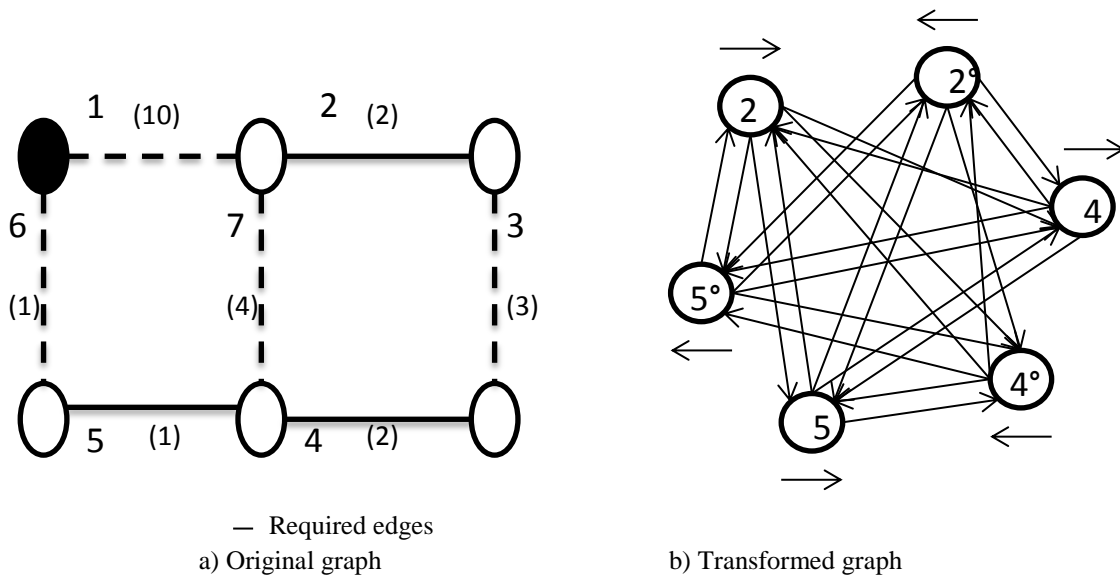


Figure 1. Transformed graph for model on required edges

Figure 1 shows an example of the transformation. The original graph is presented in a). The depot is located at the black node; the edge indices are shown near each edge; and the traversal costs are in parentheses. There are three required edges (2, 4, and 5). The transformed graph is illustrated in b). Its nodes are labeled with the same number as their corresponding required edges. Arrows over and under

the nodes indicate the traversal direction that each node represents. Note that pairs of nodes corresponding to the same required edge are not connected.

Finally we add an artificial node labeled "0" to represent the depot. We join "0" to all nodes in N by means of two arcs with a cost equal to the length of the shortest path in G from the depot to the edges, and from the edges to the depot respectively, again according to the traversal direction. Table 1 indicates the cost of the arcs of the transformed graph. The time window for each node is the same as for the corresponding edge.

Table 1. Costs of the transformed graph – Model on the required edges

To From	0	2	2°	4	4°	5	5°
0	0	6	8	2	4	1	2
2	10	0	0	7	5	8	7
2°	8	0	0	6	7	7	6
4	6	7	5	0	0	5	4
4°	4	6	7	0	0	3	2
5	3	5	6	1	3	0	0
5°	2	6	7	2	4	0	0

In the transformed graph G_0 we look for a minimum-cost tour that visits one of the two nodes that represent the same required edge. The tour starts and ends at the depot, and the visits must satisfy the time windows.

Let us consider the set C that includes the pairs of nodes (i, i°) , where $i, i^\circ \in N$, such that i, i° represent the same required edge. We consider the following parameters: c_{ij} is the traversal cost from node i to node j , T_{ij} is the traversal time from node i to node j , $[a_i, b_i]$ is the time window for node i , and $M_{ij} = \max \{ b_i + T_{ij} - a_j, 0 \}$. We set $a_0 = 0$ for the depot node.

We define the decision variables x_{ij} to be equal to 1 if node j is serviced immediately after node i , and 0 otherwise, and t_i to be the arrival time at node i . The formulation on the required edges is as follows:

$$\min Z = \sum_{i \in NU\{0\}} \sum_{\substack{j \in NU\{0\} \\ j \neq i, \\ (i,j) \notin C}} c_{ij} x_{ij} \tag{2.12}$$

subject to:

$$\sum_{\substack{j \in NU\{0\} \\ j \neq i, i^\circ}} (x_{ij} + x_{i^\circ j}) = 1 \quad \forall (i, i^\circ) \in C \tag{2.13}$$

$$\sum_{\substack{i \in N \cup \{0\} \\ i \neq j, j^\circ}} (x_{ij} + x_{ij^\circ}) = 1 \quad \forall (j, j^\circ) \in C \quad (2.14)$$

$$\sum_{j \in N} x_{0j} = 1 \quad (2.15)$$

$$\sum_{j \in N} x_{j0} = 1 \quad (2.16)$$

$$t_i + T_{ij} \leq t_j + M_{ij}(1 - x_{ij}) \quad \forall i \in N, j \in N \cup \{0\} \mid i \neq j, (i, j) \notin C \quad (2.17)$$

$$a_i \leq t_i \quad \forall i \in N \cup \{0\} \quad (2.18)$$

$$t_i \leq b_i \quad \forall i \in N \quad (2.19)$$

$$x_{ij} \leq \sum_{\substack{k \in N \cup \{0\} \\ k \neq j}} x_{jk} \quad \forall i \in N \cup \{0\}, j \in N \cup \{0\} \mid i \neq j, (i, j) \notin C \quad (2.20)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in N \cup \{0\}, j \in N \cup \{0\} \mid i \neq j, (i, j) \notin C \quad (2.21)$$

$$t_i \in \mathbb{R}^+ \quad \forall i \in N \cup \{0\} \quad (2.22)$$

The objective is to minimize the total traversal cost. Constraints (2.13) and (2.14) ensure that only one node is included in the tour for each pair of nodes that represent the same required edge. Constraints (2.15) and (2.16) force the tour to start and end at the depot. The time-window constraints are (2.17), (2.18), and (2.19). Constraints (2.20) guarantee flow conservation. Finally, the decision variables are defined in (2.21) and (2.22).

2.3. Model on the nodes

We now propose a transformation from the original problem to an equivalent problem on nodes. We define a new graph $G_1 = (N_1, A_1)$, where all the vertices incident to the required edges in the original graph $G = (V, E)$ are included in the set of nodes N_1 . It should be pointed out that if a vertex of V is incident to more than one required edge in G , then this vertex will have as many copies in N_1 as the number of incident required edges in G . A_1 is the set of arcs that connects the nodes of N_1 . The cost of an arc that joins node i to node j is equal to the cost of the edge that starts at node i plus the length of the shortest path in G from the final vertex of that edge to the initial vertex j of the other edge. We set the cost to zero when nodes i and j represent vertices incident to the same edge in G or when $i = j$.

We obtain a complete directed graph. Figure 2 shows an example of the transformation. The original graph is presented in a). The vertices are numbered, the depot is located at the black vertex, and the numbers in parentheses represent traversal costs. Figure 2b) illustrates the transformed graph. Note that there are two nodes labeled "4" and "4^o" because they represent vertex "4" of a), which is incident to two required edges.

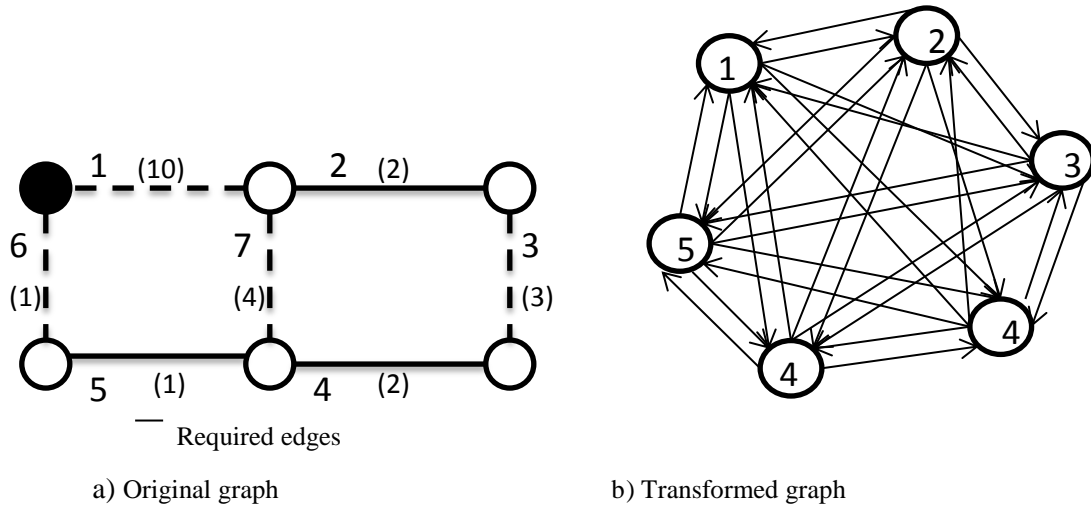


Figure 2. Transformed graph for model on nodes

We add an artificial node "0" to represent the depot. We join "0" to all nodes in N_1 by means of two arcs with costs equal to the length of the shortest path in G from the depot to the vertices, and from the vertices to the depot respectively. The time windows of each required edge in G are assigned to its incident nodes.

In the transformed graph, we look for a minimum-cost tour that starts and ends at the depot and satisfies the time windows for all nodes. Additionally, two nodes incident to the same required edge must be placed one after the other in the tour sequence. Table 2 shows the cost matrix for the transformed graph.

Table 2. Costs of the transformed graph – Model on the nodes

To From	0	1	2	4	3	5	4°
0	0	6	8	2	4	1	2
1	10	0	0	7	5	8	7
2	8	0	0	6	7	7	6
4	6	7	5	0	0	5	4
3	4	6	7	0	0	3	2
5	3	5	6	1	3	0	0
4°	2	6	7	2	4	0	0

Let us define C_0 as the set of pairs of nodes that are incident to the same required edge, and U as the set of nodes that includes one of the two nodes incident to the same required edge. Furthermore, let the parameters c_{ij} , T_{ij} , $[a_i b_i]$, a_0 , and M_{ij} and the decision variables x_{ij} and t_i be as defined in Section 2.2. The formulation on the nodes is as follows:

$$\min Z = \sum_{i \in N_1 \cup \{0\}} \sum_{j \in N_1 \cup \{0\} | j \neq i} c_{ij} x_{ij} \tag{2.23}$$

subject to:

$$x_{ii^\circ} + x_{i^\circ i} = 1 \quad \forall (i, i^\circ) \in C_0 \quad (2.24)$$

$$\sum_{\substack{j \in N_1 \cup \{0\} \\ j \neq i}} x_{ij} = 1 \quad \forall i \in N_1 \cup \{0\} \quad (2.25)$$

$$\sum_{\substack{i \in N_1 \cup \{0\} \\ i \neq j}} x_{ij} = 1 \quad \forall j \in N_1 \cup \{0\} \quad (2.26)$$

$$t_i + T_{ij} \leq t_j + M_{ij}(1 - x_{ij}) \quad \forall i \in N, j \in N_1 \cup \{0\} \mid i \neq j \quad (2.27)$$

$$a_i \leq t_i \quad \forall i \in U \cup \{0\} \quad (2.28)$$

$$t_i \leq b_i \quad \forall i \in U \quad (2.29)$$

$$t_i = t_{i^\circ} \quad \forall (i, i^\circ) \in C_0 \quad (2.30)$$

$$x_{ij} \in \{0,1\} \quad \forall i \in N_1 \cup \{0\}, j \in N_1 \cup \{0\} \mid i \neq j \quad (2.31)$$

$$t_i \in \mathbb{R}^+ \quad \forall i \in N_1 \cup \{0\} \quad (2.32)$$

The objective is to minimize the total traversal cost. Constraints (2.24) are related to the required services. Constraints (2.25) and (2.26) ensure that each node is visited. The time-window constraints are (2.27), (2.28), (2.29), and (2.30). The decision variables are defined in (2.31) and (2.32).

The model on the edges (Section 2.1) is intractable for large graphs and uses a large real value M that creates weak relaxations and numerical difficulties in the solution methods. The models based on the transformations have fewer variables and constraints. They also use large integer values M_{ij} , but these can be bounded to minimum values that allow us to find feasible solutions. The variables of model on edges need two index k, l to identify the number of times each edge could be traversed in a deadheading mode if it computes on the minimum-distance objective. We cannot associate a unique starting and completion time for each edge. We augment the graph by $k \times l$ times whereas the transformations are models better bounded because they compact the graph by taking into account only information related to required edges. The other edges (no required) are considered only for getting the shortest path among required edges. Therefore, the variables do not need an extra index to identify the number of times edges are traversed in a deadheading mode and we can make the assumption that elements representing the required edges are visited no more than once. The model on the nodes (Section 2.3) results in the Asymmetric TSP with Time Windows and side constraints. Therefore, existing algorithms for the Asymmetric TSPTW can be used.

3. Valid inequalities

We focus on the model on the nodes (Section 2.3), taking advantage of its structure. This model has elements of the Precedence Constrained Asymmetric TSP (PC-ATSP). Polyhedral approaches to solve problem instances to optimality are known to work well for the PC-ATSP (Ascheuer et al., 2000), and, as already mentioned, for the TSP. We study some of the known valid inequalities with respect to the two problems that are also valid for the formulation (2.23)–(2.32). In the following, we summarize the classes of inequalities that we use in our solution method.

Notation

Given the set of arcs A_f , for any arc set $W \subseteq A_f$ we define $x(W) := \sum(x_{ij} \mid (i, j) \in W)$. Given the set of nodes V_f that includes the depot "0", for any two node sets $S, T \subseteq V_f$ we define $(S:T) := \{(i, j) \in A_f \mid i \in S, j \in T\}$ and write $x(S:T)$ for $x((S:T))$.

Lifted t-bounds. Desrochers and Laporte (1991) observed that the bounds of the t -variables (see inequalities 2.28 and 2.29) can be strengthened. Indeed, let $a_{ji} = \max\{0, a_j - a_i + T_{ji}\}$ and $b_{ij} = \max\{0, b_i - b_j + T_{ij}\}$. Then the inequalities

$$a_i + \sum_{\substack{j=1 \\ |j \neq i}}^n a_{ji} x_{ji} \leq t_i \quad \forall i \in V_f \setminus \{0\} \quad (3.1)$$

$$b_i - \sum_{\substack{j=1 \\ |j \neq i}}^n b_{ij} x_{ij} \geq t_i \quad \forall i \in V_f \setminus \{0\} \quad (3.2)$$

are valid for the formulation (2.23)–(2.32).

Strengthened MTZ-inequalities. Desrochers and Laporte (1991) propose a lifted version of the MTZ subtour-elimination constraints (2.27). Let $\bar{a}_{ji} = \max\{T_{ji}, a_i - b_j\}$ and $M_{ij} \geq b_i + T_{ij} - a_j$. Then for all $i, j = 1, \dots, n, i \neq j$ the inequality

$$t_i + T_{ij} - (1 - x_{ij})M_{ij} + (M_{ij} - T_{ij} - \bar{a}_{ji}) x_{ji} \leq t_j \quad (3.3)$$

is valid for the formulation (2.23)–(2.32).

According to Desrochers and Laporte (1991), when precedence relations exist, the MTZ-inequalities can be further strengthened. Assume $i < j$. Since i must be scheduled before j , we have $t_i \leq t_j$, and the inequality

$$t_i + T_{ij} x_{ij} \leq t_j \quad (3.4)$$

is also valid. If $b_i + T_{ij} \leq a_j$ holds, inequality (3.4) can be strengthened to

$$t_i + T_{ij} x_{ij} \leq a_j \quad (3.5)$$

Subtour elimination constraints. We include the subtour elimination constraints, since they are the best known inequalities for the Asymmetric Traveling Salesman polytope (Balas et al., 1995). These inequalities $x(S : S) \leq |S| - 1$ can be written in the equivalent cut form

$$x(S : \bar{S}) \geq 1 \quad \forall S \neq \emptyset, S \subset V_f \quad (3.6)$$

where $\bar{S} := V_f \setminus S$, and (3.6) is valid for the formulation (2.23)–(2.32).

The Predecessor/Successor inequalities. The PC-ATSP is a relaxation of the ATSP-TW. We use some valid inequalities for the PC-ATSP that allow us to strengthen the subtour elimination inequalities (3.6). Balas et al. (1995) introduced these classes of inequalities.

For $S \subseteq V_f \setminus \{0\}$, $\bar{S} := V_f \setminus S$, the predecessor inequality (π -inequality)

$$x(S \setminus \pi(S) : \bar{S} \setminus \pi(S)) \geq 1 \quad (3.7)$$

and the successor inequality (σ -inequality)

$$x(\bar{S} \setminus \sigma(S) : S \setminus \sigma(S)) \geq 1 \quad (3.8)$$

are valid for the formulation (2.23)–(2.32).

For any given $i, k \in V_f \setminus \{0\}$ such that $\pi(i) \neq \emptyset$, $\sigma(k) \neq \emptyset$, and any $S \subset V_f$ such that $i, k \in S$, the inequalities

$$x(S \setminus \pi(i) : \bar{S} \setminus \pi(i)) \geq 1 \quad (3.9)$$

$$x(\bar{S} \setminus \sigma(k) : S \setminus \sigma(k)) \geq 1 \quad (3.10)$$

are called weak π - and weak σ -inequalities respectively.

4. Solution algorithm

We implement the following algorithm to solve the Undirected RPPTW.

4.1. Data preprocessing

Data preprocessing is important for efficient implementations. It allows the construction of tighter equivalent formulations of the problems, such that no optimal solution of the original problem is lost and each solution of the tighter problem corresponds to a solution of the original problem.

The structures of the formulations on the required edges (Section 2.2) and on the nodes (Section 2.3) permit such a preprocessing procedure. It is based on the work of Ascheuer et al. (1999). We tighten the time windows iteratively until no more changes are made. We then identify precedence relations, fix variables permanently, and detect infeasible paths of size two and three to reduce the set of variables.

We now present the separation procedures for the classes of valid inequalities (3.6)–(3.10).

4.2. Cutting plane algorithm

Initial linear program. We solve the relaxation of the model on nodes (2.23)–(2.32), i.e., when the decision variables x_{ij} are restricted to be nonnegative and less than or equal to one. Constraints (3.1) and (3.2) are included in the initial model instead of constraints (2.28) and (2.29) because the former are stronger. We also include the strengthened MTZ-inequalities (3.3), (3.4), and (3.5) instead of the MTZ-inequalities (2.7) when possible.

Separation routines. Let (x^*, t^*) be a solution where x^* is fractional. We want to identify a member of a family F of valid inequalities listed in Section 3. for the formulation on the nodes that is violated by x^* or else show that x^* satisfies all members of F . The implemented separation procedures are an adaptation of routines described in the literature.

- Subtour elimination constraints: For the cutset inequalities (3.6) we can solve the separation problem by computing the connected component T that includes the depot in the graph G^* induced by $x_{ij}^* > 0$. If this component does not include all the nodes in V_f , the subtour elimination constraint is violated by x^* , and we obtain the set S that includes all nodes in T . This procedure detects inequalities (3.6) that are violated only in the case where there is no path in G^* from the depot to any $j \in V_f \setminus \{0\}$.
- Predecessor inequalities: We implement the exact separation algorithm presented by Balas et al. (1995) for the separation problem of predecessor inequalities. Although this algorithm detects only a violated weak π -inequality, if one exists, rather than a stronger π -inequality of the class (3.7), the detected violated inequality (3.9) can be replaced with a strictly stronger violated inequality of the class (3.7) when we include $\pi(S)$ instead of $\pi(j)$. If we apply the algorithm for $j \in V_f \setminus \{0\}$ such that $\pi(j) = \emptyset$, we detect the known cutset inequality, and obtain an algorithm that simultaneously solves the separation problem for both the subtour elimination inequalities and the π -inequalities.
- The successor inequalities: With an analogues procedure to Balas et al. (1995) we can detect if x^* violates a weak σ -inequality. For any fixed j with $\sigma(j) \neq \emptyset$, delete $\sigma(j)$ from V_f and in the resulting network with arc capacities x_{ij}^* try to send one unit of flow from node 0 to node j . If this is possible, all inequalities (3.10) associated with the given j are satisfied by x^* ; otherwise the minimum capacity identified by the failed attempt to send a unit of flow specifies the σ -inequality most violated by x^* . We reverse the sets S and \bar{S} , i.e., S will be replaced by \bar{S} and vice versa. As in the previous case, if a violated inequality (3.10) is found, it is replaced with a strictly stronger violated inequality of the class (3.8), when we include $\sigma(S)$ instead of $\sigma(j)$.

Steps for the separation routine

- Subtour elimination constraint
- “Shrinking”: The separation algorithm for the predecessor/successor inequalities implies the computation of the maximum flow for each pair $i, j \in V_f$. We use “shrinking” procedures (Padberg and Rinaldi, 1990) to reduce the problem size and to avoid as many maximum-flow calculations as possible. “Shrinking” checks whether or not certain nodes lie on the same side of a minimum-capacity cut. If the results are positive, the subset is contracted or “shrunk” to a single node. We contract nodes i and j if they are incident to the same required edge. Also, we contract nodes i and j if $x_{ij}^* = 1$ in the fractional solution x^* .
- Predecessor inequalities: We use the separation problem for the predecessor inequalities, and we simultaneously check the subtour elimination constraints when there is one connected component in the fractional solution x^* .

- Successor inequalities: We use the separation problem for the successor inequalities, and we simultaneously check the subtour elimination constraints when there is one connected component in the fractional solution x^*

We generate at most one cutting plane for each separation routine per iteration. The linear problems are solved using standard parameters of CPLEX 12.4.0.

4.3. Solution of the MIP program

We stop the cutting plane algorithm whenever the last 10 linear problems produce no improvement in the lower bound, or in case the improvement is less than 0.1%, or when the running time reaches three hours. In those scenarios the decision variables x_{ij} are restricted to be binary, and we solve the problem using the callable library of CPLEX 12.4.0.0. with its default parameters except the threads set on value 1.

5. Computational results

In this section we describe the results of the comparison of the models on a set of generated instances and the performance of our cutting plane algorithm which was tested also on a set of instances based on the real network of the Estrie region in Quebec. Our implementation is coded in Python 2.6 and runs on a 2.38 GHz AMD 250.

5.1. Generated instances

There are no published benchmark instances for the undirected RPPTW. We modified the CARP-TW instances of Wøhlk (2005). The author combines five values for the number of nodes ($\{10,13,20,40,60\}$) and the number of edges ($\{15,23,31,69,90\}$) and generates five different graphs for each combination.

Basically, we selected some required edges randomly and found a path through all of them using the nearest-neighbor heuristic. Then, we established the time-window intervals by reducing and extending by 10, 20, and 30% the values of the arrival time given by the heuristic. By combining the percentage of required edges $\{10,30,50\}$ and the width of the time windows $\{10,30,50\}$ we generated 225 instances.

All the instances can be downloaded from <http://ftpprof.uniandes.edu.co/~pylo/inst/RPPTW/instances.htm>.

5.2. Instances of Estrie network

We tested the cutting plane algorithm in a real undirected network that represents a part of the Estrie administrative region in Quebec. The network has 1472 km in total, 140 nodes and 187 edges. We simulated nine weather forecasting for one day with 4 or 5 time slots. If any edge is located in a time slot where rain is presented, the edge will be required to be visited in its respective time slot. Figure 3 shows an example of simulated weather forecasting for Estrie region. We defined the cost of traverse as the length of the road multiplied by a fractional number in order to get scalar representation, and the time of traverse proportionally to this value.

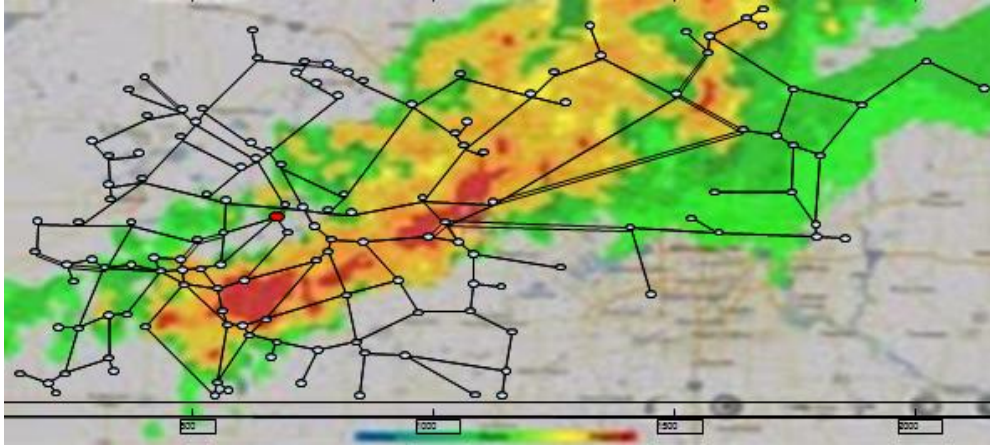


Figure 3. Estrie network – Weather forecast with 5 time slots

5.3. Preprocessing

As noted earlier, the structures of the formulations on the required edges (Section 2.2) and on the nodes (Section 2.3) allow data preprocessing. Table 3 shows the effect of data preprocessing, giving the average percentage of removed variables. In general, there was a considerable reduction in the problem size. When the time windows are tighter more variables can be fixed.

Table 3. Reduction in number of variables

T.W. Width	Model on the required edges	Model on the nodes
10	47.44%	40.15%
30	36.96%	30.43%
50	26.09%	21.88%

5.4. Preliminary test and cutting plane algorithm

In this section we present a summary of the presented models performance as well the cutting plane algorithm performance. We solved the models using CPLEX 12.4.0 with the callable library, setting a running time of up to three hours excluding the data-preprocessing time.

We compare only the models based on the transformations because the model on the edges (Section 2.1) is too large even when the number of copies of required edges is small. Table 4 summarizes the results for subsets of instances grouped by size. However detailed results for each instance are available at <http://ftpprof.uniandes.edu.co/~pylo/inst/RPPTW/instances.htm>. Column *TW* lists the different time-window widths, n is the number of nodes, e is the number of edges, $|R|$ is the average number of required edges, N is the total number of instances in the set, sol is the number of problems solved to optimality, and t is the average running time in seconds.

The model on the nodes (Section 2.2) is superior to the model on the required edges (Section 2.3) because it finds more optimal solution and optimal solutions for harder instances, and the computational times are smaller.

The results for the cutting plane algorithm are summarized in the follow columns. Column *gap* shows the average gap (%) given by the aggregation of cuts with respect to the optimal solution. Columns *mic* and *mac* show the minimum and maximum number of added cuts. Finally, *ave t* and *t max* present the average and maximum running times.

Our algorithm was able to solve 222 of 225 instances. The cutting plane approach solved 78 problems to optimality when the decision variables x_{ij} were restricted to be less than or equal to one. We solved to optimality 10 of the hardest instances (60 nodes, 90 edges, 45 required edges, and time windows width equal to 50 with an average computational time of 814.9 seconds. Detailed results for each instance are available at <http://ftpprof.uniandes.edu.co/~pylo/inst/RPPTW/instances.htm>

Table 4. Models comparison and cutting plane algorithm

TW	n	e	R	N	Model on the required edges		Model on the Nodes		Cutting plane algorithm (Model on nodes)					
					sol	t	sol	t	sol	gap	mic	mac	ave t	t max
10	10	15	5	9	9	0.003	9	0.004	9	0	0	2	0.057	0.19
30				9	9	0.016	9	0.010	9	0	0	5	0.08	0.13
50				9	9	0.018	9	0.018	9	1.2	0	22	0.21	0.82
10	13	23	7.4	27	27	0.016	27	0.011	27	0.2	0	10	0.21	0.81
30				27	27	0.267	27	0.069	27	1.6	0	17	0.61	2.85
50				27	27	0.286	27	0.098	27	2.6	0	22	0.50	1.99
10	20	31	10	9	9	0.050	9	0.028	9	0.1	0	6	0.66	2.31
30				9	9	4.901	9	1.641	9	2.7	0	19	1.30	7.27
50				9	9	8.491	9	0.590	9	4.6	0	17	1.82	6.53
10	40	69	21	18	18	51.84	18	2.43	18	1.1	4	25	11.18	25.19
30				18	15	1085.69	17	824.08	18	3.5	2	25	17.11	83.02
50				18	14	285.88	17	510.49	17	5.2	0	24	764.52	6675.22
10	60	90	27	12	11	405.02	12	104.31	12	2.1	5	27	17.83	60.23
30				12	7	361.78	10	1398.49	11	1.2	0	25	168.83	1659.91
50				12	5	6028.02	8	2354.40	11	4.0	3	24	1101.85	6030.17

The results of the cutting plane algorithm on the set of real instances are summary in table 5. Column *O.F.* presents the value of the objective function, *suc* the number of cuts for subtour elimination added, *pc* the cuts for precedence relations and *sc* the cut for successor relations. We were able to solve 5 of the 9 instances in less than 3,5 minutes. Instances that do not show values were not solved by the algorithm.

Table 5. Cutting plane on real instances

Instance	T.W	R	O.F.	suc	pc	sc	gap	t
Inst-00	Tight	74	--	--	--	--	--	--
Inst-01	Intermediate	74	259,7	2	10	11	7.6	165,46
Inst-02	Wide	74	--	--	--	--	--	--
Inst-03	Tight	104	289,2	2	11	11	2.3	202,29
Inst-04	Intermediate	104	--	--	--	--	--	--
Inst-05	Wide	104	--	--	--	--	--	--
Inst-06	Tight	93	299,7	2	11	10	7	193,41
Inst-07	Intermediate	93	299,7	2	11	9	7	208,75
Inst-08	Wide	93	299,7	1	11	11	7	137,76

6. Directed case

In this section we present two formulations for the directed case, which are extensions of the formulations for the undirected problem. The first is a formulation on the arcs, i.e., the decision variables express the number of times an arc is traversed. The second is based on a transformed graph, where the required arcs are connected by an arc that represents the shortest path among them. We carry out some preliminary tests to evaluate the performance of the models.

6.1. Model on the arcs

As in the undirected case, this formulation is based on the work presented by Gueguen (1999).

Consider a directed graph $G_d(V_d, D)$, where V_d is the set of vertices, $0 \in V_d$ represents the depot, and D is the set of arcs. Let \hat{A} be the set that includes D and two artificial arcs " a_l " and " a_e " leaving and entering the depot respectively. \hat{R} is the set of required arcs plus " a_l " and " a_e ". Furthermore, we define the following parameters: $\hat{m} = |\hat{R}| + 1$ is the maximum number of times an arc can be traversed in a feasible solution, \hat{c}_i is the traversal cost of arc i , \hat{T}_i is the traversal time of arc i , $[\hat{a}_i \hat{b}_i]$ is the time window for arc i , M is a large real value, $\delta_i^{(+)}$ is the end vertex of arc i , and $\delta_i^{(-)}$ is the initial vertex of arc i .

The decision variables for this formulation are defined as $x_{ijkl} = 1$ if copy l of arc j is traversed immediately after copy k of arc i , and 0 otherwise, and t_{ik} indicates the arrival time at copy k of arc i . The model is given below:

$$\min Z = \sum_{i \in \hat{A}} \sum_{j \in \hat{A}} \sum_{k=1}^{\hat{m}} \sum_{l=1}^{\hat{m}} \hat{c}_i x_{ijkl} \quad (6.1)$$

$$\delta_j^{(-)} = \delta_i^{(+)}$$

subject to:

$$\sum_{j \in \hat{A}} \sum_{l=1}^{\hat{m}} x_{ij1l} \geq 1 \quad \forall i \in \hat{R} \quad (6.2)$$

$$\delta_j^{(-)} = \delta_i^{(+)}$$

$$\sum_{i \in \hat{A}} \sum_{k=1}^{\hat{m}} x_{ijkl} = \sum_{i \in \hat{A}} \sum_{k=1}^{\hat{m}} x_{jilk} \quad \forall j \in \hat{A}, \quad l = 1, \dots, \hat{m} \quad (6.3)$$

$$\delta_i^{(+)} = \delta_j^{(-)} \quad \delta_i^{(-)} = \delta_j^{(+)}$$

$$t_{ik} + \hat{T}_i \leq t_{jl} + M(1 - x_{ijkl}) \quad \forall i \in \hat{A}, j \in \hat{A} | \delta_i^{(-)} = \delta_j^{(+)}, \delta_i^{(+)} \neq \{a_e\},$$

$$k, l = 1, \dots, \hat{m} \quad (6.4)$$

$$t_{i1} \geq \hat{a}_i \quad \forall i \in \hat{R} \quad (6.5)$$

$$t_{i1} \leq \hat{b}_i \quad \forall i \in \hat{R} | i \notin a_e \quad (6.6)$$

$$\sum_{l=1}^{\hat{m}} x_{ijkl} \leq 1 \quad \forall i \in \hat{A}, j \in \hat{A} | \delta_i^{(-)} = \delta_j^{(+)}, k = 1, \dots, \hat{m} \quad (6.7)$$

$$x_{ijkl} \in \{0,1\} \quad \forall i \in \hat{A}, j \in \hat{A} \mid \delta_i^{(-)} = \delta_j^{(+)}, k, l = 1, \dots, \hat{m} \tag{6.8}$$

$$t_{ik} \in \mathbb{R}^+ \quad \forall i \in \hat{A}, \quad k = 1, \dots, \hat{m} \tag{6.9}$$

The objective function minimizes the total traversal cost. Services are ensured by constraints (6.2). Constraints (6.3) define the flow conservation. Inequalities (6.4), (6.5), and (6.6) are the time-window constraints. Inequalities (6.7) reduce the set of equivalent feasible solutions. Finally, the decision variables are defined in (6.8) and (6.9).

As in the undirected case, the set of constraints (6.7) is added to obtain a strengthened formulation, and we set $\hat{m} = 5$ in our implementation.

6.2. Model on the required arcs

This formulation is equivalent to the previous one; it reduces the size of the problem by considering only the network information related to the required arcs. Let the original problem be defined on the directed graph $G_d = (V_d, D)$ described in Section 8.1. If $\hat{R} \subset D$ is the set of required arcs without the artificial arcs "a_l" and "a_e", the formulation on the required arcs is defined on the graph $\hat{G}_d = (\hat{N}_d, \hat{A}_d)$ with $|\hat{R}|$ nodes in \hat{N}_d . Each node in \hat{N}_d corresponds to a required arc in \hat{R} , and each arc in \hat{A}_d represents the length of the shortest path in G_d between a pair of required arcs. \hat{G}_d results in a complete graph.

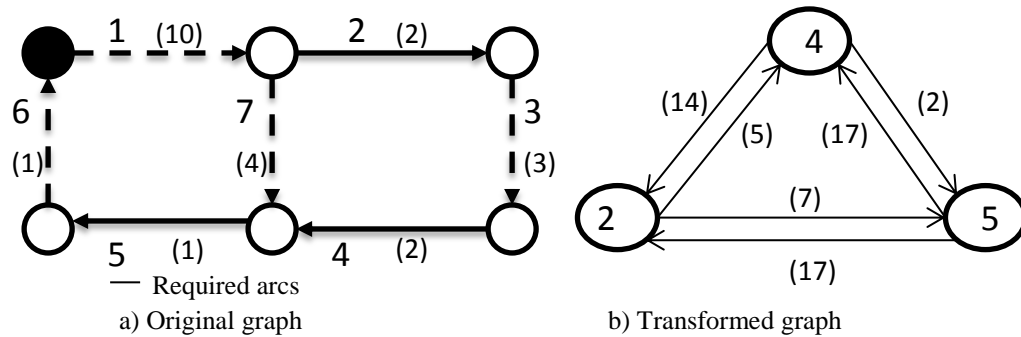


Figure 4. Transformed graph for model on required arcs

Figure 4 shows an example of the transformation. In the original graph presented in a) the depot is located at the black node. The numbers in parentheses represent traversal costs, and the other numbers are the arc indices. There are three required arcs (2, 4, and 5). The graph in b) with three nodes is complete. The nodes are labeled with the same number as their respective required arcs, and the distances are indicated in parentheses.

An artificial node "0" is added to represent the depot. We connect this node with each node in \hat{G}_d by means of two arcs, one entering and one leaving the depot. Finally, each node in \hat{G}_d adopts the time window corresponding to that for the arc that it represents.

In the transformed graph we look for a minimum-cost tour that starts and ends at the depot and satisfies the time-window constraints for each node. Given the parameters c_{ij} , T_{ij} , $[a_i b_i]$, and M_{ij} , and the decision variables x_{ij} and t_i defined in Section 2.2, the formulation is as follows:

$$\min Z = \sum_{i \in \dot{N}_d \cup \{0\}} \sum_{j \in \dot{N}_d \cup \{0\} | j \neq i} c_{ij} x_{ij} \quad (6.10)$$

subject to:

$$\sum_{\substack{j \in \dot{N}_d \cup \{0\} \\ j \neq i}} x_{ij} = 1 \quad \forall i \in \dot{N}_d \cup \{0\} \quad (6.11)$$

$$\sum_{\substack{i \in \dot{N}_d \cup \{0\} \\ i \neq j}} x_{ij} = 1 \quad \forall j \in \dot{N}_d \cup \{0\} \quad (6.12)$$

$$t_i + T_{ij} \leq t_j + M_{ij}(1 - x_{ij}) \quad \forall i \in \dot{N}_d, j \in \dot{N}_d \cup \{0\} / i \neq j \quad (6.13)$$

$$a_i \leq t_i \quad \forall i \in \dot{N}_d \cup \{0\} \quad (6.14)$$

$$t_i \leq b_i \quad \forall i \in \dot{N}_d \quad (6.15)$$

$$x_{ij} \in \{0,1\} \quad \forall i, j \in \dot{N}_d \cup \{0\} | i \neq j \quad (6.16)$$

$$t_i \in \mathbb{R}^+ \quad \forall i \in \dot{N}_d \cup \{0\} \quad (6.17)$$

The formulation corresponds to the ATSP-TW. The objective is to minimize the total traversal cost. Inequalities (6.11) and (6.12) are the assignment constraints. The time-window constraints are (6.13), (6.14), and (6.15). Finally, the decision variables are defined by (6.16) and (6.17).

To solve this model we refer to Ascheuer et al. (2001), who solved the ATSP-TW using a branch-and-cut method with satisfactory results.

6.3. Preliminary results

To evaluate the performance of the proposed formulations, we tested the models on two sets of instances.

We adapted a set of instances for the directed RPP presented by Campos (1995). The author presents 60 instances with different numbers of nodes (80, 160, and 240). We use only those with 80 and 160 nodes. The number of arcs ranges from 211 to 539, and the required arcs from 16 to 127. We implemented a process similar to that in the undirected case, and we varied the width of the time windows, i.e., $\{10,30,50\}$. We thus created 120 instances for the directed case. All the instances can be downloaded from <http://ftpprof.uniandes.edu.co/~pylo/inst/RPPTW/instances.htm>

The model on the arcs (Section 6.1) did not find any solution in less than three hours. We therefore generated some smaller random planar graphs with the aim of identifying the size of problem that this model can solve.

The three largest instances (with the highest number of required arcs) solved to optimality (in less than three hours) have the following characteristics: $\{70, 142, 71, 10\}$, $\{100, 180, 18, 30\}$, and $\{42, 71, 36, 50\}$ for respectively the number of nodes, the number of arcs, the number of required arcs, and the width of the time window. Table 6. shows the summary of results for this set of instances.

Detailed results for each instance are available at <http://ftpprof.uniandes.edu.co/~pylo/inst/RPPTW/instances.htm>. As it was mentioned before our interest points to the undirected case; the transformation to the model on the required arcs (Section 6.2) emerges to the TSP with time windows and this is the most accurate way to deal with the problem because again we obtain a formulation with less variables and better bounded. Our aim in this case is to evaluate the problem size solved for the model on arcs.

Table 6. Model on the arcs

<i>TW</i>	$\% R $	<i>n</i>	<i>a</i>	$ R $	<i>N</i>	<i>sol</i>	<i>t</i>
10	10	41.57	70.64	7.57	14	9	75.85
	30	35.29	59.64	18.35	17	8	67.28
	50	35.29	59.41	29.94	17	7	2.23
30	10	41.57	70.64	7.57	14	11	32.88
	30	35.29	59.64	18.35	17	8	20.20
	50	35.29	59.41	29.94	17	6	86.64
50	10	41.57	70.64	7.57	14	9	30.65
	30	35.29	59.64	18.35	17	9	16.95
	50	35.29	59.41	29.94	17	9	64.93

7. Conclusions

We have introduced several formulations for the undirected and directed RPPTW, and we have tested them on instances adapted from the literature and on a real network.

The results show that the models on the arcs and edges are not practical because they are too large and use the “big M.” For the directed case the model on the arcs could solve instances with up 36 requirements and wide time windows.

We propose two transformations for the undirected case. Preliminary results show that the model on the nodes is superior to the model on the required edges, i.e., it finds twelve more optimal solutions, and the running times are smaller.

In the undirected version of the problem, we exploited the formulation called “Model on the nodes”. The resulting problem is an Asymmetric TSP with Time Windows and side constraints. This model is solved using a cutting plane algorithm. On one hand, we were able to solve to optimality 222 of 225 generated instances in less than two hours. We solved to optimality 10 of the 12 hardest instances (60 nodes, 90 edges, 45 required edges, and time-window width equal to 0.5) in less than 15 minutes on average. On the other hand, we solved 5 of 9 instances of a real network with up 104 required edges in less than 3.5 minutes.

For the directed case the problem is transformed to an equivalent Asymmetric TSP with Time Windows. Existing methods for large instances, such as cutting plane algorithms, can be used.

Future research will develop a branch-and-cut algorithm to solve the problem. We will study efficient strategies to decide on which variables to branch and we hope to improve the performance of the

cutting plane algorithm. We also believe that metaheuristics should be explored to get good solutions in a short time when the dynamic case of the black-ice detection problem is considered, i.e., when the time windows or the road segments to visit vary over time.

References

- [1] N. Ascheuer, M. Fischetti, and M. Grötschel, Solving the Asymmetric Traveling Salesman Problem with time windows by branch-and-cut, *Math. Prog.* 90(A) (2001), 475-506
- [2] N. Ascheuer, M. Jünger, and G. Reinelt, A branch and cut algorithm for the asymmetric Traveling Salesman Problem with Precedence constraints, *Computational Optimization and Applications* 17(1) (2000), 61-84
- [3] N. Ascheuer, M. Fischetti, and M. Grötschel, Solving the Asymmetric Traveling Salesman Problem with Time Windows by branch-and-cut, Technical report Takustraße D-14195, Konrad-Zuse-Zentrum für informationstechnik Berlin, Germany, 1999
- [4] E. Balas, M. Fischetti, and W. Pulleyblank, The precedence constrained asymmetric traveling salesman polytope, *Math. Prog.* 68 (1995), 241-265
- [5] V. Campos, A computational study of several heuristics for the DPPP, *Computational optimization and applications*, 4 (1995), 67-77
- [6] A. Corberán and J.M. Sanchis, A polyhedral approach to the Rural Postman Problem, *European Journal of Operational Research*, 79 (1994), 95-114
- [7] M. Desrochers and G. Laporte, Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints, *Operations Research Letters*, 10(1) (1991), 27-36
- [8] M. Fischetti and P. Toth, A polyhedral approach to the asymmetric traveling salesman problem, *Management Science*, 43(11) (1997), 1520-1536
- [9] C. Gueguen. Méthodes de résolution exacte pour les problèmes de tournées de véhicules. Doctoral dissertation, Laboratoire Productique Logistique, École Centrale Paris, France, 1999
- [10] M. Jünger, G. Reinelt, and G. Rinaldi, "The traveling salesman problem," *Handbooks in Operations Research and Management Science*, Ch. 4, Vol 7: Network Models, M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser (Editors), Elsevier, 1995, pp. 225-330
- [11] M.-J. Kang and C.-G. Han, "Comparison of crossover operators for Rural Postman Problem with Time Windows," *Soft Computing in Engineering Design and Manufacture*, P. K. Chawdhry, R. Roy, and R. K. Pant (Editors), Springer, 1998, pp. 259-267
- [12] A.N. Letchford and R.W. Eglese, The Rural Postman Problem with deadline classes, *European Journal of Operational Research*, 105 (1998), 390-400
- [13] Y. Nobert and Y. Picard, A heuristic algorithm for the Rural Postman Problem with Time Windows, *Meetings2 INFORMS*, Detroit, 1994

- [14] M. Padberg, and G. Rinaldi, An efficient algorithm for the minimum capacity cut problem, *Math. Prog.* 47 (1990), 19-36
- [15] M. Padberg, and G. Rinaldi, A branch and cut algorithm for the resolution of large-scale Symmetric Traveling Salesman Problems, *SIAM Review* 33 (1991), 60-100
- [16] S. Wøhlk, Contributions to Arc Routing, Doctoral dissertation, Faculty of Social Sciences, University of Southern Denmark, 2005