European Journal of Operational Research 258 (2017) 1131–1142

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

The windy rural postman problem with a time-dependent zigzag option

Jenny Nossack[b,c], Bruce Golden[a], Erwin Pesch[b,c,*], Rui Zhang[d]

[a] Robert H. Smith School of Business, University of Maryland, College Park, MD 20742, USA
[b] Department of Management Information Science, University of Siegen, 57068, Siegen, Germany
[c] Center for Advanced Studies in Management (CASiM), HHL, 04109, Leipzig, Germany
[d] Leeds School of Business, University of Colorado, Boulder, CO, 80303, USA

ARTICLE INFO

Article history:
Received 17 March 2015
Accepted 4 September 2016
Available online 10 September 2016

Keywords:
Windy rural postman problem
Time window
Routing

ABSTRACT

In this research, we focus on the windy rural postman problem with the additional option to zigzag street segments during certain times of the day. If a street is narrow or traffic is light, it is possible (and often desirable) to service both sides of the street in a single pass by zigzagging. However, if a street is wide or traffic is heavy, we must service the street by two single traversals. For some streets, we further impose the restriction that they may only be zigzagged at specific times of the day, e.g., in the early morning when there is virtually no traffic. Real-life applications arise, among others, in trash collection and newspaper delivery. This specific arc routing problem combines two classes of problems known from the literature, arc routing problems with zigzag options and arc routing problems with time dependencies. We present and discuss two (mixed) integer programming formulations for the problem at hand and suggest exact solution approaches. Furthermore, we analyze the effects of zigzag and time window options on the objective function value and test our solution approaches on real-world instances.

1. Introduction and Problem Description

There have been many papers published on arc routing topics over the last several decades; see Corberán and Laporte (2014) for a comprehensive treatment of this research area. In this paper, we focus on one important class of arc routing problems, called the windy rural postman problem (WRPP), with several practical extensions.

In the WRPP, some (not necessarily all) of the streets in a network must be serviced and the street distances are asymmetric. Furthermore, we consider the additional option to zigzag street segments during certain times of the day. If a street is narrow or traffic is light, it is possible (and often desirable) to service both sides of the street in a single pass by zigzagging. However, if a street is wide or traffic is heavy, we must service the street by two single traversals. For some streets, we further impose the restriction that they may only be zigzagged at specific times of the day, e.g., in the early morning when there is virtually no traffic. This variation was first suggested by Golden (2013). We point out that, in this paper, we consider time windows with upper bounds only (lower bounds are always 0). This reflects the real-world environment in which zigzagging could only happen in the early morning hours.

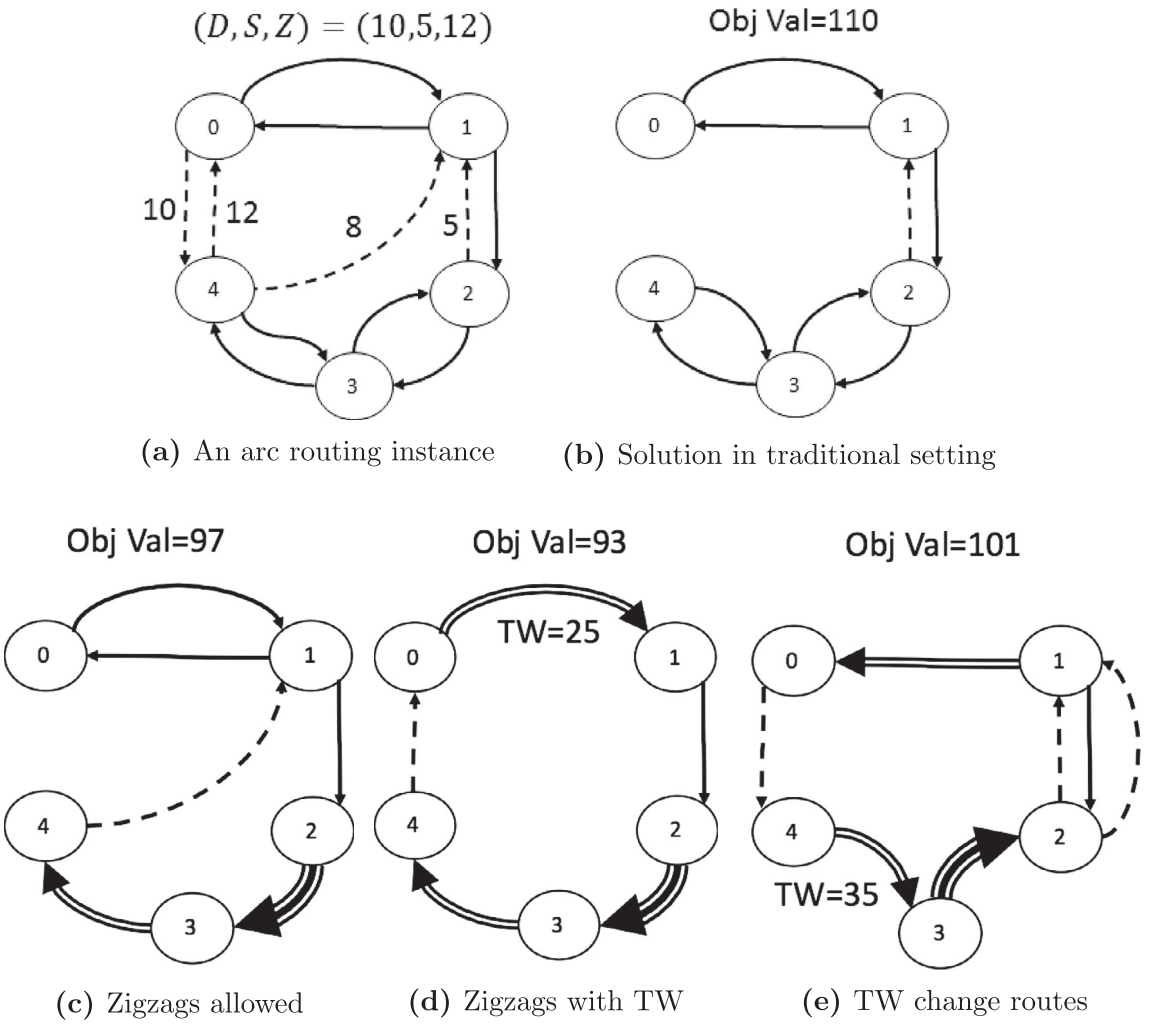
Real-life applications arise commonly in trash collection and newspaper delivery. This specific arc routing problem combines two classes of problems known from the literature, arc routing problems with zigzag options and arc routing problems with time dependencies. We present and discuss two (mixed) integer programming formulations for the problem at hand and suggest exact solution approaches. Furthermore, we analyze the effects of zigzag and time window options on the objective function value and test our solution approaches on real-world instances.

Based on our experiments and analysis, we make at least two key observations. The first is that this is a very computationally challenging problem, especially when there are numerous time windows and/or the time windows are narrow in width. The second is that the presence of zigzag options, both with and without time windows, can result in dramatic route changes and cost savings.

We now present an example in Fig. 1 to motivate the use of zigzags with and without time windows. In Fig. 1(a), we have an arc routing problem instance. There are five nodes. Node 0 is the

* Corresponding author at: Department of Management Information Science, University of Siegen, 57068 Siegen, Germany.
E-mail addresses: jenny.nossack@hhl.de (J. Nossack), bgolden@rhsmith.umd.edu (B. Golden), erwin.pesch@uni-siegen.de (E. Pesch), rui.zhang@colorado.edu (R. Zhang).

http://dx.doi.org/10.1016/j.ejor.2016.09.010
0377-2217/



(a) An arc routing instance (b) Solution in traditional setting

(c) Zigzags allowed (d) Zigzags with TW (e) TW change routes

Fig. 1. A motivating example (TW: time windows).

depot. Solid lines represent arcs which require service. For simplicity, in this instance, all solid lines have travel time $D = 10$, normal service time $S = 5$, and service time in zigzag (zigzag time) $Z = 12$. Dashed lines are arcs over which one can travel. The number besides a dashed line is its travel time. For instance, for arc (0, 4), the travel time is 10. If we solve this instance as a traditional arc routing problem, the optimal route is shown in Fig. 1(b) and its objective value is 110. Notice that arcs (0, 4), (4, 0), (4, 1) are not used in this solution although they can be shortcuts to access node 0. Now, if we require zigzagging on segment {2, 3} (one segment $\{i, j\}$ consists of two arcs (i, j) and (j, i)) and allow segment {3, 4} to be zigzagged, the route in Fig. 1(c) is now optimal and the objective value is 97. Here, the double-line arc and the triple-line arc represent zigzagging an edge which is zigzag possible and zigzag required, respectively. Furthermore, if segment {0, 1} can be zigzagged before time 25, we can obtain an even better route (see Fig. 1(d)). The objective value is reduced to 93. Lastly, these zigzags with time windows can change a route dramatically. If we allow the same three segments to be zigzagged, but a time window is imposed on segment {3, 4}, instead of segment {0, 1}, then a very different result is obtained. For example, if we can zigzag on segment {3, 4} before time 35, then the optimal route is changed to be the one in Fig. 1(e) with objective value 101.

This is the first paper to present models and solution approaches for an arc routing problem that combines zigzag options and time-dependencies. Furthermore, we study the impact of zigzag options and zigzag time, based on real-world data. The plan for the remainder of this paper is as follows. A review of the literature is provided in Section 2. Mathematical formulations are presented in Section 3. Our computational study is presented in Section 4. In Section 5, we discuss our conclusions and mention some ideas for future research.

2. Literature review

The WRPPZTW (windy rural postman problem with zigzags and time windows) combines two variants of arc routing problems, arc routing problems with zigzag options and arc routing problems with time dependencies/constraints. In what follows, we will summarize the relevant literature of both variants; we present a literature overview in Fig. 2.

A limited amount of literature has been published on arc routing problems with time dependencies. These dependencies relate, e.g., to service and travel times or costs that change over time or to time windows that are imposed to denote when a service on a required arc has to start or end (see Fig. 2). Most of these arc routing problems are addressed from a node routing perspective. A common modeling approach in node routing problems with time constraints is to incorporate a time variable for each customer to capture the starting time of its service (e.g., see Toth & Vigo,

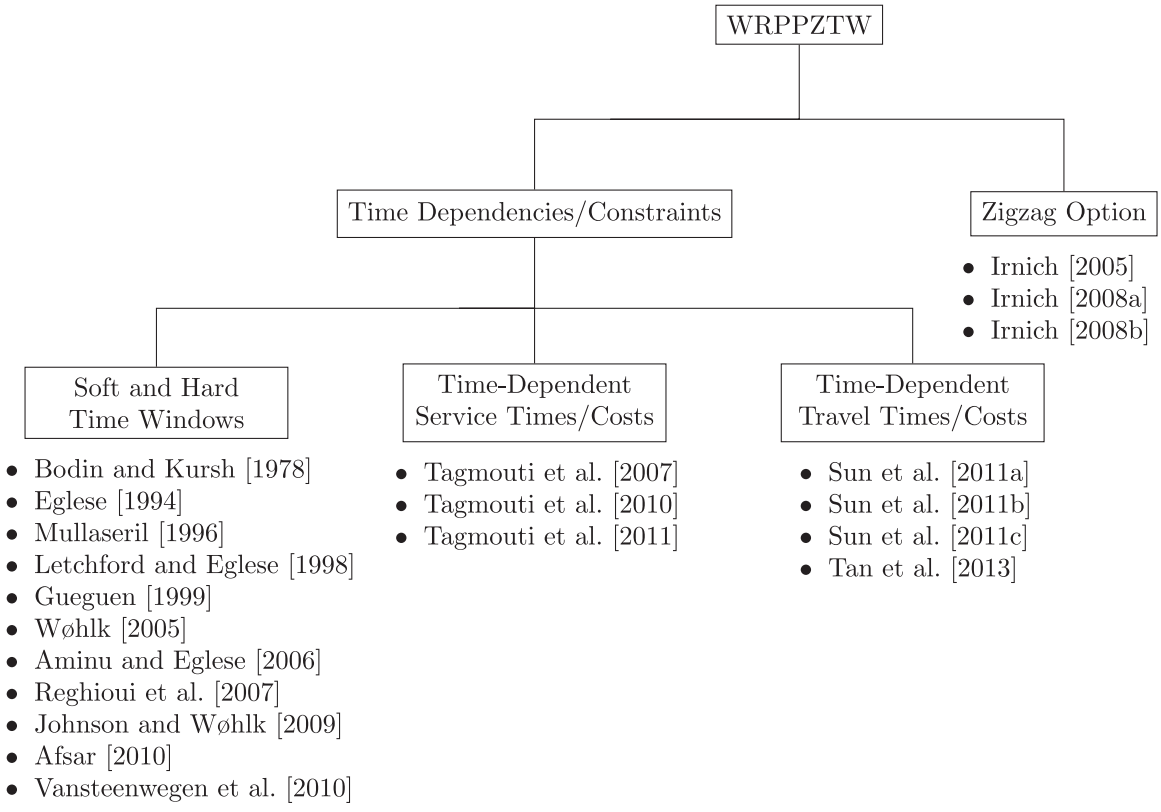


Fig. 2. Literature overview.

2002). Mullaseril (1996) points out the modeling difficulty of incorporating equivalent time variables into an arc routing formulation: "Each arc is serviced exactly once but can be traversed an additional number of times in deadheading mode if so required by the minimum distance objective. Thus, a trip may pass through a given pair of nodes more than once in that route and consequently such nodes would have more than one time corresponding to the start of traversal of an arc from that node. Hence we cannot associate a unique starting and completion time for an arc". Thus, a common modeling approach is to transform an arc routing problem into a node routing problem. After such a transformation, deadheading and multiple traversals are hidden in shortest paths that are used to connect two consecutive services. Then, the solution approaches known from the node routing literature can be applied.

Arc routing problems with time dependencies are encountered, e.g., in street sweeping (Bodin & Kursh, 1978) and winter gritting (Eglese, 1994; Tagmouti, Gendreau, & Potvin, 2007). Bodin and Kursh (1978) address an arc routing problem that arises when streets have to be swept by mechanical brooms. Due to parking regulations, some streets can only be swept at certain times of the day. The authors present "cluster first – route second" and "route first – cluster second" heuristics to solve this so-called street sweeper problem. Eglese (1994) considers the routing of winter gritting vehicles where streets have to be de-iced, e.g., by salt. Streets are categorized based on their time of service, i.e., some streets have to be serviced within two hours, some within four hours, etc. This problem is addressed as a capacitated arc routing problem with time restrictions and multiple depots and is solved by a two-stage heuristic. Letchford and Eglese (1998) study a similar arc routing problem and refer to it as a rural postman problem with deadline classes. Here, arcs are partitioned into classes where each class has an individual service deadline. Real-life applications arise, e.g., in parcel delivery and winter gritting. The authors generate valid inequalities and apply them in a customized cutting plane approach. Mullaseril (1996), in his dissertation, studies a capacitated rural postman problem with time windows on a directed graph. He transforms the problem into an equivalent node routing problem with time windows and applies a path-scanning algorithm to the latter. The dissertation of Gueguen (1999) considers the same problem on undirected graphs, proposes an integer program formulation and a further transformation into a node routing problem. The dissertation of Wøhlk (2005) also studies the capacitated rural postman problem with time windows on undirected graphs. The author suggests two model formulations, one arc routing and one node routing formulation. In addition, she presents several heuristics (augment-merge heuristic, path-scanning heuristic, and a preferable neighbor heuristic), a metaheuristic (dynamic programming with simulated annealing) and an exact solution approach based on column generation. Johnson and Wøhlk (2009) introduce two column generation approaches and one heuristic procedure to solve the capacitated arc routing problem with time windows on undirected graphs to optimality and near-optimality, respectively. Aminu and Eglese (2006) consider a Chinese postman problem with time windows. The authors propose two constraint programming formulations, one from an arc routing and one from a node routing perspective. Reghioui, Prins, and Labadi (2007) study the capacitated rural postman problem with time windows on undirected graphs and present a greedy randomized adaptive search procedure (GRASP) combined with path relinking. Tagmouti et al. (2007) address a capacitated arc routing problem with time-dependent service costs. The problem is motivated from winter gritting and is transformed into a node routing problem with time-dependent service costs. The latter is solved by a column generation approach. In a subsequent study (Tagmouti, Gendreau, & Potvin, 2010), the authors investigate the same problem from an arc routing perspective and solve it by a variable neigh-

borhood descent heuristic. This solution approach is further extended in Tagmouti, Gendreau, and Potvin (2011) to solve a dynamic version of the problem. Afsar (2010) studies the capacitated arc routing problem on undirected graphs with soft time windows, where time windows can be violated by paying appropriate penalties. The author proposes a branch-and-price algorithm to solve problem instances to optimality. Vansteenwegen, Souffriau, and Sörensen (2010) consider a so-called mobile van problem that is motivated by a real-life application where mobile mapping vans are routed to take pictures of streets and road signs. Since taking pictures towards the sun is undesirable, the problem includes time considerations. This problem is addressed as capacitated arc routing problem with soft time windows, is transformed into an equivalent node routing problem, and is solved by a hybrid metaheuristic. Recently, several papers have considered arc routing problems with time-dependent travel times, i.e., where the travel time of an arc depends on the time of day. Sun, Tan, Guangjian, and Wang (2011b) propose a column generation approach, Sun, Tan, and Qu (2011c) a dynamic programming algorithm, and Sun, Tan, and Guangjian (2011a) a branch-and-bound algorithm for the Chinese postman problem with time-dependent travel times. Tan, Sun, and Hou (2013) consider the time-dependent rural postman problem, present an arc-path formulation, and propose a cutting plane algorithm. A more comprehensive treatment of this topic on node routing problems can be found in Sections 5.5.2 and 15.3.2 of Toth and Vigo (2014).

The literature on arc routing problems with zigzag options is sparse and has only been addressed by Irnich. The problem was initially defined in Irnich (2005) for undirected graphs. Irnich proposes a mixed integer model formulation and a transformation into a symmetric traveling salesman problem. Furthermore, he analyzes the effects of zigzag options on the objective function value. In Irnich (2008a), a real-world postman problem is presented that arises in mail delivery and which includes zigzag options. The problem is transformed into an asymmetric traveling salesman problem and is solved by a neighborhood search heuristic. In another paper, Irnich (2008b) considers the undirected Chinese postman problem with zigzag options (UCPPZ) and the undirected rural postman problem with zigzag options (URPPZ). He shows that the UCPPZ can be solved in polynomial time and introduces a cutting plane solution algorithm for the URPPZ.

Finally, our problem also relates to the arc routing problem with so-called mode choices. In Vidal (2015), mode choices are defined as different ways to fulfill a service and it is left to the optimization model to decide which modes are selected in a solution. Vidal (2015) considers, e.g., mode choices that relate to different service orientations. In the WRPPZTW, a mode choice relates to different service types, i.e., zigzag or normal traversal. We can generalize our mathematical model such that we can also cope with different mode choices rather than just the service types. We do so by incorporating a vertex for each mode of a mode choice in the transformed graph which will be explained later and by adding constraints that forbid the selection of multiple modes.

3. Mathematical formulations

In the following, we will present two mathematical formulations for the WRPPZTW from a node routing perspective. The following notation is used throughout the paper: let $G = (V, A)$ denote a directed graph with node set V and arc set $A \subseteq \{(i, j) | i \neq j \in V\}$. Some arcs, denoted by A_r, are required to be served exactly once by a normal traversal (not zigzagging), some segments (recall that one segment $\{i, j\}$ consists of two arcs (i, j) and (j, i)), denoted by M_r, are required to be served exactly once by zigzagging, and some segments, denoted by M, can either be served once by zigzagging or by two normal traversals. We furthermore impose time restrictions on a subset M_{TW} of M to ensure that some segments can only be zigzagged early. Zigzag operations are only allowed on these segments in $M_r \cup M$. In other words, arcs in A_r cannot be served by zigzagging. The set of required arcs and segments is given by $R = A_r \cup M_r \cup M$. The parameter D_{ij} denotes the travel time on arc (i, j) without service, S_{ij} the extra time required to service arc (i, j), Z_{ij} the time of a zigzag service of segment $\{i, j\}$ by using arc (i, j), and T_{ij} represents the latest finishing time of a zigzag option on segment $\{i, j\}$ by traversing from i to j. Recall that we consider time windows of the form $[0, T_{ij}]$. A summary of notation is presented in Table 1. For simplicity we assume that $Z_{ij} = Z_{ji}$ and $T_{ij} = T_{ji}$ in our computational study. The goal is to find a route that has the minimum completion time while it serves all arcs either by zigzagging or normal traversal and respects all time restrictions. Thus, the problem we study here considers time (without other costs for service) and one-sided time windows which only have upper bounds.

Table 1
Notation.

Parameter	Parameter description
$G = (V, A)$	directed graph with node set V and arc set $A \subseteq \{(i, j) \mid i \neq j \in V\}$
A_r	arcs that have to be served exactly once by a normal traversal
M_r	segments that have to be served exactly once by zigzagging
M	segments that can either be served once by zigzagging or by two normal traversals
$M_{TW} \subseteq M$	segments for which a finishing time is defined for the zigzag option
$R = A_r \cup M_r \cup M$	arcs and segments that require service
D_{ij}	travel time on arc (i, j) without service
S_{ij}	service time on arc (i, j)
Z_{ij}	zigzag time on segment $\{i, j\}$ by using arc (i, j)
T_{ij}	latest finishing time on segment $\{i, j\}$ for zigzagging by using arc (i, j)

3.1. Graph transformation

The two formulations are based on a graph transformation. The WRPPZTW is thereby transformed into an equivalent node routing problem that is defined on a transformed digraph $G' = (V', A')$. Throughout the paper, we assume that digraph G' is simple (i.e., loopless and without multiple arcs) and complete. A detailed description of the node set V' and arc set A' is given in the subsequent Sections 3.1.1 and 3.1.2, respectively.

3.1.1. Node set

For each member (an arc or a segment) in R, we incorporate one to four nodes in the transformed graph G'. The nodes are constructed as follows (refer to Fig. 3 for an illustration):

- Each arc $(i, j) \in A_r$ is represented by a single node i_j. We further denote the set of these nodes by $V_{A_r} := \{i_j | (i, j) \in A_r\}$.
- Each segment $\{i, j\} \in M_r$ is represented by two nodes $\tilde{i}_j$ and $\tilde{j}_i$. Node $\tilde{i}_j$ represents the zigzag service on arc (i, j) and node $\tilde{j}_i$ the zigzag service on arc (j, i). Let $\tilde{V}_{M_r} = \{\tilde{i}_j, \tilde{j}_i | \{i, j\} \in M_r\}$.
- Each segment $\{i, j\} \in M$ is represented by four nodes i_j, j_i, $\tilde{i}_j$, $\tilde{j}_i$. Nodes i_j and j_i denote the service of arc (i, j) and (j, i) by a normal traversal, respectively. Nodes $\tilde{i}_j$ and $\tilde{j}_i$ denote the service of segment $\{i, j\}$ by zigzagging on arc (i, j) and (j, i), respectively. Let $V_M = \{i_j, j_i | \{i, j\} \in M\}$ and $\tilde{V}_M = \{\tilde{i}_j, \tilde{j}_i | \{i, j\} \in M\}$.

For ease of exposition, we add an artificial depot 0_1 to G' and connect every node to this depot. The set of nodes V' is then given by $V' = V_{A_r} \cup \tilde{V}_{M_r} \cup \tilde{V}_M \cup V_M \cup \{0_1\}$.

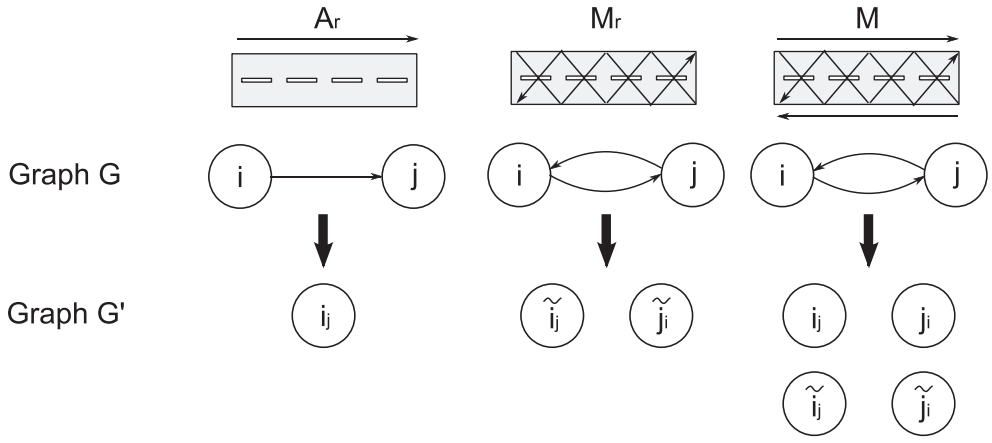


Fig. 3. Nodes of the graph transformation G' (motivated by Irnich, 2005).

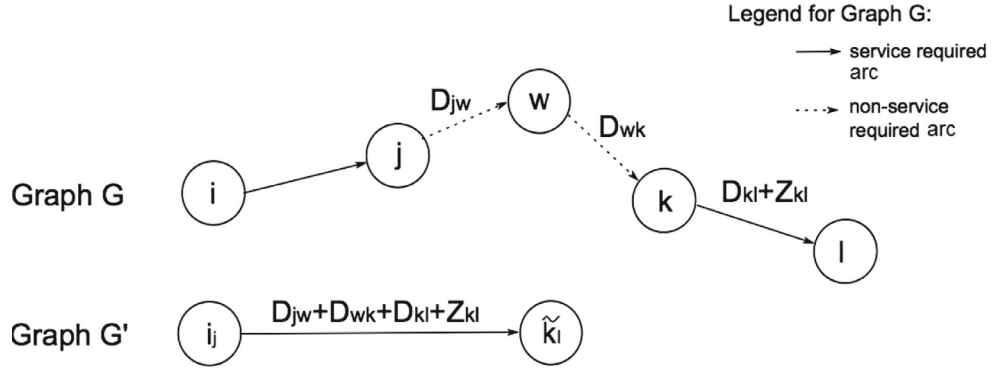


Fig. 4. Arc set of the graph transformation G' (motivated by Tagmouti et al., 2007).

3.1.2. Arc set

Each pair of distinct nodes $i_j \in V'$ and $k_l \in V'$ is connected by an arc $(i_j, k_l) \in A'$ with length $c_{i_j k_l} \in \mathbb{Q}^+$. The values of $c_{i_j k_l}$ are equal to the shortest path distances defined between the two corresponding arcs (i, j) and (k, l) in G. These shortest paths values $c_{i_j k_l}$ are calculated from the end node of the first arc (i, j) to the start node of the second arc (k, l) and include the sum of the travel time and the service or zigzag time of the second arc (refer to Fig. 4 for an illustration). The shortest path distances can be computed by the Johnson–Dijkstra algorithm (Johnson, 1977) in $\mathcal{O}(|A||V| + |V|^2 log|V|)$ steps.

3.2. Model formulation P_1

The first formulation P_1 that we present is based on the transformed graph G' and addresses the WRPPZTW as a node routing problem. Two types of variables are incorporated for this formulation: the binary decision variable $x_{i_j k_l}$ takes the value 1, if arc $(i_j, k_l) \in A'$ is traveled to first service node i_j and subsequently node k_l, and 0 otherwise. Furthermore, a time variable $t_{i_j} \geq 0$ is introduced for each node $i_j \in V' \setminus \{0_1\}$ to denote the time when the service of node i_j ends. The model formulation is then given by the following mixed integer programming problem:

$$\min \quad \sum_{(i_j,k_l)\in A'} c_{i_j k_l} x_{i_j k_l} \tag{3.1}$$

$$s.t. \quad \sum_{k_l \in V' : k_l \neq i_j} x_{k_l i_j} = 1 \qquad \forall i_j \in V_{A_r} \cup \{0_1\} \tag{3.2}$$

$$\sum_{k_l \in V' : k_l \neq i_j} x_{k_l i_j} \leq 1 \qquad \forall i_j \in \tilde{V}_{M_r} \cup V_M \cup \tilde{V}_M \tag{3.3}$$

$$\sum_{k_l \in V' : k_l \neq \tilde{i}_j} x_{k_l \tilde{i}_j} + \sum_{k_l \in V' : k_l \neq \tilde{j}_i} x_{k_l \tilde{j}_i} = 1 \qquad \forall \{i, j\} \in M_r \tag{3.4}$$

$$\sum_{k_l \in V' : k_l \neq \tilde{i}_j} x_{k_l \tilde{i}_j} + \sum_{k_l \in V' : k_l \neq \tilde{j}_i} x_{k_l \tilde{j}_i} \leq 1 \qquad \forall \{i, j\} \in M \tag{3.5}$$

$$\sum_{k_l \in V' : k_l \neq i_j} x_{k_l i_j} = \sum_{k_l \in V' : k_l \neq j_i} x_{k_l j_i} \qquad \forall \{i, j\} \in M \tag{3.6}$$

$$\sum_{k_l \in V' : k_l \neq \tilde{i}_j} x_{k_l \tilde{i}_j} + \sum_{k_l \in V' : k_l \neq \tilde{j}_i} x_{k_l \tilde{j}_i} + \sum_{k_l \in V' : k_l \neq i_j} x_{k_l i_j} = 1 \qquad \forall \{i, j\} \in M \tag{3.7}$$

$$\sum_{k_l \in V' : k_l \neq \tilde{i}_j} x_{k_l \tilde{i}_j} + \sum_{k_l \in V' : k_l \neq \tilde{j}_i} x_{k_l \tilde{j}_i} + \sum_{k_l \in V' : k_l \neq j_i} x_{k_l j_i} = 1 \qquad \forall \{i, j\} \in M \tag{3.8}$$

$$\sum_{k_l \in V' : k_l \neq i_j} x_{k_l i_j} - \sum_{k_l \in V' : k_l \neq i_j} x_{i_j k_l} = 0 \qquad \forall i_j \in V' \tag{3.9}$$

$$t_{\tilde{i}_j} \leq T_{ij}, t_{\tilde{j}_i} \leq T_{ji} \qquad \forall \{i, j\} \in M_{TW} \tag{3.10}$$

$$t_{i_j} + c_{i_j k_l} - t_{k_l} \leq (1 - x_{i_j k_l})B \qquad \forall i_j \in V' k_l \in V' \setminus \{0_1, i_j\} \tag{3.11}$$

$$x_{i_j k_l} \in \{0,1\} \qquad \forall (i_j, k_l) \in A' \tag{3.12}$$

$$t_{i_j} \geq 0 \qquad \forall i_j \in V' \setminus \{0_1\} \tag{3.13}$$

The objective function (3.1) corresponds to the minimization of the total completion time. Constraints (3.2) ensure that each node that represents a required arc $(i, j) \in A_r$ is serviced exactly once. Furthermore, constraints (3.2) ensure that the depot 0_1 is visited exactly once. Inequalities (3.3) guarantee that a node that represents a zigzag or normal service on segment $\{i, j\} \in M_r \cup M$ cannot be serviced more than once. Equalities (3.4) ensure that one zigzag service is chosen for segment $\{i, j\} \in M_r$. Equalities (3.5) verify that the zigzag service is only allowed once (in either direction) on a segment $\{i, j\} \in M$. The constraints (3.6) ensure that if a segment $\{i, j\} \in M$ is serviced by a normal traversal in one direction, the reverse direction also has to be serviced by a normal traversal. Equalities (3.7) and (3.8) guarantee that each required segment $\{i, j\} \in M$ is either serviced by zigzagging or by two normal services. Flow conservation is enforced by constraints (3.9). Constraints (3.10) verify that a zigzag service has to be finished before time T_{ij} (T_{ji}) on node $\tilde{i}_j$ ($\tilde{j}_i$) for segment $\{i, j\} \in M_{TW}$. Constraints (3.11) connect variables x and t and enforce a time increase of t_{k_l} compared to t_{i_j}, if $x_{i_j k_l} = 1$. The parameter B hereby denotes a big number. In Section 4.2.2, we describe the way to set its value. Finally, constraints (3.12) and (3.13) define the domains of the variables.

A drawback of this model formulation is the following: even though time restrictions are imposed on a small number of arcs and generally only in the early hours of the planning horizon, we have to incorporate time variables for all of the nodes in V' in order to fulfill the time constraints. To partially overcome this disadvantage, we propose a second mathematical model P_2.

3.3. Model formulation P_2

The second model P_2 also addresses the WRPPZTW from a node routing perspective. The formulation is again defined on the transformed graph G' and is based on a formulation that was initially proposed by Ascheuer, Fischetti, and Grötschel (2000) for the asymmetric traveling salesman problem with time windows (TSPTW). The authors introduce a model formulation for the TSPTW that is solely based on the arc variables $x_{i_j k_l}$. The time restrictions are fulfilled by imposing a set of inequalities, the so-called infeasible path constraints.

We define an infeasible path as a path $W = ((0_1, f_g), (f_g, k_l), \ldots, (s_t, p_q))$ for which the last node p_q of W does not fulfill the associated finishing time T_{pq}. Let C denote the set of connected components formed by the x variables. The model P_2 is then given as follows:

$$\min \sum_{(i_j, k_l) \in A'} c_{i_j k_l} x_{i_j k_l} \tag{3.14}$$

$$\text{s.t.} \quad \text{Eqs. (3.2)–(3.9)} \tag{3.15}$$

$$\sum_{i_j \in S,\ k_l \in V' \setminus S} x_{i_j k_l} \geq 1 \qquad \forall S \in C \ \&\ |C| > 1 \tag{3.16}$$

$$\sum_{(i_j, k_l) \in W} x_{i_j k_l} \leq |W| - 1 \qquad \forall \text{ infeasible paths } W \tag{3.17}$$

$$x_{i_j k_l} \in \{0,1\} \qquad \forall (i_j, k_l) \in A' \tag{3.18}$$

The objective function (3.14) and the constraints (3.15) are defined as in model P_1. Route connectivity is imposed by the classical subtour elimination constraints (3.16). Constraints (3.17) enforce that the time restrictions are fulfilled and constraints (3.18) define the domains of the decision variables.

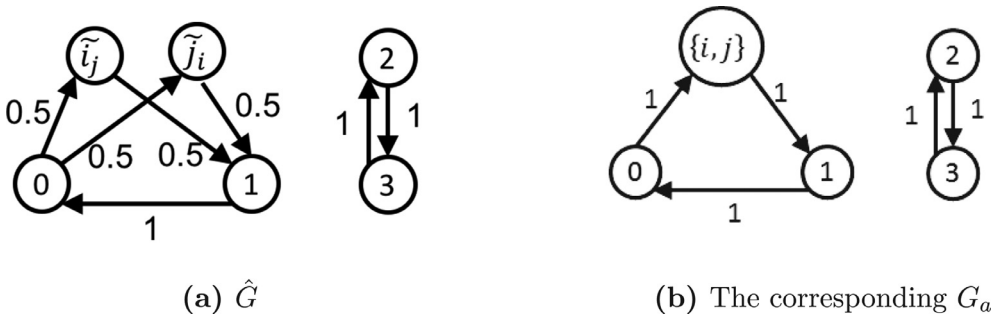
The disadvantages of the first model P_1 are overcome by model P_2. However, model P_2 contains exponentially many constraints. Since it is impossible to generate and store the entire constraint matrix for large instances of the WRPPZTW, we solve model P_2 by a branch-and-cut approach. Thus, instead of dealing with all subtour elimination (3.16) and infeasible path constraints (3.17) simultaneously, we solve a restricted version of P_2 by considering only subsets of the constraints (3.16) and (3.17). The resulting solution is then used to set up separation problems which either prove the global optimality of the solution to the full problem P_2 or identify violated constraints that are added to the restricted model formulation. In the latter case, the restricted model is then resolved.

The separation procedures for the subtour elimination and the infeasible path constraints are described in Sections 3.3.1 and 3.3.2, respectively. Note that various different cutting planes and separation procedures have been presented in the literature for the asymmetric traveling salesman problem (e.g., closed alternating trail (CAT) and COMB inequalities). We, however, restrict our branch-and-cut method to the subtour elimination and infeasible path constraints and refer the reader, e.g., to Applegate, Bixby, Chvátal, and Cook (2006), Cook (2012), and Gutin and Punnen (2002) for further details.

3.3.1. Separation procedure for the subtour elimination constraints

To separate the subtour elimination constraints, we construct a supporting digraph $\hat{G} = (\hat{V}, \hat{A})$ where $\hat{V} = V'$ and $\hat{A} = \{(i_j, k_l) \in A' | x_{i_j k_l} > 0\}$. If we have an integer solution, we first detect the connected components of $\hat{G}$. Then, if $\hat{G}$ is disconnected, some subtour elimination constraints are violated. In that case, all connected components are detected, the corresponding subtour elimination constraints are formulated, they are added to the restricted version of P_2, and P_2 is resolved. If $\hat{G}$ is connected, the solution fulfills the subtour elimination constraints.

However, if we have a fractional solution, the standard min-cut separation procedure for the subtour elimination constraints cannot be applied. There are two reasons for this. First, for the min-cut procedure, we need the condition that every node is entered exactly once and is exited exactly once. However, in our case this does not hold. In the transformed graph G', some nodes do not have to be visited at all. For instance, a segment $\{i, j\}$ in M_r is represented by two nodes $\tilde{i}_j$ and $\tilde{j}_i$, but it is possible that only one of them is visited in this fractional solution. Furthermore, we use multiple nodes to represent one segment in $M_r \cup M$. Therefore, when all these nodes are visited in the solution, some of them can be on one side of a min-cut, but the rest can be on the other side. Then, it is invalid to impose a subtour elimination constraint for this situation. We propose the following method to work around these two issues. Based on the supporting graph $\hat{G}$, we construct one aggregated graph G_a. First, initialize G_a as a copy of $\hat{G}$. Then, for each segment $\{i, j\}$ in $M_r \cup M$, we aggregate all its representative nodes into one node $\{i, j\}$. For this node, its incoming arcs are the aggregation over incoming arcs of representative nodes for this segment. The same is true for its outgoing arcs. Specifically, for a segment $\{i, j\}$ in M_r, we have two nodes for it in $\hat{G}$, $\tilde{i}_j$ and $\tilde{j}_i$. We only have one node $\{i, j\}$ in G_a after aggregation. For an outgoing arc to node k, $x_{\{i,j\}k} = x_{\tilde{i}_j k} + x_{\tilde{j}_i k}$. For an incoming arc from node k, $x_{k\{i,j\}} = x_{k\tilde{i}_j} + x_{k\tilde{j}_i}$. We can also handle a segment $\{i, j\}$ in M in this way, although it has four representative nodes (i_j, j_i, $\tilde{i}_j$, and $\tilde{j}_i$) in $\hat{G}$. Basically, we treat those two nodes for a segment in M_r



(a) $\hat{G}$ (b) The corresponding G_a

Fig. 5. Transformation of fractional solution subtour elimination illustration.

or four nodes for a segment in M as one larger node. If a violated min-cut is found and node $\{i, j\}$ is on the side containing the depot (called side A), we put both nodes $\tilde{i}_j$ and $\tilde{j}_i$ on side A. Nodes for all segments in $M_r \cup M$ are handled in this way once we are given a min-cut based on G_a. Then, we can formulate the corresponding subtour elimination constraints and add them to the restricted version of P_2, and P_2 is resolved. In this way, we ensure that each node in G_a is entered and exited exactly once and those representative nodes for one segment are put on the same side of a min-cut. A side benefit is that the size of the supporting graph is reduced because we use G_a instead of $\hat{G}$. So, the computational efficiency is improved. We use the example in Fig. 5 to illustrate the procedure.

In Fig. 5(a), we have one supporting graph $\hat{G}$. The numbers beside arcs are the solution values. Here, we have one segment $\{i, j\}$ in M_r; it is represented by two nodes $\tilde{i}_j$ and $\tilde{j}_i$. Then, it is aggregated into one larger node $\{i, j\}$ in the corresponding aggregated graph G_a shown in Fig. 5(b). Also, arc $(0, \{i, j\})$ has value 1 because it is the sum of arcs $(0, \tilde{i}_j)$ and $(0, \tilde{j}_i)$. It is similar for arc $(\{i, j\}, 1)$. Here, the min-cut is the one which has nodes 0, 1, and $\{i, j\}$ on one side and nodes 2 and 3 on the other side. We put nodes $\tilde{i}_j$ and $\tilde{j}_i$ on the same side with nodes 0 and 1. Then, we can add constraint $x_{02} + x_{03} + x_{12} + x_{13} + x_{\tilde{i}_j 2} + x_{\tilde{i}_j 3} + x_{\tilde{j}_i 2} + x_{\tilde{j}_i 3} \geq 1$ to the restricted version of P_2.

3.3.2. Separation procedure for the infeasible path constraints

To separate the infeasible path constraints, we need to verify if a solution of the restricted model fulfills the time constraints. For an integer solution, let us assume a path is given by $W = ((0_1, f_g), (f_g, k_l), \ldots, (s_t, p_q))$ and $(p, q) \in M_{TW}$. We introduce a time variable $t_{i_j} \geq 0$ for each node i_j in W to denote the time when its service ends. The values of the time variables are determined sequentially as follows: we start off by setting t_{0_1} equal 0. Afterwards, we compute the value of t_{i_j} by setting it equal to $t_{0_1} + c_{0_1 i_j}$. The values of the remaining variables are determined accordingly. We verify whether path W is feasible by checking if $t_{i_j} \leq T_{ij}$ for all $i_j \in W$ and $\{i, j\} \in M_{TW}$. If this does not hold, path W is infeasible and we set up the corresponding infeasible path constraint.

For a fractional solution, based on the idea in Ascheuer, Fischetti, and Grötschel (2001), our separation is as follows: given a fractional solution x^*, let S be the set of those nodes which represent a zigzag with time window operation and are visited in this fractional solution. For each node v in S, we can build a tree with v as the root node and every non-root node of the tree corresponds to an elementary path ending at v. The tree starts with node v. Then, it is expanded by adding one more leaf at a time by following the arcs in the support graph of the fractional solution x^*. Let u be the current leaf and node w be a neighbor in the support graph and it is the next one we will examine. A branch of this tree is expanded in a depth-first fashion until any of the following two termination criteria is satisfied:

1. u is node 0_1, the depot.
2. w has been visited before in this branch, e.g., one of the predecessors of u is w. We can discard this branch because it is not a simple path.

Then, for a path obtained by the termination criterion 1, we can check its feasibility in this way:

Let the path be $W = \{v^0, v^1, v^2, v^3, \ldots, v^{k-1}, v^k\}$. We calculate its time as $T = c_{v^0 v^1} x^*_{v^0 v^1} + c_{v^1 v^2} x^*_{v^1 v^2} + c_{v^2 v^3} x^*_{v^2 v^3} + \cdots + c_{v^{k-1} v^k} x^*_{v^{k-1} v^k}$. If $T > T_{v^k}$ (the total time is larger than the deadline of node v^k), we have one infeasible path. Furthermore, in order to find more infeasible paths, we calculate the time as $T = c_{v^0 v^1} + c_{v^1 v^2} + c_{v^2 v^3} + \cdots + c_{v^{k-1} v^k}$. In other words, we assume that in the path from v^k to the depot, all the $\mathbf{x}$ variables are equal to value 1. Therefore, some feasible paths with fractional solutions could become infeasible.

Then, we lift this infeasible path for stronger constraints. The first one is the tournament constraint:

$$\sum_{i=0}^{k-1} \sum_{j=i+1}^{k} x_{v^i v^j} \leq |W| - 1. \tag{3.19}$$

There are four more sets of valid inequalities proposed in Ascheuer et al. (2001). The main idea of these inequalities is to either take one node or two nodes from the infeasible path. Then, if one node is taken, it is fixed as the first or the last node. If two nodes are taken, they are served as the first and the last node. Then, a tournament constraint is derived with the fixed first node and/or the fixed last node. However, given the special form of time windows in our problem, the first node and the last node must be the depot and the node v, respectively. Hence, we can focus on the following constraint:

$$\sum_{j \in W^-} x_{v^0 j} + \sum_{i \in W^-} \sum_{j \in W^- \setminus i} x_{ij} + \sum_{i \in W^-} x_{i v^k} \leq |W| - 1, \tag{3.20}$$

where $W^- = \{v^1, v^2, v^3, \ldots, v^{k-1}\}$ based on W.

If the LHS of any of the above two constraints is bigger than $|W| - 1$, we can add that constraint. Otherwise, we try to find another infeasible path.

4. Computational study

All described solution methods have been implemented in Java 2 under Windows 7 and were run on an Intel Pentium Core 2 Duo, 2.2 GHz PC with 4 GB system memory. We used ILOG CPLEX 12.5 Concert Technology to solve all model formulations.

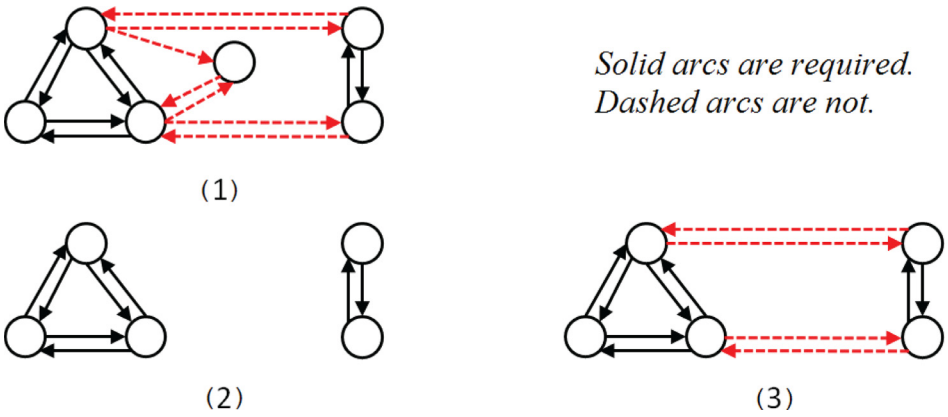


Fig. 6. Illustration of graph reduction.

4.1. Test instances

Our test instances are based on real-world data from RouteSmart Technologies, Inc (located in Columbia, Maryland). It contains a directed graph with 407 nodes and 1578 arcs. We do not have specific information related to zigzag options. The procedure used for generating that will be explained later. For each arc, we have the travel time and the service time if applicable. However, many arcs do not require service. Keeping them in the input data can cause extra computational burden. Thus, we apply a graph reduction procedure which was proposed by Christofides, Campos, Corberán, and Mota (1986). First, define required nodes as nodes that are incident to at least one required arc. From the original graph, we construct a new graph which is induced by required nodes. Then, arcs are added to make it a complete graph. For each pair of nodes, its cost is set as the shortest path length based on the original graph. Then, we delete those arcs whose corresponding shortest path contains required nodes except the start and destination nodes in the original graph. The triangular inequality is satisfied in our data. We reduce the number of nodes from 407 to 171 and the number of arcs from 1578 to 994. This procedure reduces computing time significantly.

Fig. 6 is an illustration of the reduction procedure. In subgraph 1, we have the original graph. Solid arcs are required ones and dashed arcs are not. We keep those nodes associated with at least one required arc in subgraph 2. Then, as shown in subgraph 3, we add back four dashed arcs to restore strong connectivity.

Next, we take a portion of this graph to create a test instance. A node is randomly picked as the start point and we expand from there by including adjacent nodes. The expanding stops when the number of required nodes reaches a predefined threshold. Thus, the induced graph of these selected nodes forms the basis of a test instance. Then, from this basis, we define a segment $\{i, j\}$ as a qualified segment if it needs service in both directions, on arc (i, j) and (j, i). Next, we randomly assign these qualified segments to be one of three kinds:

1. Zigzag required: these segments must be served exactly once by zigzagging and direction could be from i to j or from j to i.
2. Zigzag option: these segments can either be served once by zigzagging or by two normal traversals.
3. Zigzag with time window: these segments can either be served by two normal traversals or be served once by zigzagging if it happens within a pre-specified time window.

The probabilities are p_{M_r}, p_M, $p_{M_{TW}}$ for zigzag required, zigzag option, and zigzag with time window, respectively. Initially, $p_{M_r} = 0.25$, $p_M = 0.375$, $p_{M_{TW}} = 0.375$. Notice that there are 86 qualified segments in the original graph, i.e., 86 possible zigzag segments.

Recall that Z_{ij} denote the zigzag time of segment $\{i, j\}$ by using arc (i, j). Also, S_{ij} and D_{ij} denote the service time and the travel time for arc (i, j), respectively. Note that the service time and the travel time for arc (i, j) could be different from that of arc (j, i). For this problem, we make an assumption about the relationship between zigzag time and normal service time:

$$\min\{Z_{ij} + D_{ij}, Z_{ji} + D_{ji}\} \leq S_{ij} + D_{ij} + S_{ji} + D_{ji} \tag{4.1}$$

This assumption means there should be a benefit to zigzag for at least one direction of a zigzag possible segment. We set $Z_{ij} = Z_{ji} = \alpha * (S_{ij} + S_{ji})$ where $\alpha \geq 1$ and it is 1.5 initially. In this way, it respects the assumption (4.1) based on the given data. Furthermore, it reflects the reality that the zigzag time for that segment should not be less than the sum of service time if it is served by two normal traversals. Lastly, T_{ij} represents the latest finishing time of a zigzag option (i.e., time window width) on segment $\{i, j\}$ by using arc (i, j). Its value will be explained later in Section 4.2.2.

4.2. Computational results

4.2.1. Study of zigzag options

In this section, we explore the effect of zigzag options and that of zigzag time. Here, we ignore time windows and use the whole graph from RouteSmart Technologies, Inc. Both models P_1 and P_2 can be easily adapted to accommodate this situation. In P_1, constraint (3.10) can be removed. In P_2, the infeasible paths constraint (3.17) are no longer needed. Furthermore, we can also use the MIP formulation in Irnich (2005) on the original graph without the transformation described in Section 3.1. This might be more computationally efficient because the original graph is much smaller than the transformed one. We do not report computational time here because all instances are solved to optimality within several seconds once time window constraints are ignored.

First, we focus on the effect of zigzag options. In particular, we changed the number of zigzags segments from 0 to 86, the maximum possible value. Fig. 7 summarizes the result. It plots the total time, travel time, and service time which includes both normal service time and zigzag service time in hours versus the number of zigzag options. We can see that when the number of zigzag options increases, the objective value (i.e., total time) decreases. Furthermore, there is a tradeoff between travel time and service time. Service time is increasing and travel time is decreasing when the number of zigzag options increases.

Second, we study the effect of zigzag time. Suppose the number of zigzag options is fixed at 86 and we vary the zigzag time as a function of S_{ij} and S_{ji} by increasing α from 0 to 4. The results are shown in Fig. 8. First, we can see that when the zigzag time gets larger, there will be less zigzagging. This is not surprising. On

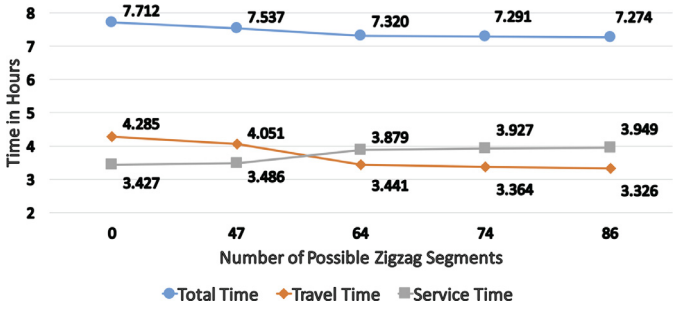


Fig. 7. The effect of zigzag options.

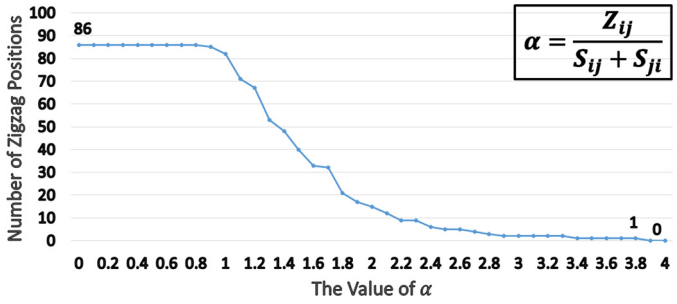


Fig. 8. The effect of zigzag time.

the other hand, even when α is as big as 3.8, at least one zigzag segment is profitable. Lastly, we calculate the ratio

$$\gamma = \frac{Z_{ij} + D_{ij}}{S_{ij} + D_{ij} + S_{ji} + D_{ji}}$$

which measures the amount of time required by zigzag relative to two-way arc traversal. When α is 3.8, the only zigzag operation that takes place has $\gamma = 1.2112$. Also, in Fig. 8, the largest γ is 1.3523; this corresponds to $\alpha = 3.4$. These γ values are bigger than 1 which means that even when the assumption (4.1) is not fulfilled, the zigzag options can still be beneficial. In other words, the zigzag time can be relatively large and yet still offer cost-saving opportunities.

4.2.2. Effects of time windows

In a last series of tests, we analyze the effects of time windows on objective function values and computation times. Here, we conduct two experiments. First, we vary the number of time windows by imposing different probabilities 0.2, 0.4, 0.6, 0.8, and 1.0 for attaching time windows to zigzag segments. In the second experiment, we gradually increase the time window width (i.e., T_{ij}) from 500 to 1000, 1500, 2000, and 2500 seconds. We base our study on five different 20 node graphs. For each time window probability (e.g., 0.2), we derive five instances for each time window width (500, 1000, 1500, 2000, 2500). (Analogously, for a fixed time window width (e.g., 500), we generate five instances for each time window probability (0.2, 0.4, 0.6, 0.8, 1.0).) In that way, we obtain $5 * 5 = 25$ instances for each graph. We point out that we also ran these two experiments on a small number of 40 node graphs. The results were essentially the same, and we do not present the details here.

First, we report the parameters of each data set in Table 2. These values are identical for all data sets, since we only vary the number and the width of the time windows. We list the number of vertices ("$|V|$") and the average number of arcs ("$|A|$") of $G = (V, A)$. Furthermore, we report the average number of arcs that require some kind of service ("$|A_r|$","$|M_r|$", "$|M|$"). The rows "$|V'|$" and "$|A'|$" list the average number of nodes and arcs in the transformed graph $G' = (V', A')$, respectively. We also present the average optimal value if zigzagging is forbidden ("Obj. Val. (no zigzag option)") and if no time windows are imposed ("Obj. Val. (no time window)") while zigzagging is allowed. Notice that the values of "Obj. Val. (no time window)" and "Obj. Val. (no zigzag option)" serve as a lower and an upper bound on the average objective function values in our study, respectively. Furthermore, we use "Obj. Val. (no zigzag option)" (the upper bound value) to set the value of B in P_1.

Table 2
Parameters of data set.

Parameter	Average value
$\|V\|$	20.00
$\|A\|$	56.00
$\|A_r\|$	24.00
$\|M_r\|$	0.00
$\|M\|$	12.80
$\|V'\|$	75.20
$\|A'\|$	2789.92
Obj. Val. (no time window)	3430.12
Obj. Val. (no zigzag option)	3705.40

Tables 3 and 4 report the results of our computational study for the different time window probabilities and widths. Three settings are included in our experiment. P_1 is provided to the solver directly. For the second model formulation, two implementations are tested. P_{2I} stands for the implementation where separation procedures for the infeasible path constraints are only called when an integral solution is obtained. In the P_{2F} implementation, separation procedures for the infeasible path constraints are called for fractional solutions as well and only in the root node of the branch-and-cut procedure. Then, if an infeasible path constraint is violated, we lift it as a tournament constraint and only add this tournament constraint when the tournament constraint is stronger than the infeasible path constraint. After that, we try to obtain a constraint (3.20) as well. For both implementations, the fractional solution subtour elimination constraints are only separated in the root node. Once we enter the branch phrase, only the integer solution subtour elimination constraints are separated. For all settings, we report the average number of zigzag arcs with time windows ("$|M_{TW}|$"), the average CPU time in seconds ("CPU"), and the average objective function value ("Obj. Val."). If an instance has not been solved to optimality in the time frame of one hour, we use the objective function value of the incumbent solution (best found solution so far). Moreover, we report the average duality gap ("Gap") and summarize under table entry "Solved/Unsolved" the number of instances that have been solved and have not been solved to optimality in the given time frame. For each model, we list the average number of zigzag traversals that have been executed in the optimal solution ("#Zigzag") and report how many of them have time window constraints ("#Zigzag TW"). Next, we list the average number of branch-and-bound nodes ("#BnB") for each model. For model P_{2I} and P_{2F}, we additionally give the average number of computed subtour elimination constraints based on integral solutions ("#SEC") and fractional solutions ("#FSEC"), respectively. For model P_{2I}, the average number of computed infeasible path constraints for integral solutions ("#IPC") is reported. For model P_{2F}, we also give the average number of computed tournament constraints based on integral solutions ("#TC") and fractional solutions ("#FTC"), respectively, and the average number of computed constraints (3.20) based on integral solutions ("#AC") and fractional solutions ("#FAC"), respectively.

For model P_2, it is hard to tell which one is better among these two implementations, P_{2I} and P_{2F}. Regarding the average computation time, P_{2I} dominates P_{2F} in eight out of ten groups. But in all ten groups, P_{2F} has significantly smaller average numbers of branch-and-bound nodes than P_{2I}. Lastly, in terms of the average

Table 3
Computational results for 5*25 instances with different time window probabilities.

	Time window probability														
	0.2			0.4			0.6			0.8			1.0		
	P_1	P_{2I}	P_{2F}	P_1	P_{2I}	P_{2F}	P_1	P_{2I}	P_{2F}	P_1	P_{2I}	P_{2F}	P_1	P_{2I}	P_{2F}
$\|M_{TW}\|$	2.40	2.40	2.40	5.00	5.00	5.00	7.60	7.60	7.60	10.80	10.80	10.80	12.80	12.80	12.80
CPU (in seconds)	42.02	46.69	164.87	1158.19	1090.05	1119.27	2072.16	1842.79	2009.03	2085.53	2058.52	2058.54	2504.83	2482.42	2601.18
Obj. Val.	3430.12	3430.12	3430.12	3469.51	3478.21	3480.40	3506.38	3546.34	3530.59	3538.07	3549.06	3558.17	3543.59	3591.76	3580.38
Gap (in %)	0.00	0.00	0.00	1.01	1.40	1.47	1.95	3.27	2.89	2.67	3.36	3.62	2.89	4.53	4.28
Solved/Unsolved	25/0	25/0	25/0	19/6	18/7	18/7	13/12	14/11	12/13	12/13	11/14	11/14	10/15	8/17	7/18
#Zigzag	9.00	9.00	9.00	7.84	8.00	7.48	6.76	5.60	5.96	5.88	5.40	5.04	5.68	3.96	4.08
#Zigzag TW	1.00	1.00	1.00	2.80	2.64	2.52	4.24	3.52	3.76	4.36	4.12	3.84	5.68	3.96	4.08
#BnB	5825	6355	6232	161,285	24,663	17,521	341,191	51,900	33,423	295,174	52,675	35,166	354,726	69,578	43,086
#SEC	–	90.72	153.08	–	1285.76	990.88	–	2089.28	1927.56	–	2409.60	1942.80	–	2739.76	2383.24
#FSEC	–	10.80	11.24	–	10.80	11.12	–	10.80	11.16	–	10.80	11.16	–	10.80	11.12
#IPC (TC)	–	104.80	171.80	–	1435.76	1481.12	–	2418.08	3186.24	–	2873.96	3193.64	–	3046.88	4119.72
#FIPC (FTC)	–	–	1.08	–	–	2.00	–	–	7.08	–	–	4.48	–	–	6.24
#AC	–	–	117.60	–	–	510.28	–	–	586.32	–	–	548.76	–	–	499.12
#FAC	–	–	0.16	–	–	1.36	–	–	3.00	–	–	3.00	–	–	3.00

Table 4
Computational results for 5*25 instances with different time window widths.

	Time window width														
	500			1000			1500			2000			2500		
	P_1	P_{2I}	P_{2F}	P_1	P_{2I}	P_{2F}	P_1	P_{2I}	P_{2F}	P_1	P_{2I}	P_{2F}	P_1	P_{2I}	P_{2F}
$\|M_{TW}\|$	7.72	7.72	7.72	7.72	7.72	7.72	7.72	7.72	7.72	7.72	7.72	7.72	7.72	7.72	7.72
CPU (in seconds)	2736.44	2766.81	2886.05	2134.53	2225.49	2071.02	1602.04	1361.99	1563.86	1045.14	908.56	908.35	344.58	257.62	523.62
Obj. Val.	3584.01	3615.82	3620.13	3532.73	3577.42	3569.45	3484.60	3494.46	3494.50	3453.44	3474.91	3452.76	3432.87	3432.87	3442.81
Gap (in %)	3.66	5.30	5.43	2.75	4.09	3.92	1.44	1.79	1.83	0.60	1.26	0.69	0.07	0.10	0.40
Solved/Unsolved	7/18	6/19	6/19	11/14	10/15	11/14	16/9	17/8	15/10	21/4	19/6	19/6	24/1	24/1	22/3
#Zigzag	4.52	3.16	2.96	6.08	4.88	4.52	7.36	7.32	6.88	8.32	7.72	8.36	8.88	8.88	8.84
#Zigzag TW	1.40	0.60	0.48	2.80	1.72	1.48	3.88	3.80	3.44	4.84	3.96	4.68	5.16	5.16	5.12
#BnB	419,901	70,186	50,066	316,926	60,333	38,167	240,472	36,147	23,581	144,704	27,185	13,767	36,198	11,322	9847
#SEC	–	3176.92	2624.64	–	2563.48	2031.68	–	1557.04	1488.76	–	1013.88	851.92	–	303.80	400.56
#FSEC	–	10.80	12.52	–	10.80	10.72	–	10.80	10.96	–	10.80	10.80	–	10.80	10.80
#IPC (ITC)	–	3215.00	3341.52	–	2753.64	3084.64	–	2244.28	3011.80	–	1307.00	1882.64	–	359.56	831.92
#FIPC (FTC)	–	–	14.16	–	–	5.52	–	–	1.20	–	–	0.00	–	–	0.00
#AC	–	–	1902.68	–	–	328.08	–	–	26.24	–	–	5.08	–	–	0.00
#FAC	–	–	10.52	–	–	0.00	–	–	0.00	–	–	0.00	–	–	0.00

duality gap, P_{2I} wins in five out of ten groups, P_{2F} wins in four out of ten groups, and there is a draw for one group. Overall, we favor the P_{2I} implementation because it has better results in terms of the number of solved instances. From P_{2F}, we can observe that those valid inequalities studied in node routing problems with time windows cannot provide significant improvement here. A further study along this line for deeper cuts could be interesting.

However, it is clearly shown from the test results in Tables 3 and 4 that the model P_1 performs better than model P_2 in both P_{2I} and P_{2F} implementations on almost all instances in terms of the objective function values, the number of solved instances, and the duality gap.

Unfortunately, neither model is capable of solving all instances in the given time frame. The data in Table 3 reveals that the computation time of both models is positively correlated with the time window probability. As the probability increases, the CPU time increases and the number of instances that are solved to optimality decreases. Furthermore, the average objective function value of probability 0.2 is equal to the lower bound (as if no time windows are imposed). As the probability is increased, the objective function values increase. The opposite can be observed in Table 4. As the time window width is increased, the computation times and objective function values decrease and more instances are solved to optimality. For small time window widths, model P_2 has to generate a large amount of infeasible path constraints, which results in substantial computation times. Notice that the average number of fractional solution subtour elimination constraints is roughly the same for all columns of Tables 3 and 4. This is due to the fact that these data sets are based on the same instances and differ solely in the time window probability and width. Since we call the fractional solution subtour elimination constraints only in the root node of our branch-and-cut procedure, where no infeasible path constraint has been generated so far, the results are the same. We observe that the smaller the time window widths, the harder an instance. This observation is in contrast to node routing problems with time windows in Ascheuer et al. (2001), Dash, Günlük, Lodi, and Tramontani (2012) and so on. In those papers, an instance with smaller time window widths is easier to solve. In these problems, the time windows are “hard” such that the nodes with time windows must be visited within the time windows. Hence, a precedence order can be derived based on the time windows and is critical for some preprocessing techniques which improve computing efficiency. In our problem, the time windows are “soft” in the sense that even if the postman could not zigzag a street segment within the time window, he still is able to serve this street segment by two normal traversals. Thus, the smaller time windows only reduce the feasible region, but do not provide a precedence order of the street segments.

5. Conclusions

We study a novel variant of the windy rural postman problem. In additional to the WRPP, zigzag options with and without time windows are incorporated into the problem. We propose two MIP formulations and conduct computational experiments using real-world data from RouteSmart Technologies, Inc. First, we ignore time windows and focus on the effect of zigzag options by varying the number and cost of zigzag options. When we have more zigzag options, the total cost is smaller, as expected. But, surprisingly, even when the cost of zigzag service is 3.8 times as large as the sum of normal service in both directions, it still might make sense to zigzag in our experiment. In other words, the zigzag time can be relatively large and yet still offer cost-saving opportunities. Second, we study the effect of time windows by changing the number and width of time windows. The model P_1 is better than model P_2 on almost all instances in terms of the objective function values, the number of solved instances, and the duality gap. However, not all 20 node instances are solved to optimality. Time windows dramatically increase the difficulty level of the WRPPZTW. A general observation is that the larger the number of time windows and/or the smaller the width, the harder an instance. Overall, zigzag options are valuable in the real-world. They should receive more research attention. In future work, to improve the computational performance, deeper cutting planes might be considered for P_2; currently the large number of infeasible path constraints, tournament constraints, and constraint (3.20) is not very helpful. Also, a specialized branch-and-price scheme might be another fruitful direction to explore.

References

Afsar, H. M. (2010). A branch-and-price algorithm for capacitated arc routing problem with flexible time windows. *Electronic Notes in Discrete Mathematics, 36*, 319–326.

Aminu, U. F., & Eglese, R. W. (2006). A constraint programming approach to the Chinese postman problem with time windows. *Computers & Operations Research, 33*, 3424–3431.

Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2006). *The traveling salesman problem: A computational study*. Princeton: Princeton University Press.

Ascheuer, N., Fischetti, M., & Grötschel, M. (2000). A polyhedral study of the asymmetric travelling salesman problem with time windows. *Networks, 36*, 69–79.

Ascheuer, N., Fischetti, M., & Grötschel, M. (2001). Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming, 90*(3), 475–506.

Bodin, L. D., & Kursh, S. J. (1978). A computer-assisted system for the routing and scheduling of street sweepers. *Operations Research, 26*, 525–537.

Christofides, N., Campos, V., Corberán, A., & Mota, E. (1986). An algorithm for the rural postman problem on a directed graph. In *Netflow at pisa* (pp. 155–166). Springer.

Cook, W. J. (2012). *In pursuit of the traveling salesman: Mathematics at the limits of computation*. Princeton: Princeton University Press.

Corberán, A., & Laporte, G. (2014). *Arc routing: Problems, methods, and applications*. MOS-SIAM Series on Optimization.

Dash, S., Günlük, O., Lodi, A., & Tramontani, A. (2012). A time bucket formulation for the traveling salesman problem with time windows. *INFORMS Journal on Computing, 24*(1), 132–147.

Eglese, R. W. (1994). Routeing winter gritting vehicles. *Discrete Applied Mathematics, 48*, 231–244.

Golden, B. (2013). Some interesting vehicle routing research topics: Suggestions from an oldtimer. University of Maryland: Presentation VeRoLog Conference, Southampton, U.K.

Gueguen, C. (1999). *Exact solution methods for vehicle routing problems*, Ph.D. thesis. Central School of Paris.

Gutin, G., & Punnen, A. P. (2002). *The traveling salesman problem and its variations*. Dordrecht: Kluwer Academic Publishers.

Irnich, S. (2005). A note on postman problems with zigzag service. *INFOR, 43*, 33–39.

Irnich, S. (2008a). Solution of real-world postman problems. *European Journal of Operational Research, 190*, 52–67.

Irnich, S. (2008b). Undirected postman problems with zigzagging option: A cutting–plane approach. *Computers & Operations Research, 35*, 3998–4009.

Johnson, D. B. (1977). Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM, 24*(1), 1–13.

Johnson, E. L., & Wøhlk, S. (2009). Solving the capacitated arc routing problem with time windows using column generation. *CORAL Working Paper L-2008-09*. University of Aarhus.

Letchford, A. N., & Eglese, R. W. (1998). The rural postman problem with deadline classes. *European Journal of Operational Research, 105*, 390–400.

Mullaseril, P. A. (1996). *Capacitated rural postman problem with time windows and split delivery*, Ph.D. thesis. University of Arizona.

Reghioui, M., Prins, C., & Labadi, N. (2007). GRASP with path relinking for the capacitated arc routing problem with time windows. In M. Giacobini (Ed.), *Applications of evolutionary computing*. In *Lecture Notes in Computer Science* (pp. 722–731). Berlin: Springer.

Sun, J., Tan, G., & Guangjian, H. (2011a). Branch-and-bound algorithm for the time dependent Chinese postman problem. In *Proceedings of the international conference on mechatronic science, electric engineering and computer* (pp. 949–954).

Sun, J., Tan, G., Guangjian, H., & Wang, B. (2011b). Column generation approach for the time dependent Chinese postman problem. In *Proceedings of the international joint conference on computational sciences and optimization* (pp. 450–454).

Sun, J., Tan, G., & Qu, H. (2011c). Dynamic programming algorithm for the time dependent Chinese postman problem. *Journal of Information & Computational Science, 5*, 833–841.

Tagmouti, M., Gendreau, M., & Potvin, J.-Y. (2007). Arc routing problems with time-dependent service costs. *European Journal of Operational Research, 181*, 30–39.

Tagmouti, M., Gendreau, M., & Potvin, J.-Y. (2010). A variable neighborhood descent heuristic for arc routing problems with time-dependent service costs. *Computers & Industrial Engineering, 59*, 954–963.

Tagmouti, M., Gendreau, M., & Potvin, J.-Y. (2011). A dynamic capacitated arc routing problem with time-dependent service costs. *Transportation Research Part C: Emerging Technologies, 19*, 20–28.

Tan, G., Sun, J., & Hou, J.-Y. (2013). The time-dependent rural postman problem: polyhedral results. *Optimization Methods & Software, 28*, 855–870.

Toth, P., & Vigo, D. (2002). *The vehicle routing problem*. Philadelphia: SIAM.

Toth, P., & Vigo, D. (2014). *Vehicle routing: Problems, methods, and applications*: Vol. 18. SIAM.

Vansteenwegen, P., Souffriau, W., & Sörensen, K. (2010). Solving the mobile mapping van problem: A hybrid metaheuristic for capacitated arc routing with soft time windows. *Computers & Operations Research, 37*, 1870–1876.

Vidal, T. (2015). Arc routing, vehicle routing, and turn penalties: Multiple problems – One combined neighborhood. *Technical Report*. Rio de Janeiro: Pontifcia Universidade Catlica do.

Wøhlk, S. (2005). *Contributions to arc routing,* Ph.D. thesis. University of Southern Denmark.