Soft Computing
https://doi.org/10.1007/s00500-023-08032-z

OPTIMIZATION

Solving the hierarchical windy postman problem with variable service costs using a math-heuristic algorithm

Muhammed Emre Keskin[1] · Mustafa Yılmaz[1] · Chefi Triki[2,3]

Accepted: 10 March 2023

Abstract
The Hierarchical Windy Postman Problem (HWPP) is an arc routing problem in which an order relation is imposed on the arcs/edges of the graph, and one has to pass through each edge at least once while adhering to the hierarchical priority relations. The tour starts from and ends at a specific node and the aim is to minimize the length of the tour. We consider a variant of the HWPP in which (i) the precedence order of the edge hierarchies is linear and edges within each hierarchy are connected and (ii) the cost of serving each edge decreases with the number of times it is traversed, and we refer to it as HWPP with variable service costs. An integer non-heuristic linear mathematical formulation is proposed, and a solution approach is designed. Our solution heuristic adapts the layer algorithm of Dror et al. (Networks 17:283–294, 1987) but employs an integer mathematical formulation as a sub-procedure instead of the blossom algorithm to find the least cost path between the nodes of the graph. This choice is based on the fact that the blossom algorithm requires a symmetric cost structure while we deal here with the general case of asymmetric cost structure, which makes our problem a windy variant of the postman problem. It should be noted that our problem is not asymmetric in the sense that there are no opposite arcs with different costs but there are edges which have different costs depending on the traversal direction. In order to compare the performance of our heuristic algorithm with respect to the performance of the mathematical model that is solved by the commercial solver Gurobi, 84 test instances are generated having varying sizes and densities and with different number of hierarchies. These test instances are solved by both methods and the generated results show that the proposed heuristic method is much faster and generates better quality solutions.

Keywords Routing · Hierarchical windy postman problem · Variable service cost · Windy layer algorithm

1 Introduction and literature review

Arc Routing Problems (ARPs) are a broad class of problems in the graph theory that arises in delivering many kinds of services. The recent surveys of Lahyani et al. (2015), Mourão and Pinto (2017) and Corberán et al. (2021) enumerate several applications of ARP including street cleaning, salt spreading, snow cleaning, road maintenance and drone inspection. If capacity limitations are imposed on the arcs, the problem is called as capacitated ARP (see Golden and Wong, (1981) for the first introduction of capacitated ARP or see Mandal et al. (2015) for a more generalized version of capacitated ARP). If services of the arcs have to follow certain periodic patterns, the problem is called as periodic ARP (see for instance Monroy et al. (2013) and Triki (2017)). Mathematically, ARP can be defined on a connected graph $G = (N, E)$ having n nodes and m arcs/edges and where N is the node set and E represents the arc/edge set. An edge set can be said to exist if traversal is possible in both directions, otherwise, if transition is only possible in the intended direction, then there is an arc set at hand. The graph can be either directed or undirected and to every arc/edge is associated a cost that can represent, for example, its length or traversal time. The focus of ARPs is on servicing the

✉ Chefi Triki
chefi.triki@unisalento.it

[1] Industrial Engineering Department, Atatürk University, Erzurum, Turkey

[2] Kent Business School, University of Kent, Canterbury, UK

[3] Department of Engineering for Innovation, University of Salento, Lecce, Italy

Published online: 01 April 2023

arcs/edges of the graph, rather than serving its nodes. A particular case of ARP, known as the Chinese Postman Problem (CPP), consists in identifying a least cost closed tour that starts from a depot, traverses each arc/edge of the graph at least once before coming back to the same depot. The CPP is usually defined over an undirected graph and all its edges need to be serviced. Some of the above-mentioned applications may be characterized by the fact that the edges do not have the same importance of being serviced but are rather categorized according to their class of priority (called hierarchies). We deal here with the Hierarchical Windy Postman Problem (HWPP) that has the same features of the HCPP (Hierarchical Chinese Postman Problem) with the only exception that the service costs do not have to be symmetric (see Keskin and Triki 2022).

The HCPP is defined in Afanasev et al. (2021) as the problem of finding the shortest traversal of all edges of a graph, respecting the precedence constraints given by a partial order on classes of edges. The authors also identify three special variants of the problem: (i) the case in which the precedence order of the edge hierarchies is linear, denoted as HCPP(l) (ii) the case involving connected edges within each hierarchy, denoted as HCPP(c) and (iii) when both (i) and (ii) restrictions hold, and was denoted as HCPP(c,l). In the sequel we will refer to third variant of the problem, i.e., HCPP(c,l) where a linear precedence order is imposed and each edge hierarchy induces a connected subgraph, unless differently specified. Likewise, we will often refer to the HCPP terminology in the problem definition and the literature review sections since most of the features that ground on HCPP apply to the HWPP as well.

It is to be noted that especially the second restriction related to the connectivity of the edges within each hierarchy does not to seem too restrictive from the practical viewpoint. Indeed, in the urban snowplow problem, for example, interconnected main roads have primary priority to be plowed which will form the first hierarchy. Similarly, second hierarchy consists of second priority roads that are usually connected and so on. Thus, our assumption that each hierarchy is connected is fully compatible with real-life applications.

Formally, if we denote by $|H|$ the number of pre-defined hierarchies and by $E^1, E^2, \ldots, E^{|H|}$ the subsets of edges belonging to each hierarchy, then all edges belonging to E^g must be serviced before any one belonging to E^h (with $E^g \bigcap E^h = \emptyset$) if g has a higher hierarchy than h, i.e., if $h, g \in \{1, \ldots, |H|\}$ and $g < h$. However, there are contradicting views among the scholars on the feasibility of traversing any edge (with or without servicing it) having a lower hierarchy before completing the service of any other edge belonging to a higher hierarchy category. This fact will define, indeed, three different variants of the HCPP:

- The first allows any edge from class E^h to be traversed (without service) even before completing the service of the edges belonging to a higher priority class E^g, i.e., when $g < h$. This variant of the problem, that can be applied for example to the case of street cleaning and inspection, meter reading, waste collection, and restoration of the roads after disasters, has been proposed by Alfa and Liu (1988), Eiselt et al. (1995), Colombi et al. (2017a), Colombi et al. (2017b), Shao et al. (2020), Oruc and Kara (2018) and Akbari et al. (2021) and is, sometimes, called HCPP with weak precedence relation.
- The second variant of the HCPP adopts a more restrictive interpretation of the precedence relation (strong relation) and does not allow any edge from class E^h to (not even) be traversed before completing the service of all edges of class E^g having a higher priority. This interpretation of the hierarchy has been adopted by Dror et al. (1987), Ghiani and Improta (2000) and more recently by Çodur and Yılmaz (2020) and can be suitable, for example, for the snow removal application since it would be difficult traversing a road that still needs to be cleaned.
- Yet, a different interpretation of the precedence relation has been introduced by Quirion-Blais et al. (2017) and that arose in a case-study related to the multi-vehicle road snow plowing and de-icer spreading. It consists in allowing the vehicles to start plowing any category of streets but those having higher priority should be completed before all the others.

In this paper, we will adopt the second and more restrictive type of precedence relation, but our mathematical model as well as our heuristic method can be easily adapted to suit the weak hierarchy requirement. We will consider that the cost of serving each edge will decrease with the number of times it is traversed (see Corberán et al. (2014) and Keskin and Yılmaz (2019)). This variant of the problem, that we call the Variable HWPP (V-HWPP), belongs to a wider family of variable service cost routing problems. In the sequel, we will first review the most relevant works that dealt with the different variants of the HCPP and the approaches adopted for their solution, and then we will analyze the different ways of assuming variable service cost and identify the specific version we selected for our study.

1.1 Related works on the HCPP

The HCPP has been defined over three decades ago in the seminal paper of Dror et al. (1987). The authors dealt with a CPP in which the edges of the underlying network are roads characterized by a hierarchical precedence relation.

Even though their HCPP defined on an undirected graph is NP-hard, they succeeded to solve for the special class HCPP(c,l) in $O(kn^5)$, where n is the cardinality of the nodes set and k is the number of hierarchies in the network. Later on, several studies have focused on reducing the complexity of Dror et al. (1987)'s algorithm by trying to improve its features.

Ghiani and Improta (2000) developed an exact approach for the HCPP(c,l) having complexity $O(k^3n^3)$ by constructing an alternative auxiliary graph. In this same direction, Korteweg and Volgenant (2006) reduced the complexity of Dror et al. (1987)'s algorithm to $O(kn^4)$ but rather than focusing on building a different auxiliary graph, they employed post-optimality techniques to improve the matching in the resulting auxiliary graph. Given the complexities of both the above exact approaches, the selection of the most appropriate algorithm depends on the characteristics of the instance to be solved, i.e., the value assumed by the number of nodes n compared to that of number of hierarchies k (Korteweg 2002).

Similar to Ghiani and Improta (2000), Cabral et al. (2004) developed an exact solution approach but for the HCPP(l) that was based on the idea of transforming the HCPP into a Rural Postman Problem (RPP) and solving it by a branch-and-cut technique (see also Ghiani et al. 2005, 2008).

The attention of the researchers in solving the HCPP has been directed even toward developing heuristic approaches. The first heuristic method, proposed by Alfa and Liu (1988), was defined for a directed network with weak precedence relation and without any connectivity restrictions on the arcs belonging to the same hierarchy. Their method consists in connecting and balancing the nodes that are endpoints of arcs belonging to the same hierarchy and then producing a single CPP tour based on the hierarchy-wise tours. Later on, Alfa and Liu's method was outperformed by another heuristic suggested by Damodaran et al. (2008) which was based on a more efficient way of combining the unconnected subnetworks. Damodaran et al. (2008) focused on proposing good quality lower bounds for the HCPP with the aim of defining an efficient heuristic method. Colombi et al. (2017a) employed a math-heuristic to solve the HCPP starting from the exact solution of a Mixed RPP defined on an auxiliary graph for each hierarchy, whereas Colombi et al. (2017b) performed a polyhedral analysis of a hierarchical mixed rural postman problem and developed a branch-and-cut algorithm for its exact solution. For the sake of completeness, it is worth noting that meta-heuristic approaches, namely a genetic algorithm and a hybrid simulated annealing, have also been employed by Çodur and Yılmaz (2020).

The last family of approaches defined to solve the HCPP is based on solving the minimum spanning tree problem. Sayata and Desai (2015) implemented an efficient heuristic based on the use of Kruskal's algorithm to reduce the number of edges in the underlying auxiliary graph. More recently, Colombi et al. (2017a) tackled the HCPP with mixed graph and employed a minimum spanning tree problem to identify an initial feasible solution and then embedded it within a tabu search procedure in order to improve and diversify the generated solutions.

1.2 Variable service cost in HWPPs

As mentioned above, one of the novel aspects of our work is related to considering a particular variant of the HWPP in which a variable service cost scheme is implemented, i.e., the V-HWPP. Most of the scientific literature in the context of routing optimization has focused on variable service cost expressed in terms of time varying models. Interested readers may refer to the excellent review of Gendreau et al. (2015), in which the authors highlighted the scarcity of contributions related to the time varying ARPs, compared to their node routing counterpart. Few works appeared after Gendreau et al.'s review to deal with the time-dependent ARP in general (such as Sun et al. (2015) and Vidal et al. (2021)) and one is specifically dealing with the time-dependent HCPP (Çodur and Yılmaz 2020). However, our contribution here is not related to the variable service cost from the traversal time viewpoint, but rather to the service cost that decreases with the number of server passages. To the best of our knowledge there are only two papers that have adopted this particular variable cost scheme in the context of the windy CPP. The first is due to Dussault et al. (2013) who solved a snow plowing problem in which the cost of traversing a street varies based on its order within the plowing tour. Specifically, the authors considered that the cost of serving the roads decreases after the first pass by the plowing vehicle. The second paper, by Keskin and Yılmaz (2019), generalizes variable service cost scheme by assuming that each server pass will contribute to reducing further the roads' service cost. This assumption seems to be very realistic and intuitive in a wide range of applications in which the multi-passing through roads and/or the learning process by the server contribute to decreasing the deadhead cost. For instance, the postman is expected to learn the roads better as the number of passes increases in the post-delivery application. Similarly, if each pass on the edges represents the shoveling of the accumulated snow on a specific edge, then the cost of the shoveling should not increase by the number of the passes on that edge as the snow becomes less and less at each pass of the snowplow vehicle. Our study will, thus, adopt the variable cost scheme of Keskin and Yılmaz (2019) and will implement it for the first time in the framework of the HWPP. On the other hand, it is a common practice to differentiate the service cost of the arcs from deadheading cost of the arcs as it is done for instance in Perrier et al. (2008). Similarly, Aráoz et al. (2013) associate prizes to arcs rather than costs and also differentiate between service and deadheading prizes, namely they take deadheading prize as zero.

1.3 Problem definition and our contributions

This study will specifically focus on solving the V-HWPP, a particular variant of the HWPP in which is defined over an asymmetric network whose service cost decreases with the number of edge passes. Moreover, we restrict our attention here to the variant of the problem characterized by a linear precedence order of hierarchies and by a connected subgraph within each hierarchy.

From the above analyses it results evident that the above V-HWPP has never been addressed in the scientific literature. The objective of this study is to fill in this gap and to provide the practitioners and academic researchers with a new methodology that can contribute to solving the several real-life applications that arise in this specific ARP context.

The paper is organized as follows. Section 2 will formally introduce the problem and will propose an original formulation for the V-HWPP. Then an ad hoc math-heuristic approach based on the concept of layer algorithm and computational experiments are discussed in Sects. 3 and 4, respectively. Finally, some concluding remarks and future research suggestions are provided in Sect. 5.

2 Optimization model

In this section, we first describe the sets, parameters, and variables that are used in the formulation. Later, the mathematical formulation of the V-HWPP is provided.

2.1 Sets, parameters and variables

We assume that the set of nodes and the set of edges connecting the nodes are, respectively, denoted by N and E. Indices i and j are used to represent nodes of N and an edge connecting nodes i and j is denoted by edge (i,j). H represents the set of hierarchies and index h is used to specify a hierarchy. The set of edges that belongs to hierarchy h is denoted as E^h. T stands for the set of steps while indices t and m are used to refer to specific steps while $|T|$ represents the maximum number of steps. By the way, a step is used as a measurement tool to track the transition of the vehicle from one node to another. In other words, at each step, the vehicle passes from one node to another. For instance, if a route of a vehicle consists of nodes 1, 5, 3, 2, 5, 1, then we say that the vehicle is at the starting node 1 at the beginning of step 1 and it goes from node 1 to node 5 at step 1, it is at node 5 at the beginning of step 2 and it goes from node 5 to node 3 at step 2, then it is at node 3 at the beginning of step 3 and it goes from node 3 to node 2 at step 3, etc. Note that, the duration of the steps may change from step to another and from a vehicle to another as the distances between the node-pairs do not have to be identical. Finally, K is the set of number of passes and the index k is used to point out the members of K.

The service cost parameters used in the formulation depend on the above-described sets and indices. c_{ijk} denotes the k th service cost of edge (i,j) if the service is through the direction from i to j. Note that we assume an undirected graph implying that edges can be traversed through either direction. However, if an edge is served (independently from the service direction), the cost of the service reduces for both directions which also makes sense for real-life applications. For instance, if a road is served for a snowplow operation then the cost of the next service reduces for both directions. In addition, we assume that $c_{ijk_1} \geq c_{ijk_2}$ if $k_1 < k_2$. That is, service costs do not become costlier as the number of the passes increases. Note that for each edge (i,j) we also define c_{jik} in the same manner to define the service cost for the direction from j to i. Finally, parameter $|E^h|$ represents the number of edges belonging to set E^h.

There are five sets of decision variables in our model. θ_{ht}, x_{ijt}, y_{ijt}, and α_{ijtk} are the binary variables of the model. θ_{ht} shows whether all edges of hierarchy h are served before step t, or not. x_{ijt} (similarly x_{jit}) indicates whether or not edge (i,j) is served at step t through direction from i to j (through direction from j to i), and y_{ijt} takes value 1 if edge (i,j) is served before or exactly at step t independently from the passage direction, and it is 0 otherwise. Finally, α_{ijtk} is 1 if edge (i,j) is served exactly k times before or at step t independently from the direction, and it is 0 otherwise. Our model results to be an integer nonlinear program since the objective function involves a multiplication of the binary variables α_{ijtk} and x_{ijt} (α_{ijtk} and x_{jit}). For the sake of convenience, a summary of all the sets, parameters and variables used along our formulation are provided in Table 1.

2.2 Mathematical Model

Our integer nonlinear program abbreviated as V-HWPPF (Variable HWPP Formulation) is given as:

$$\min \sum_{(i,j)\in E} \sum_{t\in T} \sum_{k\in K} \left(c_{ijk}\alpha_{ijtk}x_{ijt} + c_{jik}\alpha_{ijtk}x_{jit} \right) \tag{1}$$

s.t.

Table 1 Sets, indices, parameters and decision variables

Sets/indices	Definition
$N/i,j$	Set of nodes/node indices
$E/(i,j)$	Set of edges/edge indices
H/h	Set of hierarchies/hierarchy index
E^h	Set of edges belonging to hierarchy h
$T/t,m$	Set of steps/step indices
K/k	Set of the number of passes/number of passes index
Parameters	Definition
c_{ijk}	Service cost of the edge (i,j) at the kth pass
$\lvert E^h\rvert$	Number of edges belonging to hth hierarchy
Variables	Definition
θ_{ht}	Indicates whether or not all edges belonging to E^h are served until step t
$x_{ijt}(x_{jit})$	Indicates whether or not edge (i,j) is served at step t through the direction from i to j (through the direction from j to i)
y_{ijt}	Indicates whether or not edge (i,j) is served before or at step t in total, independently from the direction
α_{ijtk}	Indicates whether or not edge (i,j) is served k times before or at step t independently from the direction
$\mu_{ijkt}(\mu_{jikt})$	Represents the multiplication $\alpha_{ijtk}x_{ijt}$ $(\alpha_{ijtk}x_{jit})$

$$\sum_{j\in N} x_{1j1} = 1 \tag{2}$$

$$\sum_{j\in N} x_{j1|T|} = 1 \tag{3}$$

$$\sum_{(i,j)\in E} \left(x_{ijt} + x_{jit}\right) = 1 t \in T \tag{4}$$

$$\sum_{t\in T} \left(x_{ijt} + x_{jit}\right) \geq 1 (i,j) \in E \tag{5}$$

$$\sum_{(i,j)\in E} x_{jit} - \sum_{(i,j)\in E} x_{ij(t+1)} = 0 i \in N, t \in T\backslash\{|T|\} \tag{6}$$

$$\sum_{k\in K} k\alpha_{ijtk} = \sum_{m=1}^{t} \left(x_{ijm} + x_{jim}\right)(i,j) \in E, t \in T \tag{7}$$

$$\sum_{m=1}^{t} \left(x_{ijm} + x_{jim}\right) \leq t y_{ijt}(i,j) \in E, t \in T \tag{8}$$

$$y_{ijt} \leq \sum_{m=1}^{t} \left(x_{ijm} + x_{jim}\right)(i,j) \in E, t \in T \tag{9}$$

$$|E^h|\theta_{ht} \leq \sum_{(i,\,j)\in E^h} y_{ijt}. \tag{10}$$

$$\sum_{(i,j)\in E^{(h+1)}} y_{ijt} \leq |E^{(h+1)}|\theta_{ht} h \in H\backslash\{|H|\}, t \in T \tag{11}$$

$$\theta_{ht}, x_{ijt}, x_{jit}, y_{ijt}, \alpha_{ijtk} \in \{0,1\}(i,j) \in E, h \in H, t \in T, k \in K \tag{12}$$

The objective function (1) minimizes the total service cost. Constraint (2) and constraint (3) ensure that vehicle leaves the first node (depot) at step 1 and goes back to the same node at the last step, $|T|$. Note that $|T|$ is an upper limit value for the number of steps that can be considered in any day (we selected its value as the number of edges times twice the number of hierarchies). In order to avoid forcing the vehicle to perform exactly $|T|$ steps on each day, we added a dummy arc that connects the starting node to itself and that has a traversal cost equal to 0. Thus, if on any day the vehicle needs to return to the starting point with fewer steps than $|T|$, it can return to the starting node at the end of step $|T|$ by traversing the dummy arc at 0 cost during the remaining steps. This ensures the satisfaction of constraint (3) for that day. Constraint (4) guarantees that the vehicle travels through a single edge at each step. By constraint (5) we make sure that each edge is traversed at least once independently from the direction of the passing. Constraint (6) ensures that the vehicle leaves any node i at step $t+1$ if it entered to it at the previous step t, and moreover it cannot leave node i at step $t+1$ if it did not enter it at the previous step t. Constraint (7) determines the value of each variable α_{ijtk} that takes value 1 depending on the number of travels through each edge before or at step t. The constraint ensures that if α_{ijtk} is 1, then the number of travels through the edge (independent of the direction of the passing) is equal to k. Note that only one of the α_{ijtk} variables existing on the left-hand side of the constraint will be 1 in the optimal solution, although it is possible to satisfy the constraint by setting more than one α_{ijtk} variables to 1. This is because the objective is a minimization function and the variable service costs are chosen to decrease by each pass. For instance, suppose that the number of total passes of an edge $(i,j) \in E$ before or step t is 5. Hence, the right-hand side of constraint (7) written for edge (i,j) and for period t will be 5. It is possible to assign, for instance, both α_{ijt2} and α_{ijt3} to 1 (and all other α_{ijtk} variables to 0) which will make the left-hand side of the constraint as $2 \times \alpha_{ijt2} + 3 \times \alpha_{ijt3}$ which is equal to 5. Another alternative is, of course, to set only α_{ijt5} variable to 1, making the left-hand side of the constraint as $5 \times \alpha_{ijt5}$ which is also equal to 5. Since the cost of the latter alternative in the objective function, which is c_{ij5}, is lower than the cost of the first alternative, which is $c_{ij2} + c_{ij3}$, the solver will naturally choose the latter alternative. Note that if an edge, say (i,j), is traversed through i to j, then the next passage cost should be decreased not only for the passage direction from i to j but also for the passage direction from j to i. Hence, if an edge $(i,j) \in E$ is traversed for instance 4 times in total independent of the passage direction, its fifth passage cost will be c_{ij5} if the passage direction is from i to j and c_{ji5} if the passage direction is from j to i. Constraints (8) and (9) define the relationship between x_{ijt}, x_{jit} variables and y_{ijt} variable. Namely, if the number of travels through an edge is zero until a particular step, i.e., $\sum_{m=1}^{t}(x_{ijm} + x_{jim}) = 0$, then $y_{ijt} = 0$ follows. On the contrary, if $y_{ijt} = 1$, then $t \geq \sum_{m=1}^{t}(x_{ijm} + x_{jim}) > 0$ must hold. Constraint (10) defines the variable θ_{ht}, i.e., it guarantees that θ_{ht} variable is equated to 0 if if any edge belonging to hierarchy h is not traversed before or during step t. Similarly, constraint (11) avoids traversing edges belonging to hierarchy $h+1$ if all the edges of hierarchy h are not traversed yet. Finally, constraint (12) puts the usual binary restrictions.

The above nonlinear model can be easily linearized by introducing a new continuous variable μ_{ijkt} (μ_{jikt}) to replace each multiplication $\alpha_{ijtk}x_{ijt}$ $(\alpha_{ijtk}x_{jit})$. We also introduce seven new sets of constraints, which are linear and force $\mu_{ijtk}(\mu_{jitk})$ to be equal to $\alpha_{ijtk}x_{ijt}$ $(\alpha_{ijtk}x_{jit})$. These constraints are given in the following.

$$\mu_{ijtk} \leq \alpha_{ijtk}(i,j) \in E, t \in T, k \in K \quad (13)$$

$$\mu_{jitk} \leq \alpha_{ijtk}(i,j) \in E, t \in T, k \in K \quad (14)$$

$$\mu_{ijtk} \leq x_{ijt}(i,j) \in E, t \in T, k \in K \quad (15)$$

$$\mu_{jitk} \leq x_{jit}(i,j) \in E, t \in T, k \in K \quad (16)$$

$$\mu_{ijtk} \geq \alpha_{ijtk} + x_{ijt} - 1(i,j) \in E, t \in T, k \in K \quad (17)$$

$$\mu_{jitk} \geq \alpha_{ijtk} + x_{jit} - 1(i,j) \in E, t \in T, k \in K \quad (18)$$

$$\mu_{ijtk}, \mu_{jitk} \geq 0(i,j) \in E, t \in T, k \in K \quad (19)$$

As can be understood, if α_{ijtk} and x_{ijt} (x_{jit}) take value 1, then μ_{ijtk} is equated to 1 by means of constraints (13), (15) and (17) (constraints (14), (16) and (18)). Similarly, if α_{ijtk} or x_{ijt} (α_{ijtk} or x_{jit}) is zero, μ_{ijtk} (μ_{jitk}) is set to zero by means of constraints (13), (15) and (19) (constraints (14), (16) and (19)). This implies that the nonlinear multiplication $\alpha_{ijtk}x_{ijt}$ $(\alpha_{ijtk}x_{jit})$ can be replaced by μ_{ijtk} (μ_{jitk}) after adding constraints (13)–(19) to the model.

Note that if there is only one hierarchy and if the cost parameters are taken as constant then the V-HWPP reduces to the WPP which is known to be an NP-Complete problem. Hence, V-HWPP is also a difficult problem and the solution time for formulation (1)–(19) will increase exponentially with the size of the instances which is characterized by the number of nodes, edges and hierarchies. For this reason, we will propose in the next section a heuristic method that is able to tackle even large-scale instances.

3 Windy layer algorithm heuristic

In our heuristic approach, we adapt the layer algorithm of Dror et al. (1987), that solves the HCPP with linear precedence relationships and edges connected within each hierarchy to our variant of the problem. Note that the exact solution methods of Ghiani and Improta (2000) and Korteweg and Volgenant (2006) are more efficient in terms of computation time, but their methods also depend on the layer algorithm as well. The layer algorithm employs the famous blossom method of Edmonds (1965) as a sub-procedure to find the path that goes through all the edges of the current hierarchy without passing through any edge from the lower hierarchies. In the sequel, we will call, for convenience, this path as the least cost path. However, the blossom algorithm requires the edge traversal costs to be symmetric and as our problem is a windy postman problem the edge traversal costs possess an asymmetric nature. Hence, we construct here mathematical models and run them as sub-procedures in order to find the least cost paths, passing through each edge belonging to the current hierarchy, between any two nodes within the layer algorithm. We will refer to this heuristic as the Windy Layer Algorithm (WLA). The details of such least cost path mathematical models are presented in the following after explaining the WLA in the context of the V-HWPP.

Let $G^h = (N^h, E^h)$ be the subgraph induced by the set of edges of hierarchy h. Here, N^h represents the incident nodes of E^h. Define F^q as the subgraph induced by the first q sets of edges, i.e., $F^q = \bigcup_{h=1}^{q} G^h$. We assume that subgraphs F^q are connected for all $q = 1, \ldots, |H|$. Moreover, we define node $i \in N^h$ as an entry node of G^h if it is also incident to an edge of F^{h-1}. The set of entry nodes of G^h is called as Y^h. Mathematically, $Y^h = \left(\bigcup_{q=1}^{h-1} N^q\right) \bigcap N^h$.

Finally, we define $Y^1 = \{i^1\}$ and $Y^{|H|+1} = \{i^1\}$ where i^1 is the depot which is the initial and the final node. Now, we construct a new graph $G' = (N', E')$ with the node set defined as $N' = \bigcup_{h=1}^{|H|+1} Y^h$. Note that if a node exists in more than one of the sets $Y^h, h = 1, \ldots, |H| + 1$, it will be treated as a different node each time. Hence, the nodes will exist exactly the number of times they exist in the sets $Y^h, h = 1, \ldots, |H| + 1$. On the other hand, the edge set is defined as $E' = \{(i,j) : i \in Y^h, j \in Y^{h+1} \text{for} h = 1, \ldots, |H| + 1\}$. Namely, we define an edge for each pair that belongs to successive Y^h sets. The graph G' is depicted in Fig. 1.

Note that if edge $(i,j) \in E'$ then $i \in Y^h$ and $j \in N^{h+1}$ for some $h = 1, \ldots, |H|$ and going from i to j in G' means that each edge belonging to E^h is traversed starting from node i and ending at node j by following the least cost path (edge lengths are taken as service costs) without violating the hierarchy restrictions. Therefore, the cost of each edge in E', i.e., cost of edge (i,j), is determined by finding the path having the least cost between node i and node j while traveling through every edge of E^h without passing through any edge from lower hierarchies. As mentioned earlier, instead of employing the blossom algorithm, we make use of mathematical programs in order to find the least cost paths between the nodes. Let $P(i,j,h)$ denotes the problem of finding the least cost path between node i and node j that travels through every edge of E^h without passing through any edge from lower hierarchies. Now suppose that u and v represent the nodes for which $(u,v) \in E$ and β^i_{uv} (β^i_{vu}) indicates the number of times edge (u,v) is traversed through direction from u to v (from v to u) along the least cost path from the starting point until node i without violating the hierarchy restrictions. Note that $P(i,j,h)$ is an open WPP and any edge may be traversed more than twice in the optimal solution. Nevertheless, we restrict the number of traversals to two in order to obtain a very fast solution of $P(i,j,h)$ since this problem will be solved very often throughout the algorithm (i.e., at every step corresponding to different values of i, j and h). We are aware that it is possible, for some instances, that a vehicle may need to pass some edges more than twice in the optimal solution. However, we observed through preliminary empirical experiments that the rate of such occurrence is very low and restricting the number of passes to two will not harm the optimality for most of the instances while contributing in significatively reducing the overall solution time. Moreover, we drop index t from x_{ijt} variables in the problem. Consequently, we define x_{uv} (x_{vu}) as a continuous variable that indicates the number of times edge (u,v) is traversed through the direction from u to v (from v to u). We also define two new sets of variables y_{uv} and z_{uv} (y_{vu} and z_{vu}) that, respectively, indicate if (u,v) is traversed once or twice through the direction from u to v (from v to u) in each solution of $P(i,j,h)$. The formulation of problem $P(i,j,h)$ with these new set of variables, for each i, j and h, is given below.

$$\min \sum_{(u,v)\in E} c_{uv\left(\beta^i_{uv}+1\right)} y_{uv} + c_{vu\left(\beta^i_{vu}+1\right)} y_{vu} + \left(c_{uv\left(\beta^i_{uv}+1\right)} + c_{uv\left(\beta^i_{uv}+2\right)}\right) z_{uv} + \left(c_{vu\left(\beta^i_{vu}+1\right)} + c_{vu\left(\beta^i_{vu}+2\right)}\right) z_{vu} \tag{20}$$

s.t.

$$x_{uv} + x_{vu} \geq 1 (u,v) \in E^h \tag{21}$$

$$\sum_{(u,v)\in E} x_{uv} - \sum_{(u,v)\in E} x_{vu} = 0 u \in N : u \neq i,j \tag{22}$$

$$\sum_{(i,v)\in E} x_{iv} - \sum_{(i,v)\in E} x_{vi} = 1 \tag{23}$$

$$\sum_{(j,v)\in E} x_{jv} - \sum_{(j,v)\in E} x_{vj} = -1 \tag{24}$$

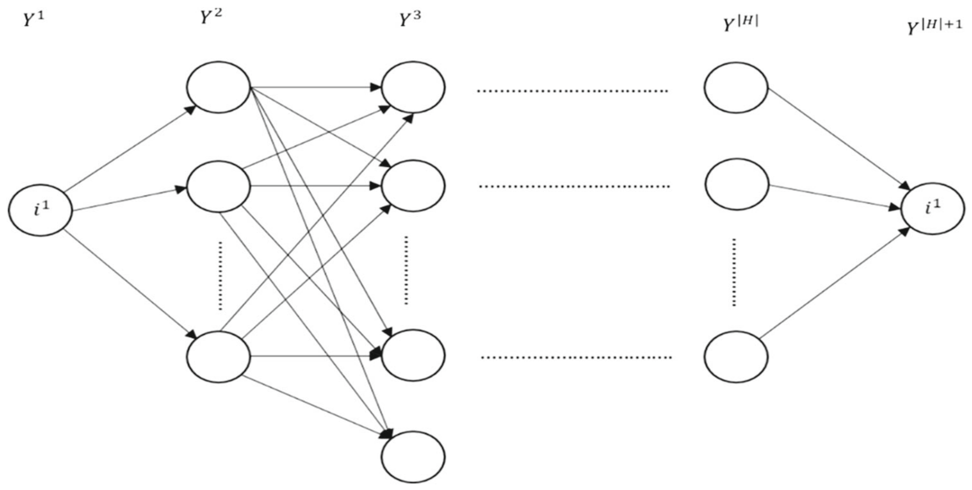

Fig. 1 The graph G' (from Dror et al. (1987))

$$x_{uv} = y_{uv} + 2z_{uv}(u,v) \in E \tag{25}$$

$$x_{vu} = y_{vu} + 2z_{vu}(u,v) \in E \tag{26}$$

$$x_{uv} = 0(u,v) \in E^{h'}, h' = h+1,\ldots,|H| \tag{27}$$

$$x_{vu} = 0(u,v) \in E^{h'}, h' = h+1,\ldots,|H| \tag{28}$$

$$x_{uv} \le 2(u,v) \in E \tag{29}$$

$$x_{vu} \le 2(u,v) \in E \tag{30}$$

$$x_{uv}, x_{vu} \ge 0(u,v) \in E \tag{31}$$

$$y_{uv}, y_{vu}, z_{uv}, z_{vu} \in \{0,1\} \quad (u,v) \in E \tag{32}$$

We minimize the total traversal cost in (20). Note that an edge belonging to higher hierarchies (i.e., an edge from $E^{h'}$ $h' = 1,2,\ldots,h-1$), may have been passed several times before we start to traverse the edges of E^h. As the traversal costs of the edges vary with the number of passes through edges, we keep track of the number of passes for each edge and for each direction.

If edge (u,v) is traversed through u to v β^i_{uv} times before, its next traversal will be the $(\beta^i_{uv}+1)^{st}$ time. Hence, y_{uv} variable (which indicates whether or not edge (u,v) is traversed from u to v once in $P(i,j,h)$, i.e., during traversing edges of E^h without passing through any edge from lower hierarchies starting from node i and ending with node j) is multiplied by the corresponding cost $c_{uv(\beta^i_{uv}+1)}$. Similarly, z_{uv} variable (which indicates whether or not edge (u,v) is traversed from u to v twice in $P(i,j,h)$) is multiplied by the corresponding total cost which is $(c_{uv(\beta^i_{uv}+1)} + c_{uv(\beta^i_{uv}+2)})$. In a similar manner, the objective function coefficients related to y_{vu} and z_{vu} variables are, respectively, determined as $c_{vu(\beta^i_{vu}+1)}$ and $(c_{vu(\beta^i_{vu}+1)} + c_{vu(\beta^i_{vu}+2)})$. However, we still need to adjust the objective function value after obtaining the solution. The point is that after passing through an edge (u,v) from u to v, the cost of passage from v to u should also be decreased and vice versa, and this reality is not captured in the objective function (20). Hence, after solving $P(i,j,h)$ problem, we determine the exact route of the vehicle making use of the values of the x, y and z variables and adjust the objective function value accordingly. Therefore, after solving $P(i,j,h)$ for each edge (i,j) in E' for each hierarchy h, we adjust the objective function value depending on the route of the vehicle obtained from the x_{uv} and x_{vu} values. One may object here that $P(i,j,h)$ should have been constructed in such a way that the objective function is calculated correctly. However, in order to be able to drop the t index from x variables and in order to construct $P(i,j,h)$ without using the α variables (that has four indices) we had to construct $P(i,j,h)$ formulation as given by (21)−(33) and adjust the objective function later. This is a reason for which our method is not an exact method but a heuristic. By constraint (21) we make sure that each edge from E^h is traversed at least once independently from the direction of the passing. Constraint (22) maintains the flow balance for each node different from i and j. In other words, constraint (22) guarantees that the total number of enters to a node different than i and j is equal to number of exits from the same node. On the other hand, as the journey starts from node i and finishes at node j for $P(i,j,h)$, the total number of exits minus total number of enters should be equal to 1 and -1 for nodes i and j, respectively. These requirements are satisfied by the help of constraints (23) and (24), respectively. Constraint (25) and (26) define y_{uv} and z_{uv} (y_{vu}, z_{vu}) variables and sets their relationship with x_{uv} (x_{vu}) variables. Constraints (27) and (28) avoid passing through an edge belonging to lower-level hierarchies and constraints (29) and (30) limit the number of passages of edges by 2 for both directions. Finally, constraint (31) defines x_{uv} and x_{vu} as continuous variables while constraint (32) defines y_{uv}, y_{vu}, z_{uv}, and z_{vu} as binary variables. It should be noted that each x_{uv} and x_{vu} can take only the integer values 0, 1 or 2, but we can define them as a continuous variable (to improve the efficiency of the model solution) knowing that they cannot take fractional values due to constraints (25) and (26).

Note that although problem $P(i,j,h)$ is an open WPP, it does not require subtour elimination constraints. This is because the edges of each hierarchy are connected within themselves. On the contrary, suppose there is a path from i to j including some (or all) edges of hierarchy h and probably some edges from higher hierarchies and a subtour that is disconnected from the path. If the disconnected subtour does not include any edge from hierarchy h, then it will not be included in the optimal solution since it is not necessary to pass through edges included in the subtour. Hence, any subtour that does not include any edge from hierarchy h will be eliminated by the solver. On the other hand, if there is at least one edge from hierarchy h in the subtour, then there must be at least one edge from hierarchy h connecting the subtour with the path that makes the subtour and the path connected which is a contradiction. This implies that although formulation (20)–(32) does not include subtour elimination constraints, disconnected subtours are not formed in the optimal solution due to the connected nature of the edges within the hierarchies.

By running $P(i,j,h)$ for each $(i,j) \in E'$ such that $i \in Y^h$ and $j \in N^{h+1}$ for each $h = 1,\ldots,|H|$, the length of every edge of graph G' can be calculated, i.e., the length of each $(i,j) \in E'$ such that $i \in Y^h$ and $j \in N^{h+1}$ for some $h = 1,\ldots,|H|$ is the optimal value of the related $P(i,j,h)$. Then, a shortest path from $i^1 \in Y^1$ to $i^1 \in Y^{|H|+1}$ can be easily found by Dijkstra's label algorithm (Dijkstra 1959). We do not think it is necessary to give details of this very well-known algorithm here. However, along the algorithm, it is necessary to calculate the number of times each edge is passed through each direction while proceeding on the shortest path from the beginning node to each point. In order to achieve this, if shortest path (from $i^1 \in Y^1$ to $i^1 \in Y^{|H|+1}$ in graph $G^{'}$) pass through node i just before node j, we define $\beta^j_{uv} = \beta^i_{uv} + x_{uv} + x_{vu}$ and $\beta^j_{vu} = \beta^i_{vu} + x_{uv} + x_{vu}$ where x_{uv} and x_{vu} values come from the solution of $P(i,j,h)$. By this updating mechanism, it is ensured that each β^i_{uv} is always equal to β^i_{vu} for every edge (u, v) and for each $i \in N$.

The layer algorithm in which problem $P(i,j,h)$ is solved by the above-mentioned mathematical model could have been an exact solution algorithm for the V-HWPP. However, the solution of the $P(i,j,h)$ problems take more and more computational time as the size of the original network increases. Hence, during the implementation phase, we avoided traversing the edges more than twice (as mentioned before) and we forced the commercial solver Gurobi to finalize the run as soon as it finds a feasible solution having an objective function value that is at most 5% away from the optimal value and returns the feasible solution it finds. Consequently, the specific variant of the layer algorithm WLA that we implement and define here results to be a heuristic procedure.

We conclude this section by a small illustrative example involving a network with five nodes and six edges, as shown in Fig. 2. The numbers reported on both sides of the edges indicate the passage costs through each direction for that edge. We also suppose that there are two hierarchies, for which edges (1, 2), (1, 4), (2, 5) and (4, 5) constitute hierarchy 1, whereas (2, 3) and (2, 4) constitute hierarchy 2 edges. Moreover, we also assume that the traversal costs are reduced by half after each pass. For example, the cost of the second pass through edge (1, 2) from 1 to 2 is 47 (which is the half of 94, the first passage cost).

One may notice that there are two entry nodes for hierarchy 2 which are node 2 and node 4 since these are the nodes that are incident to the edges of hierarchy 2 and hierarchy 1. Hence, graph $G^{'}$ that is constituted of the entry nodes of the network will be similar to the network given in Fig. 3.

The length of edge (1, 2) in $G^{'}$ can be calculated by solving $P(1,2,1)$ and it is easy to see that the optimal path starting from 1 an ending at 2 that passes through each edge of hierarchy 1 is simply 1–4-5–2-1–2 with a total cost of 36 + 30 + 32 + 50 + 47 = 195. Note that the cost of passage from 1 to 2 is taken as 47 since edge (1, 2) is already traversed from 2 to 1. Similarly, the optimal path starting from 1 and ending at 4 that passes through each edge of hierarchy 1 is 1–4-5–2-1–4 with a total cost of 36 + 30 + 32 + 50 + 18 = 166. Moreover, the optimal path starting from 2 and ending at the starting node 1 that passes through each edge of hierarchy 2 is 2–3-2–4-1 with a total cost 25 + 21 + 33 + 22.5 = 101.5. Note that the passage cost from node 3 to 2 is taken as 21 (half of 42, the first passage cost) since edge (2, 3) is already traversed from 2 to 3 and the passage cost from node 4 to 1 is taken also as 22.5 (half of 45) since it is already traversed during the first leg in which all the hierarchy 1 edges are passed. Besides, the optimal path starting from 4 and ending at the starting node 1 that passes through each edge of hierarchy 2 is 4–2-3–2-1 with a total cost of 18 + 25 + 21 + 25 = 89. Note that passage costs from 3 to 2 and from 2 to 1 are also taken as half of the original costs since the related edges are passed before. Finally, the shortest path from $i^1 = \{1\}$ to $i^3 = \{1\}$ in graph $G^{'}$ is simply 1–4-1 with a total cost of 166 + 89 = 255. The optimal path corresponds to 1–4-5–2-1–4-2–3-2–1 in the original network with a total cost of 255.

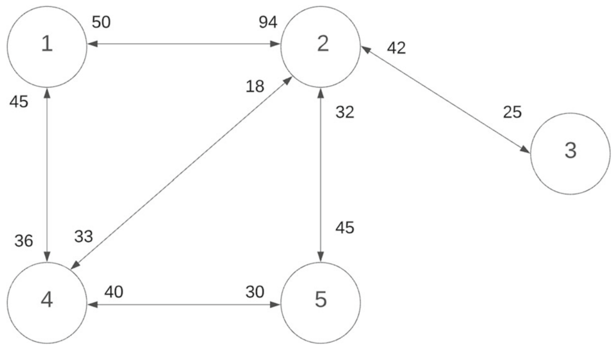

Fig. 2 A simple illustrative example for WLA

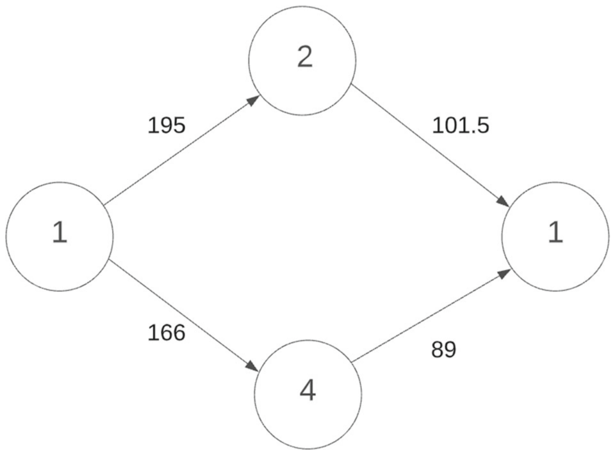

Fig. 3 Network $G^{'}$ for the illustrative example

4 Computational experiments

In this section, the selection of the parameters of the V-HWPPF and the generation of the test instances are given in the first part, and then the efficiency and accuracy of our heuristic approach are illustrated on extensive set of test instances.

4.1 Selection of the parameters and instance generation

Given that problem V-HWPP and its formulation are being introduced for the first time in this study, there are no test instances related to such problem in the existing scientific literature. We generate, thus, a total of 84 test instances having 10, 20, 30, 40, 50, 75, and 100 nodes each with three different edge numbers varying from 13 to 3300 and four different levels of hierarchies. If n is the number of nodes, then the three different numbers of generated edges are defined as $\lceil \frac{n\times(n-1)}{3} \rceil, \lceil \frac{n\times(n-1)}{5} \rceil, \lceil \frac{n\times(n-1)}{7} \rceil$, respectively. For example, for a 10-node test problem, we generate three different instances having $\lceil \frac{10\times 9}{3} \rceil = 30, \lceil \frac{10\times 9}{5} \rceil = 18, \lceil \frac{10\times 9}{7} \rceil = 13$ edges. We then randomly assign a number of edges to each level of hierarchy so that they sum up to the total edges number. For instance, if the number of edges is 13 and the number of hierarchies is 3, then the number of edges in hierarchies 1, 2 and 3 can take the values of 3, 6 and 4, respectively. In order to literally create the edges, we randomly select two nodes and form an edge and assign its hierarchy (starting from the first hierarchy) if the edge is connected to the previously generated edges belonging to the same hierarchy. If the created edge is not connected to the previously generated edges belonging to the same hierarchy, then we delete the edge and repeat the procedure until the number of edges belonging to the hierarchy reaches the assigned number of edges for that hierarchy. We then proceed to generate the edges of the next hierarchy. For instance, if the number of edges in hierarchy 1 is 3, then the first edge is created randomly between two randomly selected nodes and its hierarchy is assigned as 1. We then keep randomly generating the next edges until we find an edge that is connected to the first edge and assign its hierarchy as 1 too. We repeat the same procedure for edge 3 and assign its hierarchy as 1 and then proceed for generation of edges of hierarchy 2, and so on. Moreover, the first edge of a given hierarchy is forced to be connected to at least one of the higher hierarchy edges. In this way, we ensured that hierarchies are interconnected and that the edges within each hierarchy show the linear relationship. The number of hierarchies are selected as two, three, four and five, implying that we have four different number of hierarchies.

The first passage (or service) cost of each edge (i,j) is generated randomly according to the expression $c_{ij1} = 30 + 70 \times rand(0,1)$ where $rand(0,1)$ is a uniform random number generated from $(0,1)$ interval. c_{ji1} values for each edge (i,j) are also calculated in a similar manner. Hence, the first passage costs are random values within the interval $(30,100)$ for all edges. The values of the other cost parameters, i.e., c_{ijk} and c_{jik} values for each $(i,j) \in E, k > 1$, are generated using the formulas $c_{ijk} = \frac{c_{ij(k-1)}}{2} + \mathrm{rand}(\frac{c_{ij(k-1)}}{2})$ and $c_{jik} = \frac{c_{ji(k-1)}}{2} + \mathrm{rand}(\frac{c_{ji(k-1)}}{2})$ where $\mathrm{rand}(m)$ denotes a real number randomly generated within the interval $(0,m)$. Note that the resulting c_{ijk} and c_{jik} values naturally have a non-increasing structure, and they are all positive due to the employed generation mechanism.

4.2 Accuracy and efficiency of the WLA heuristic

In this section, we assess the performance of the layer algorithm by comparing the minimum distances found by the WLA heuristic with the distances and the minimum possible lower bound values reported by the state-of-the-art MILP commercial solver Gurobi (2020) on the generated test sets. We code the WLA in Visual Studio environment with C# language and we carry out all experiments using a single Intel i7-8750H core. We let Gurobi run for at most three hours for each of the test instances and the minimum objective function values and the lower bound values found in the allowed computation times are reported. However, although Gurobi is able to produce feasible solutions for all instances with 10 nodes, it cannot produce feasible solution for any instance having more than 10 nodes. In addition, Gurobi is not able to produce even a lower bound larger than zero for four instances with 10 nodes and for none of the instances having 20 nodes. On the other hand, WLA produces feasible solutions for all the instances even for the largest instances having 100 nodes. The objective function values and the lower bound values found by Gurobi, and the objective function values found by WLA are reported in Table 3 (in the Appendix). For all the instances for which Gurobi was not able to produce an upper bound and/or a feasible solution, we used the notation NA (for non-applicable) in the related cell.

The first and second columns of Table 3 specify the number of nodes and number of edges, respectively, while the third column reports the number of hierarchies. The fourth and fifth columns report the lower bounds and minimum cost values obtained by Gurobi while the sixth column reports the minimum cost values found by WLA. Seventh and eighth columns represent respective percentage differences between the minimum costs found by WLA and Gurobi, and between the minimum cost found by WLA and the lower bound reported by Gurobi. The percentage

Table 2 Average performance measures of the mathematical model (solved by Gurobi) and the WLA

$\|N\|$	Solution value			Percentage		Time (s)	
	GLB	GUB	WLA	%GUB-WLA	%GLB-WLA	Gurobi	WLA
10	459.35	1795.57	1675.64	5.13	61.96	8390.97	1.43
20	0.00	NA	6055.93	NA	100.00	10,800.00	5.75
30	NA	NA	13,836.00	NA	NA	10,800.00	33.05
40	NA	NA	24,333.36	NA	NA	10,800.00	125.36
50	NA	NA	37,718.39	NA	NA	10,800.00	280.90
75	NA	NA	83,873.74	NA	NA	10,800.00	1223.36
100	NA	NA	148,717.12	NA	NA	10,800.00	3357.03

difference values are calculated as $100 \times \left(\frac{c_G - c_{WLA}}{c_{WLA}}\right)$ and $100 \times \left(\frac{c_{WLA} - LB}{c_{WLA}}\right)$ where c_{WLA} and c_G represent the cost values reported by WLA and Gurobi, respectively, while *LB* stands for the lower bound value produced by Gurobi. Finally, the nineth and tenth columns represent the computation times (in seconds) needed by Gurobi and WLA to reach the obtained solution, respectively.

One can extract from Table 3 that the mathematical model results found by Gurobi outperforms WLA (by 2.24%, %0.68, and %0.47, respectively) only for three instances having 10 nodes, which are, respectively, the instances with 13 edges and five hierarchies, 18 edges and four hierarchies, and 18 edges and five hierarchies. For the one with 13 edges and five hierarchies, Gurobi's result is also the optimal solution. Both methods are able to find the optimal solutions for two instances with 10 number of nodes, and for another two instances both methods produced the same result that is higher than the lower bound found by Gurobi. Finally, WLA is better than the mathematical model (respectively, by 4.42, 18.18, 10.26, 18.63, and 13.47%) for five of the instances with 10 nodes. In summary, the objective function values of WLA and Gurobi results for the instances with 10 nodes are similar but slightly in favor of WLA. The percentage difference between the average results for the instances with 10 nodes is 5.13% in favor of WLA. More interestingly, the average computation time used by Gurobi is 8390.97 s to reach these results while WLA spends only 1.43 s on average. Hence, for the smallest instances with 10 nodes, WLA is able to produce results that are at least as successful as the model's results found by Gurobi but using much less computation time. On the other hand, Gurobi is not able to produce a feasible solution for any instance having at least 20 nodes, although it uses the entire allotted time which is three hours. We denote in the sequel as the Gurobi instances those instances for which Gurobi is able to produce a feasible solution. For a better general view of the results, we report in Table 2 the average values (extracted from Table 3) and also visualize the same objective function values for Gurobi instances in Fig. 4.

Besides, we also report the LB values found by Gurobi in order to better assess the quality of WLA results. Nevertheless, Gurobi is able to produce a lower bound larger than zero for eight instances with 10 nodes and it is not able to produce any lower bound larger than zero for the instances with at least 20 nodes. This implies that Gurobi is not able to solve even the linear relaxation of the problem for those instances within the given three hours time limit. The instances for which Gurobi is able to produce a lower bound larger than zero is called as Gurobi lower bound instances. We illustrate WLA results and the lower bound values for the Gurobi lower bound instances in Fig. 5 below.

As can be seen from Fig. 5, WLA produces almost the same LB values for the relatively small instances having 13 edges. Indeed, the average percentage difference between WLA results and the LB values are only 4.76% for the instances with 10 nodes and 13 edges. However, the average difference increases to 81.12% for the instances with 18 edges and for the remaining instances Gurobi is run (that is the instances with 10 nodes and 30 edges and the instances with 20 nodes) Gurobi reports zero as a lower bound implying that the percentage difference to be 100%. However, this does not directly indicate the deterioration of WLA performance with the increase in the instance size since even the lower bound values found by Gurobi worsen as the problem sizes increases. As a matter of fact, WLA is able to produce feasible solutions for much larger instances, reaching 100-node networks, whereas lower bounds larger than zero cannot be found by Gurobi for instances with 10 nodes and 30 edges and for all instances with 20 nodes. This may indicate that the capability of Gurobi in generating lower bounds is very sensitive to the increase in the instances size. For instance, it is possible to observe a rapid decrease in the lower bounds provided by Gurobi when the number of edges is increased from 13 to 18 for the instances having 10 nodes, whereas the objective function values of WLA possess a smooth linear-like structure increasing nearly at a constant rate. Hence, the

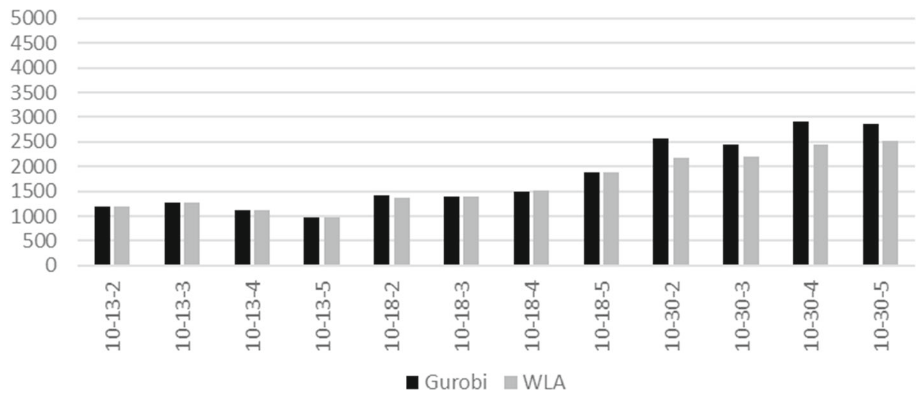

Fig. 4 Gurobi and WLA solution values for Gurobi Instances

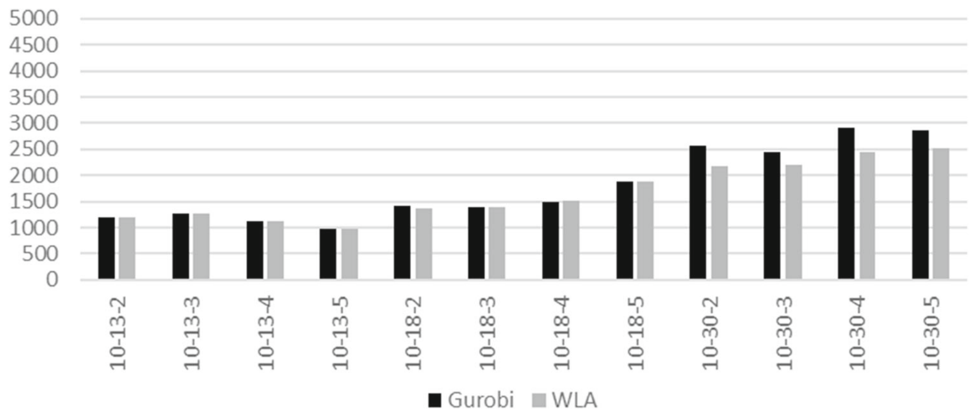

Fig. 5 Efficiency of WLA for Gurobi LB Instances

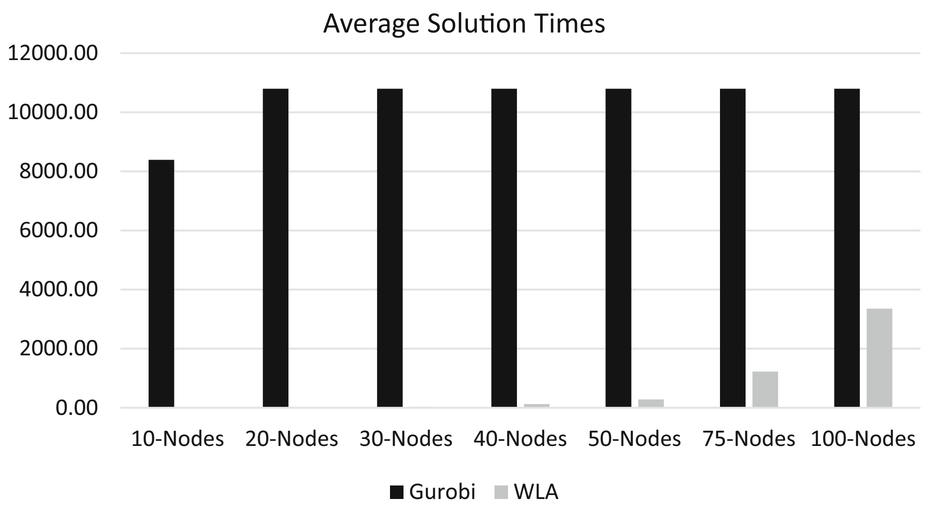

Fig. 6 Average computational times spent by V-HWPPF solved by Gurobi and WLA

increase in the percent deviation between WLA results and Gurobi's lower bounds cannot be attributed to the bad quality of WLA results but rather to the rapid deterioration in the reported Gurobi lower bound values as the instances size gets larger.

In addition, Fig. 6 reports the average computational times employed by Gurobi and WLA. One may note that the average time spent by WLA is so small for all instances with 10, 20 and 30 nodes so that they are almost at negligible level compared to the average times spent by Gurobi (this is the reason for which there is no visible average computation time of WLA for those instances). Another observation is that WLA also uses negligible amount of time (around 0.5 min) for all instances with 30 nodes on the average while it, respectively, uses around 2, 5 and 20, minutes for instances with 40, 50 and 75 nodes on the average. Finally, the average time spent for 100-node instances is around 56 min. On the other hand, the average

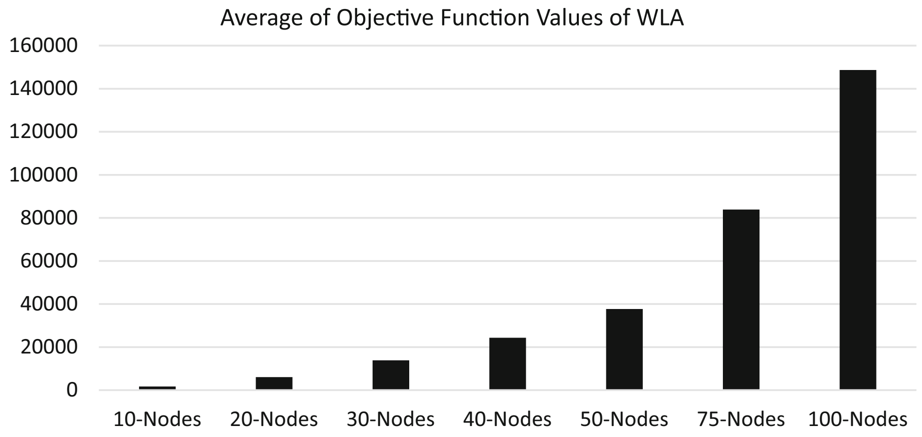

Fig. 7 Average objective function values found by WLA

computation time used by Gurobi, respectively, for 10-, and at least 20-node instances are 8390.97 and 10,800.00 s. Hence, Gurobi uses all the allotted computation time for all the instances with at least 20 nodes. In summary, it is possible to claim that not only WLA is more accurate than the mathematical model solved by the commercial solver Gurobi, in the sense that the costs of the routes generated by WLA are much lower than Gurobi instances, but also that WLA is more efficient in generating the routes in much less computation times.

Finally, we observe that the running time of WLA exceeds three hours for only one instance (with 100-node instances and 1980 edges). For this reason, we avoided to test our WLA on larger instances since we think that we have achieved already significant results while solving instances that can be met in several real-life applications. The objective function values found by WLA for all the instances are depicted, for convenience, in Fig. 7, where it is clear that the objective function values increase in a smooth manner, which is another indicator of persistent achievement of WLA.

5 Conclusions and future research

This study dealt with the Hierarchical Windy Postman Problem with variable cost, which has not been discussed in the existing scientific literature. First, we propose a mathematical model for the problem. Our results confirmed that the exact solution procedure was not able to solve large-scale problems. For this reason, a novel heuristic method, inspired from the layer algorithm of Dror et al. (1987), has been proposed to solve the problem. Several test instances have been randomly generated in order to analyze the performance of our heuristic approach and to compare it with the performance of the mathematical model solved by the commercial solver Gurobi. The obtained results have shown that our windy layer algorithm generates high quality solutions in a very limited amount of computational time compared to the mathematical model results solved by Gurobi for the smallest sized problems with 10-nodes. For the larger sized instances, the model solved by Gurobi fails to generate feasible solutions and upper bound values, if available, are huge. In the case of 10-node instances, both the methods produce, on average, very close objective function values. However, when the average solution times are compared in the same problems group, it can be seen that the windy layer algorithm generates solution in much smaller computation times. When the size of the problems grows, Gurobi fails to produce any feasible solution and not even lower bound values for slightly larger instances whereas our heuristic method succeeds to solve instances with up to 100 nodes, 3300 edges and five hierarchies.

This work can be expanded in different ways. First, the class of problems characterized by hierarchy relations that are not linear can be discussed. Another direction of research can be analyzing the situation where service costs are fuzzy or stochastic. A variant of the problem for which the edges have different number of service requirements or whose service is characterized by a periodic nature (see Triki et al. 2017) can also be studied. Moreover, the hierarchic rural postman variant of the problem, in which only a subset of edges is required to be served, is an exciting problem that should attract the attention of researchers in future. One may also try to add some valid inequalities to formulation (20)−(32) in order to produce a stronger formulation for the open WPP aiming to have shorter computation times. Similarly, employing a heuristic to solve the open WPP would be another interesting avenue of investigation. Finally, the variants of the problem that involves multiple vehicles and/or multiple depots can also be explored.

Appendix

See Table 3.

Table 3 Performance of the mathematical model (solved by Gurobi) and the WLA

|N|	|E|	|H|				%		Time (s)	
			GLB	GUB	WLA	%GUB-WLA	%GLB-WLA	Gurobi	WLA
10	13	2	1200.46	1200.46	1200.46	0.00	0.00	617.27	0.57
10	13	3	1267.22	1267.22	1267.22	0.00	0.00	993.90	0.97
10	13	4	934.55	1123.23	1123.23	0.00	16.80	10,800.00	0.43
10	13	5	965.99	965.99	988.08	-2.24	2.24	1880.46	1.78
10	18	2	324.27	1421.49	1361.30	4.42	76.18	10,800.00	1.06
10	18	3	282.80	1404.69	1404.69	0.00	79.87	10,800.00	1.91
10	18	4	233.23	1498.40	1508.71	-0.68	84.54	10,800.00	1.14
10	18	5	303.72	1875.31	1884.21	-0.47	83.88	10,800.00	1.52
10	30	2	0.00	2566.44	2171.61	18.18	100.00	10,800.00	1.69
10	30	3	0.00	2436.87	2210.11	10.26	100.00	10,800.00	3.27
10	30	4	0.00	2914.43	2456.74	18.63	100.00	10,800.00	1.99
10	30	5	0.00	2872.35	2531.26	13.47	100.00	10,800.00	0.85
Average			459.35	1795.57	1675.64	5.13	61.96	8390.97	1.43
20	55	2	0.00	NA	3847.57	NA	100.00	10,800.00	0.71
20	55	3	0.00	NA	4134.82	NA	100.00	10,800.00	3.10
20	55	4	0.00	NA	4374.03	NA	100.00	10,800.00	2.85
20	55	5	0.00	NA	4461.89	NA	100.00	10,800.00	3.48
20	76	2	0.00	NA	5265.01	NA	100.00	10,800.00	0.74
20	76	3	0.00	NA	5630.43	NA	100.00	10,800.00	4.12
20	76	4	0.00	NA	5666.24	NA	100.00	10,800.00	4.88
20	76	5	0.00	NA	5655.42	NA	100.00	10,800.00	8.72
20	127	2	0.00	NA	8217.50	NA	100.00	10,800.00	0.85
20	127	3	0.00	NA	8420.03	NA	100.00	10,800.00	10.65
20	127	4	0.00	NA	8740.73	NA	100.00	10,800.00	13.51
20	127	5	0.00	NA	8257.52	NA	100.00	10,800.00	15.46
Average			0.00	NA	6055.93	NA	100.00	10,800.00	5.75
30	125	2	0.00	NA	9318.09	NA	100.00	10,800.00	1.38
30	125	3	0.00	NA	8764.65	NA	100.00	10,800.00	22.13
30	125	4	0.00	NA	8870.25	NA	100.00	10,800.00	17.45
30	125	5	0.00	NA	9688.05	NA	100.00	10,800.00	32.08
30	174	2	0.00	NA	12,177.38	NA	100.00	10,800.00	2.04
30	174	3	0.00	NA	12,370.26	NA	100.00	10,800.00	40.49
30	174	4	0.00	NA	12,745.97	NA	100.00	10,800.00	53.89
30	174	5	0.00	NA	12,166.58	NA	100.00	10,800.00	36.93
30	290	2	0.00	NA	19,891.04	NA	100.00	10,800.00	2.04
30	290	3	0.00	NA	19,820.53	NA	100.00	10,800.00	42.06
30	290	4	0.00	NA	20,054.96	NA	100.00	10,800.00	53.13
30	290	5	0.00	NA	20,164.18	NA	100.00	10,800.00	92.95
Average			0.00	NA	13,836.00	NA	100.00	10,800.00	33.05
40	223	2	0.00	NA	15,348.27	NA	100.00	10,800.00	3.79
40	223	3	0.00	NA	15,872.37	NA	100.00	10,800.00	70.56
40	223	4	0.00	NA	15,927.84	NA	100.00	10,800.00	86.55

Table 3 (continued)

|N|	|E|	|H|				%		Time (s)	
			GLB	GUB	WLA	%GUB-WLA	%GLB-WLA	Gurobi	WLA
40	223	5	0.00	NA	16,309.26	NA	100.00	10,800.00	108.92
40	312	2	0.00	NA	21,777.71	NA	100.00	10,800.00	3.45
40	312	3	0.00	NA	21,309.96	NA	100.00	10,800.00	82.40
40	312	4	0.00	NA	22,295.42	NA	100.00	10,800.00	231.07
40	312	5	0.00	NA	22,117.61	NA	100.00	10,800.00	179.69
40	520	2	0.00	NA	34,921.08	NA	100.00	10,800.00	4.63
40	520	3	0.00	NA	34,586.27	NA	100.00	10,800.00	120.29
40	520	4	0.00	NA	35,404.84	NA	100.00	10,800.00	235.16
40	520	5	0.00	NA	36,129.67	NA	100.00	10,800.00	377.77
Average			0.00	NA	24,333.36	NA	100.00	10,800.00	125.36
50	350	2	0.00	NA	24,393.10	NA	100.00	10,800.00	5.21
50	350	3	0.00	NA	24,323.42	NA	100.00	10,800.00	163.05
50	350	4	0.00	NA	24,470.32	NA	100.00	10,800.00	295.62
50	350	5	0.00	NA	24,970.93	NA	100.00	10,800.00	336.17
50	490	2	0.00	NA	33,016.66	NA	100.00	10,800.00	3.42
50	490	3	0.00	NA	33,675.34	NA	100.00	10,800.00	265.61
50	490	4	0.00	NA	34,571.91	NA	100.00	10,800.00	504.00
50	490	5	0.00	NA	34,453.29	NA	100.00	10,800.00	557.09
50	817	2	0.00	NA	54,268.64	NA	100.00	10,800.00	4.07
50	817	3	0.00	NA	54,332.64	NA	100.00	10,800.00	92.58
50	817	4	0.00	NA	54,512.96	NA	100.00	10,800.00	389.03
50	817	5	0.00	NA	55,631.52	NA	100.00	10,800.00	754.90
Average			0.00	NA	37,718.39	NA	100.00	10,800.00	280.90
75	793	2	0.00	NA	52,733.83	NA	100.00	10,800.00	13.28
75	793	3	0.00	NA	54,413.05	NA	100.00	10,800.00	340.03
75	793	4	0.00	NA	53,693.33	NA	100.00	10,800.00	1578.85
75	793	5	0.00	NA	54,735.92	NA	100.00	10,800.00	1693.74
75	1110	2	0.00	NA	73,595.63	NA	100.00	10,800.00	7.59
75	1110	3	0.00	NA	74,285.55	NA	100.00	10,800.00	553.27
75	1110	4	0.00	NA	75,621.44	NA	100.00	10,800.00	1413.76
75	1110	5	0.00	NA	77,382.22	NA	100.00	10,800.00	3429.21
75	1850	2	0.00	NA	121,516.71	NA	100.00	10,800.00	11.83
75	1850	3	0.00	NA	121,699.91	NA	100.00	10,800.00	488.24
75	1850	4	0.00	NA	122,318.01	NA	100.00	10,800.00	2638.88
75	1850	5	0.00	NA	124,489.25	NA	100.00	10,800.00	2511.66
Average			0.00	NA	83,873.74	NA	100.00	10,800.00	1223.36
100	1415	2	0.00	NA	95,060.82	NA	100.00	10,800.00	14.43
100	1415	3	0.00	NA	94,225.24	NA	100.00	10,800.00	2164.34
100	1415	4	0.00	NA	96,885.60	NA	100.00	10,800.00	2815.96
100	1415	5	0.00	NA	95,833.80	NA	100.00	10,800.00	6560.48
100	1980	2	0.00	NA	131,410.62	NA	100.00	10,800.00	18.35
100	1980	3	0.00	NA	130,959.23	NA	100.00	10,800.00	2528.63
100	1980	4	0.00	NA	132,697.10	NA	100.00	10,800.00	5124.05
100	1980	5	0.00	NA	133,184.95	NA	100.00	10,800.00	10,950.08
100	3300	2	0.00	NA	216,576.78	NA	100.00	10,800.00	29.70
100	3300	3	0.00	NA	218,418.51	NA	100.00	10,800.00	1353.25
100	3300	4	0.00	NA	218,896.23	NA	100.00	10,800.00	3886.84

Table 3 (continued)

|N|	|E|	|H|				%		Time (s)	
			GLB	GUB	WLA	%GUB-WLA	%GLB-WLA	Gurobi	WLA
100	3300	5	0.00	NA	220,456.53	NA	100.00	10,800.00	4838.24
Average			0.00	NA	148,717.12	NA	100.00	10,800.00	3357.03

Author's contribution All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by all authors. The first draft of the manuscript was written by MEK and both the other authors revised and commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding Open access funding provided by Università del Salento within the CRUI-CARE Agreement. None.

Data availability Enquiries about data availability should be directed to the authors.

Declarations

Conflict of interest The authors of this research certify that there is no any affiliation with or involvement in any organization or entity with financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

References

Afanasev VA, van Bevern R, Tsidulko OY (2021) The hierarchical Chinese postman problem: the slightest disorder makes it hard, yet disconnectedness is manageable. Oper Res Lett 49(2):270–277

Akbari V, Shiri D, Salman FS (2021) An online optimization approach to post-disaster road restoration. Transp Res Part b Methodol 150:1–25

Alfa AS, Liu DQ (1988) Postman routing problem in a hierarchical network. Eng Optim 14(2):127–138

Aráoz J, Fernández E, Franquesa C (2013) GRASP and path relinking for the clustered prize-collecting arc routing problem. J Heuristics 19(2):343–371

Cabral EA, Gendreau M, Ghiani G, Laporte G (2004) Solving the hierarchical chinese postman problem as a rural postman problem. Eur J Oper Res 155(1):44–50

Çodur MK, Yılmaz M (2020) A time-dependent hierarchical Chinese postman problem. CEJOR 28(1):337–366

Colombi M, Corberán A, Mansini R, Plana I, Sanchis JM (2017a) The hierarchical mixed rural postman problem. Transp Sci 51(2):755–770

Colombi M, Corberán A, Mansini R, Plana I, Sanchis JM (2017b) The hierarchical mixed rural postman problem: polyhedral analysis and a branch-and-cut algorithm. Eur J Oper Res 257(1):1–12

Corberán Á, Plana I, Sanchis JM (2014) The Chinese postman problem on directed, mixed and windy graphs. Arc routing problems, methods, and applications, MOS-SIAM Series on Optimization (Chapter 4), In: Corberán Á, Laporte G (eds), SIAM Publications, Philadelphia, 2014, pp 65–84

Corberán Á, Eglese R, Hasle G, Plana I, Sanchis JM (2021) Arc routing problems: a review of the past, present, and future. Networks 77(1):88–115

Damodaran P, Krishnamurthi M, Srihari K (2008) Lower bounds for Hierarchical Chinese postman problem. Int J Ind EngTheory Appl Pract 15(1):36–44

Dijkstra EW (1959) A note on two problems in connexion with graphs. Numer Math 1(1):269–271

Dror M, Stern H, Trudeau P (1987) Postman tour on a graph with precedence relation on arcs. Networks 17:283–294

Dror M (ed) (2000). Kluwer Academic Publishers, Norwell

Dussault B, Golden B, Groër C, Wasil E (2013) Plowing with precedence: a variant of the windy postman problem. Comput Oper Res 40(4):1047–1059

Edmonds J (1965) Paths, trees, and flowers. Can J Math 17:449–467

Eiselt HA, Gendreau M, Laporte G (1995) Arc routing problems, part I: the Chinese postman problem. Oper Res 43(2):231–242

Gendreau M, Ghiani G, Guerriero E (2015) Time-dependent routing problems: a review. Comput Oper Res 64:189–197

Ghiani G, Improta G (2000) An algorithm for the hierarchical Chinese postman problem. Oper Res Lett 26(1):27–32

Ghiani G, Musmanno R, Paletta G, Triki C (2005) A heuristic for the periodic rural postman problem. Comput Oper Res 32(2):219–228

Ghiani G, Manni E, Triki C (2008) The lane covering problem with time windows. J Discrete Math Sci Cryptogr 11(1):67–81

Golden BL, Wong RT (1981) Capacitated arc routing problems. Networks 11(3):305–315

Gurobi optimizer 9.0 (2020). High-end libraries for math programming. http://www.gurobi.com/. Accessed Nov 2020.

Keskin ME, Yılmaz M (2019) Chinese and windy postman problem with variable service costs. Soft Comput 23(16):7359–7373

Keskin ME, Triki C (2022) On the periodic hierarchical Chinese postman problem. Soft Comput 26(2):709–724

Korteweg, P. (2002). Postman problems priorities and the concept of servicing. Salamanca.

Korteweg P, Volgenant T (2006) On the hierarchical Chinese postman problem with linear ordered classes. Eur J Oper Res 169(1):41–52

Lahyani R, Khemakhem M, Semet F (2015) Rich vehicle routing problems: from a taxonomy to a definition. Eur J Oper Res 241(1):1–14

Mandal SK, Pacciarelli D, Løkketangen A, Hasle G (2015) A memetic NSGA-II for the bi-objective mixed capacitated general routing problem. J Heuristics 21(3):359–390

Monroy IM, Amaya CA, Langevin A (2013) The periodic capacitated arc routing problem with irregular services. Discret Appl Math 161(4–5):691–701

Mourão MC, Pinto LS (2017) An updated annotated bibliography on arc routing problems. Networks 70(3):144–194

Oruc BE, Kara BY (2018) Post-disaster assessment routing problem. Transp Res Part b Methodol 116:76–102

Perrier N, Langevin A, Amaya CA (2008) Vehicle routing for urban snow plowing operations. Transp Sci 42(1):44–56

Quirion-Blais O, Langevin A, Trépanier M (2017) A case study of combined winter road snow plowing and de-icer spreading. Can J Civ Eng 44(12):1005–1013

Sayata UB, Desai NP (2015) An algorithm for Hierarchical Chinese postman problem using minimum spanning tree approach based on Kruskals's algorithm. In: Souvenir of the 2015 IEEE International Advance Computing Conference, IACC 7154702, pp 222–227

Shao S, Xu SX, Huang GQ (2020) Variable neighborhood search and tabu search for auction-based waste collection synchronization. Transp Res Part b Methodol 133:1–20

Sun J, Meng Y, Tan G (2015) An integer programming approach for the Chinese postman problem with time-dependent travel time. J Comb Optim 29(3):565–588

Triki C (2017) Solving the periodic edge routing problem in the municipal waste collection. Asia Pacific J Oper Res 34(03):1740015

Triki C, Akil J, Al-Azri N (2017) Optimising the periodic distribution of gas cylinders with customers priority. Int J Oper Res 28(2):279–289

Vidal T, Martinelli R, Pham TA, Hà MH (2021) Arc routing with time-dependent travel times and paths. Transp Sci 55(3):706–724