Soft Computing (2022) 26:709–724
https://doi.org/10.1007/s00500-021-06213-2

OPTIMIZATION

On the periodic hierarchical Chinese postman problem

Muhammed Emre Keskin[1,5] · Chefi Triki[2,3,4]

Accepted: 24 August 2021 / Published online: 1 November 2021

Abstract
This paper presents a mathematical formulation and a heuristic approach for a new variant of the Hierarchical Chinese Postman Problem (HCPP). Indeed, we introduce the concept of periodicity, and we define and solve, for the first time, the Periodic-HCPP, denoted as P-HCPP. Given that the resulting integer programming model makes use of a big number of binary variables and given the extended time horizon considered, 30 days in our case, the problem is characterized by a high level of complexity. However, our developed heuristic is able to solve instances having up to 40 nodes, 520 arcs and 5 hierarchies, whereas a general-purpose solver like Gurobi was not able to provide solutions for instances having more than 10 nodes. While the collected results are very encouraging, we provide at the end of this paper a set of possible future extensions of this work.

Keywords Hierarchical Chinese postman problem · Periodicity restrictions · Layer algorithm · Simulated annealing · Etc

1 Introduction and literature review

The importance of extending the planning horizon to consider multi-period horizons within routing problems has been recognized since several decades (Campbell and Wilson 2014). The objective is to ensure a better time-space consolidation in order to achieve transportation cost savings. Usually, serving customers (either placed at the nodes of a network or along its arcs) over an extended time horizon follows a specific pattern of periodicity. This means that every customer will be characterized by a given frequency (for example, once, twice or thrice per week) to be respected in order to ensure a balanced service distribution over the multi-period horizon. While the periodicity aspect in the context of solving node routing problems had intensive attention in the literature, the interest to solve periodic arc problems increased only recently. A non-exhaustive list of papers that dealt with the periodic traveling salesman problem includes Paletta (1992), Chao et al. (1995) and Paletta and Triki (2002). On the other hand, those that proposed optimization models and/or solution methods for the several variants of the periodic vehicle routing problems include Beltrami and Bodin (1974), Cordeau et al. (1997), Bommisetty et al. (1998) and Francis et al. (2006). Recently, several new works have directed their attention to the solution of periodic node routing problems arising in real-life applications such as, the containers transportation (Chen et al. 2020), the petrol stations replenishment (Triki 2013; Al-Hinai and Triki 2020), the distribution-inventory of perishable goods (Diabat et al. 2016), medical waste collection (Taslimi et al. 2020), patrolling service by security companies (Fröhlich et al. 2020), planning election logistics (Shahmanzari et al. 2020), the distribution of gas cylinders (Triki et al. 2017) and the train timetabling (Zhou et al. 2020). Most of these and other studies that have been published along the last 50 years were reviewed in the recent survey by Wang and Wasil (2020).

✉ Chefi Triki
C.Triki@kent.ac.uk

Muhammed Emre Keskin
emre.keskin@atauni.edu.tr

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

[2] Kent Business School, University of Kent, Canterbury, United Kingdom

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

[4] College of Sciences and Engineering, Hamad bin Khalifa University, Doha, Qatar

[5] Division of Engineering Management and Decision Sciences, College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar

This paper focuses on introducing the periodicity aspect within a specific variant of arc routing problems, namely the Hierarchical Chinese Postman Problem (HCPP). To the best of our knowledge, the periodic version of this problem has never been modeled or solved in the scientific literature. Our contribution consists here of filling in such gap by defining for the first time the Periodic-HCPP (i.e., P-HCPP), developing an original model for its formulation and suggesting a heuristic approach for its solution.

The HCPP arises in many application contexts such as street sweeping, garbage collection, flame cutting and snow removal (Liu 1988; Eiselt et al. 1995a, b; Perrier et al. 2008). In many of the above problems, the network's arcs do not need to be served every day of the planning horizon but are rather following a periodicity pattern that defines their service frequency. By extending the planning horizon and optimally consolidating the service over the different days, the decision maker can achieve significant cost saving and level of service improvement. Consequently, every arc in the P-HCPP network will be characterized, besides its traversal cost, by two further attributes: its frequency of service and its hierarchy class. The hierarchy of an arc represents its level of priority to be served before/after the other arcs (depending on whether they have higher/lower hierarchy). Consequently, the P-HCPP can be verbally defined as the problem of identifying a set of least-cost routes that start every day from a given depot node, visit the network's arcs according to their hierarchy level and periodicity frequency, and then come back to the same starting node at the end of each day of the planning horizon. A more formal definition of the P-HCPP, in terms of graph theory-based mathematical representation will be given in Sect. 2.

An illustrative example of the P-HCPP is drawn in Fig. 1. The colors identify the three classes of hierarchy of each arc (red, green and blue in a decreasing hierarchical order), and the number close to each arc represents its frequency of service (here assumed to be once, twice or thrice during the planning horizon, as the arcs replication shows). For the sake of simplicity, we avoided to include the arcs' traversal costs (with and without service) in this P-HCPP graph.

In the context of periodic arc routing, the P-HCPP finds its root in two different streams of research. The first stream is related to the HCPP that has been introduced in the pioneering work of Dror et al. (1987). The authors claimed that, despite the fact that the HCPP is an NP-hard, a class of it can be solved in polynomial time if the network obeys to certain conditions related to the linearity of the precedence relation and to the graph connectivity. A first heuristic method has been developed by Alfa and Liu (1988), whereas exact solution approaches for the HCPP have been proposed separately in Ghiani and Improta (2000), Cabral et al. (2004) and Korteweg and Volgenant (2006). More specifically, both the former works were based on the idea of transforming the HCPP into a rural postman problem and solving it by a branch-and-cut technique, whereas the latter work suggested an improvement of the above-mentioned algorithm of Dror et al. Damodaran et al. (2008) have focused on developing lower bounds for the HCPP, and Sayata and Desai (2015) have employed a minimum spanning tree technique to reduce the number of arcs in the underlying graph and reduce, thus, the problem's complexity. More recently, Çodur and Yılmaz (2020) and Keskin et al. (2021) defined new variants of the HCPP in which they introduce the concept of time-dependent traveling time between the nodes. Both the

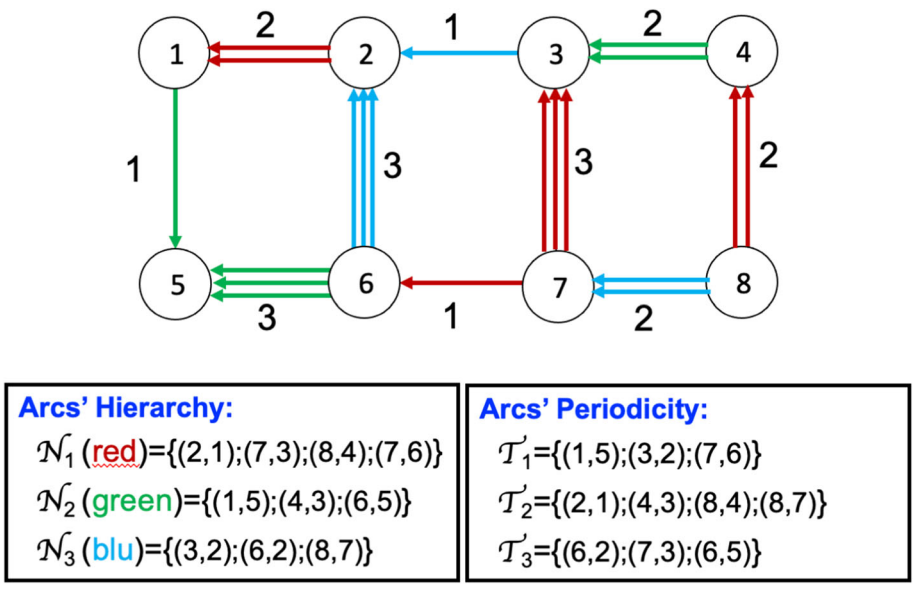

Fig. 1 Illustrative example of the P-HCPP

papers proposed mathematical mixed integer formulations and develop heuristic/metaheuristic approaches for its solution. Unfortunately, the literature does not provide any updated survey on the HCPP, but insightful details on some of the above works have been given in the review of Corberán and Prins (2010).

The second stream of research is related to the introduction of the periodicity aspect in the context of the arc routing. Despite their practical importance in representing real-life applications, the interest in modeling and solving periodic arc/edge routing problems has increased only during the last decade. The first paper in this direction is due to Ghiani et al. (2005) who focused on the periodic rural postman problem and developed a heuristic approach for its solution. Concurrently, Chu et al. (2005) developed an integer formulation for the periodic capacitated arc routing problem and proposed heuristic approaches for its solution. During the following years, a particular attention has been devoted to the introduction of several novel variants of the periodic arc routing problems in order to integrate different modeling features, such as considering the irregularity of the arcs service (Monroy et al. 2013), incorporating decisions related to inventory (Riquelme-Rodríguez et al., 2014a and 2014b), including the service facilities location along the arcs (Huang and Lin 2014; Riquelme-Rodríguez et al., 2016), considering operational time restrictions (Tirkolaee et al. 2018; Thomaz et al. 2018) and more recently involving the multi-objective aspect into the optimization model (Tirkolaee et al. 2019).

Considering the solution strategies, most of the above-mentioned studies have developed heuristic approaches to solve the particular variant of the periodic arc routing problem under exam. Population-based metaheuristics have also been adopted and have shown to be very effective in solving such problems (Lacomme et al. 2005; Chu et al. 2006; Mei et al. 2011; Zhang et al. 2017; and Batista et al. 2019). Moreover, decomposition approaches that exploit the problem's structure to reduce its complexity have been suggested by (Triki 2017), Chen and Hao (2018) and Oliveira and Scarpin (2020). Finally, an attempt to tackle the periodic arc routing problem through exact methods has appeared only recently in the work of Benavent et al. (2019).

The above literature review clearly highlights that, even though several variants of the arc/edge routing problem that incorporate periodicity restrictions have been modeled and solved, none among the available papers has ever dealt with the P-HCPP, despite its importance in modeling and solving several real-life routing problems.

The paper will be structured as follows. Section 2 will be devoted to formally defining the problem in terms of graph theory and suggesting a novel optimization model for the P-HCPP. Section 3 will describe our solution approach, and Sect. 4 will summarize our computational experiments that validate the model and assess the performance of our solution approach. Finally, Sect. 5 will draw some concluding remarks and suggest future avenues of further investigation on the topic.

2 Optimization model

In this section, we first describe the sets, parameters, and variables that are used in the formulation. Then the mathematical formulation of the P-HCPP is provided.

2.1 Sets, parameters and variables

We assume that the set of nodes is denoted by N and the arcs connecting the nodes are characterized by a set, which we call A. Indices i and j are used to represent the nodes, and an arc from set A connecting i and j is represented by (i,j). Set of of patterns that comply with the periodicity requirements of arc (i,j) is called as K_{ij}. Note that a pattern is a sequence of days in which value 1 appears for the day that is included in the pattern, and value 0 appears, otherwise. For instance, in the 30 day sequence: "1 0 1 0 … 1 0" this pattern includes days 1, 3, …, 29 and does not include days 2, 4, …, 30. A similar pattern in which the included days alternates in every other day would be "0 1 0 1 … 0 1". Note that these two patterns would be suitable for the arcs that have periodicity frequency equals to 2. We use index r for pointing the periodicity patterns. The set of hierarchies is given by H and we refer to a specific hierarchy by using index $h \in H$. Set of arcs that belong to hierarchy h is given by A_h. Finally, S stands for the set of steps or periods and T represents the set of days in the planning horizon, while s and m are the indices used to refer to specific steps from set S and index t is used to point out a specific day from set T.

There are 4 parameters used in the formulation and 3 of them depend on the above-described sets and indices. c_{ij}^1 and c_{ij}^2 respectively denote the service and traversing costs of arc (i,j). Note that serving an arc is a more time-consuming task than just traversing it without service implying that $c_{ij}^1 > c_{ij}^2$ should hold. Besides, a_{rt} indicates whether or not day t is included in periodicity pattern r. Finally, M stands for an enough large number.

The model has nine sets of decision variables. The continuous variables are $\beta_{ijt}^1, \beta_{ijt}^2, z_{hst}$ and γ_{ht}. β_{ijt}^1 and β_{ijt}^2 simply represent the number of service times and number of traversing times of arc (i,j) on day t, respectively. Variable z_{hst} is the number of arcs belonging to A_h that are served before step s at day t. Finally, γ_{ht} represents the number of arcs belonging to A_h that have to be served on day t. The binary

variables of the model are θ_{hst}, y_{ijst}, α_{ijr}, x^1_{ijst} and x^2_{ijst}. θ_{hst} indicates whether or not all arcs of hierarchy h that are required to be served on day t are served before step s of day t or not. Variable y_{ijst} takes value 1 if arc (i,j) is served before step s of day t, and it is 0 otherwise. Variable α_{ijr} represents the pattern selection variable for arc (i,j). In other words, α_{ijr} should be 1 if pattern r is assigned to arc (i,j), otherwise α_{ijr} should be set to 0. Finally, variables x^1_{ijst} and x^2_{ijst} indicate whether or not arc (i,j) is served or traversed at step s of day t, respectively. A summary of the sets, parameters and variables definitions is provided in Table 1, for the sake of convenience.

2.2 Mathematical model

Our mixed integer linear program, abbreviated as P-HCPP, is given below.

We minimize in the objective function (1) the total service and traversing cost. Constraint (2) and constraint (3) ensure that, at each day, the vehicle that leaves the first node at step 1 (which is achieved by constraint (2)), should go back to the same node at the last step $|S|$ (guaranteed by constraint (3)). Note that $|S|$ is an upper limit value for the number of steps that can be considered in any day (we selected its value as twice the number of arcs). In order to avoid forcing the vehicle to perform exactly $|S|$ steps on each day, we added a dummy arc that connects the starting node to itself and that has both the service and traversal cost equal to 0. Thus, if on any day the vehicle needs to return to the starting point with fewer steps than $|S|$, it can return to the starting node at the end of step $|S|$ by traversing the dummy arc at 0 cost during the remaining steps. This ensures the satisfaction of constraint (3) for that

$$\min \quad \sum_{(i,j)\in A, i<j} \sum_{t\in T} \left(c^1_{ij}\beta^1_{ijt} + c^2_{ij}\beta^2_{ijt} \right) \tag{1}$$

$s.t.$

$$\sum_{j:(1,j)\in A} \left(x^1_{1j1t} + x^2_{1j1t} \right) = 1 \qquad t \in T \tag{2}$$

$$\sum_{j:(j,1)\in A} \left(x^1_{j1|S|t} + x^2_{j1|S|t} \right) = 1 \qquad t \in T \tag{3}$$

$$\sum_{(i,j)\in A} \left(x^1_{ijst} + x^2_{ijst} \right) = 1 \qquad s \in S, t \in T \tag{4}$$

$$\sum_{j:(j,i)\in A} \left(x^1_{jist} + x^2_{jist} \right) - \sum_{j:(i,j)\in N} \left(x^1_{ij(s+1)t} + x^2_{ij(s+1)t} \right) = 0 \qquad i \in N, s \in S\backslash\{|S|\}, t \in T \tag{5}$$

$$\sum_{s\in S} \left(x^1_{ijst} + x^1_{jist} \right) = \beta^1_{ijt} \qquad (i,j) \in A, i<j, t \in T \tag{6}$$

$$\sum_{s\in S} \left(x^2_{ijst} + x^2_{jist} \right) = \beta^2_{ijt} \qquad (i,j) \in A, i<j, t \in T \tag{7}$$

$$\sum_{r\in K_{ij}} \alpha_{ijr} = 1 \qquad (i,j) \in A, i<j \tag{8}$$

$$\beta^1_{ijt} \geq \sum_{r\in K_{ij}} a_{rt}\alpha_{ijr} \qquad (i,j) \in A, i<j, t \in T \tag{9}$$

$$\beta^1_{ijt} \leq M \sum_{r\in K_{ij}} a_{rt}\alpha_{ijr} \qquad (i,j) \in A, i<j, t \in T \tag{10}$$

$$\sum_{m=1}^{s} \left(x^1_{ijmt} + x^1_{jimt} \right) \leq s y_{ijst} \qquad (i,j) \in A, i<j, s \in T, t \in T \tag{11}$$

$$y_{ijst} \leq \sum_{m=1}^{s} \left(x^1_{ijmt} + x^1_{jimt} \right) \qquad (i,j) \in A, i<j, s \in S, t \in T \tag{12}$$

$$\sum_{(i,j)\in A_h, i<j} y_{ijst} = z_{hst} \qquad h \in H, s \in S, t \in T \tag{13}$$

$$\sum_{(i,j)\in A_h, i<j} \sum_{r\in K_{ij}} a_{rt}\alpha_{ijr} = \gamma_{ht} \qquad h \in H, t \in T \tag{14}$$

$$\gamma_{ht}\theta_{hst} \leq z_{hst} \qquad h \in H, s \in S, t \in T \tag{15}$$

$$\sum_{(i,j)\in A_{h+1}, i<j} y_{ijst} \leq M\theta_{hst} \qquad h \in H, s \in S, t \in T \tag{16}$$

$$\theta_{hst}, x^1_{ijst}, x^2_{ijst}, y_{ijst}, \alpha_{ijr} \in \{0,1\} \qquad (i,j) \in A, h \in H, s \in S, t \in T \tag{17}$$

$$\beta^1_{ijt}, \beta^2_{ijt}, z_{hst}, \gamma_{ht} \geq 0 \qquad (i,j) \in A, h \in H, s \in S, t \in T \tag{18}$$

Table 1 Sets, indices, parameters and decision variables of the P-HCPP formulation

Set/Indices	Definition
$N/i,j$	Set of nodes/Node indices
$A/(i,j)$	Set of arcs/Arc index
K_{ij}/r	Set of patterns representing periodicity of arc (i,j)/Pattern index
H/h	Set of hierarchies/Hierarchy index
A_h	Set of arcs belonging to hierarchy h
$S/s,m$	Set of steps/Step indices
T/t	Set of days/Day index

Parameters	Definition
c^1_{ij}	Service cost of arc (i,j)
c^2_{ij}	Traversing cost of arc (i,j)
a_{rt}	Indicates whether or not pattern r includes day t
M	An enough big number

Variables	Definition
β^1_{ijt}	Number of service times of arc (i,j) on day t
β^2_{ijt}	Number of traversing times of arc (i,j) on day t
z_{hst}	Number of arcs belonging to A_h served before step s on day t
γ_{ht}	Number of arcs of hierarchy h assigned today t
θ_{hst}	Indicates whether or not all arcs belonging to A_h (assigned to day t) are served before step s on day t
y_{ijst}	Indicates whether or not arc (i, j) is served before step s on day t
α_{ijr}	Indicates whether or not pattern r is selected for arc (i,j)
x^1_{ijst}	Indicates whether or not arc (i,j) (through from i to j) is served at step s on day t
x^2_{ijst}	Indicates whether or not arc (i,j) (through from i to j) is traversed at step s on day t

day. Constraint (4) guarantees that the vehicle travels through a single arc at each step of each day. Constraint (5) ensures that the vehicle leaves the node at step $s+1$ if it entered to it at the previous step s, and it cannot leave a node at step $s+1$ if it did not enter it at the previous step s. Hence, this constraint guarantees the continuity of the route performed by the vehicle. Constraints (6) and (7) define β^1_{ijt} and β^2_{ijt} variables. At the left-hand side of the constraints, we sum up the service and traversing variables (x^1_{ijst} and x^2_{ijst}) throughout the steps and let it be equal to β^1_{ijt} and β^2_{ijt} for each day. Constraint (8) ensures that a suitable pattern is assigned for each arc. Constraint (9) is written for each arc and for each day and guarantees that each arc is served at least once independent from the direction of the passing if the arc is assigned to a pattern that requires its service for the day the constraint is written for. Similarly, constraint (10) guarantees that no arc is served on a day if the pattern assignment does not require the arc to be served for the specific day the constraint is written for. Constraints (11) and (12) define the relationship between variables x^1_{ijst} and y_{ijst}. Namely, if the number of services for an arc is zero at a particular step, i.e., $\sum_{m=0}^{s}\left(x^1_{ijmt}+x^1_{jimt}\right)=0$, then $y_{ijst}=0$ should follow. On the contrary, if $y_{ijst}=1$, then $\sum_{m=1}^{s}\left(x^1_{ijmt}+x^1_{jimt}\right)>0$ must hold. Constraint (13) computes the value of variable z_{hst} by counting the number of arcs belonging to hierarchy h that traversed before step s. Constraint (14) defines the γ_{ht} variable. It sums up the arcs that are assigned to pattern that requires the arc to be served for the day the constraint is written for and assigns the summation as the value of the γ_{ht} variable. On the other hand, constraint (15) defines variable θ_{hst}, i.e., it guarantees that θ_{hst} variable is set equal to 0 if all the arcs belonging to hierarchy h that are required to be served on a specific day are not traversed before or during step s. Namely, constraint (15) ensures that θ_{hst} can be equal to 1 only if z_{hst} is equal to γ_{ht}, and it is 0 otherwise. Similarly, constraint (16) avoids serving through arcs belonging to hierarchy $h+1$ if all the arcs of hierarchy h that are required to be served on the day are not served yet. Both the constraints (15) and (16) are written for each step of each day and for each hierarchy. These constraints impose the hierarchy

restrictions. Finally, constraint (17) and constraint (18) represent the usual binary and non-negativity restrictions on the decision variables.

As can be seen, the binary variable θ_{hst} is multiplied with the continuous variable γ_{ht} in constraint (15) which introduces a nonlinearity in the model. In order to linearize the multiplication term, we define a new continuous variable μ_{htl} to replace the multiplication term, i.e., $\mu_{hst} = \gamma_{ht}\theta_{hst}$. We also introduce four new constraint sets, which are linear and force μ_{htl} to be equal to $\gamma_{ht}\theta_{hst}$. These constraints are given in the following.

$$\mu_{hst} \leq \gamma_{ht} \qquad h \in H, s \in S, t \in T \quad (19)$$
$$\mu_{hst} \leq M\theta_{hst} \qquad h \in H, s \in S, t \in T \quad (20)$$
$$\mu_{hst} \geq \gamma_{ht} - M(1 - \theta_{hst}) \qquad h \in H, s \in S, t \in T \quad (21)$$
$$\mu_{hst} \geq 0 \qquad h \in H, s \in S, t \in T \quad (22)$$

As can be understood, if θ_{hst} takes value 1, then μ_{hst} is set equal to γ_{ht} by means of constraints (19) and (21). Similarly, if θ_{hst} is zero, μ_{hst} is set to zero by means of constraints (20) and (22). This implies that the nonlinear multiplication $\gamma_{ht}\theta_{hst}$ existing in constraint (15) can be replaced with μ_{hst} after adding constraints (19–22) to the model.

Since P-HCPP is characterized by a high level of complexity, the computational time that its exact solution may require can be prohibitively large for moderate and large size instances. It is well known that general mixed integer linear programming problems belong to NP-hard problem class, and P-HCPP is no exception. Hence, there is almost no hope for finding a solution method that exactly solves P-HCPP instances in polynomial time. (Interested readers are directed to seminal work by Wolsey (1998)). This observation underlines the need to resort to a heuristic solution procedure for the solution of relatively larger instances of P-HCPP. We propose in the sequel a novel hybrid heuristic solution in which we integrate a famous simulated annealing metaheuristic together with an adaptation of the Dror et al.'s layer algorithm in a nested manner.

3 Solution procedure

The difficulty of solving moderate and large P-HCPP instances in tolerable computation times is due to the large number of binary variables that our formulation of P-HCPP includes. Hence, one must attempt to reduce the number of binary decision variables in order to efficiently solve P-HCPP instances in a relatively acceptable amount of time. An observation which may help in reducing the number of binary variables can be inspired from the nature of the pattern selection variables, i.e., α_{ijr}. One may see that if the values of α_{ijr} variables are given a priori beforehand as a parameter, then solving the whole problem would reduce to solving an HCPP instance for each day independently. We highlight that solving HCPP instance for each day independently and aggregating their solutions to obtain a solution for the original P-HCPP is naturally easier and takes lesser time then solving the P-HCPP instance at once altogether even if the values of α_{ijr} variables are fixed. We take advantage of such observation, and we propose a 2-phase Hybrid Heuristic (HH) for the solution of P-HCPP instances. At the outer phase of the HH, we search for the values of α_{ijr} variables by a Simulated Annealing (SA) algorithm. In the inner phase, we solve an instance of HCPP for each day independently, while using the α_{ijr} values identified by the SA at the outer phase, and we then aggregate the partial solutions to provide a solution for the original P-HCPP instance. Our choice of implementing the SA algorithm is based on two different reasons. First, the SA is a metaheuristic that involves a limited number of parameters and is relatively easy to code to tune to validate. The second reason is the fact that it is necessary to execute the layer algorithm to evaluate each solution, which needs relatively a long computational time. The SA algorithm, in which only one solution is evaluated at each step, results to be more advantageous empirically with respect to the more modern metaheuristics (such as tabu search, ILS or ALNS) that require the evaluation of a large number of solutions at each step, and also with respect to the genetic algorithm that requires the evaluation of a large number of solutions to form the initial population.

3.1 Hybrid heuristic

We find unnecessary to give the theoretical foundations of the famous simulated annealing algorithm here (interested reader can refer to Delahaye (2019)). A sketch of our SA implementation is given in the following. Note that by the phrase 'solution' we mean an assignment of each $(i,j) \in A$ to the one of the suitable periodicity patterns.

1. Initialization: Determine the initial solution (the values of α_{ijr} variables are set by assigning each arc $(i,j) \in A$ to the first suitable pattern), set $iterlim = 1000$, $nonopt = 0, nonopt\_limit = 10$, $iterlength = 10$, $T_0 = -\frac{s_C}{\ln(0.9)}$ where s_C is the standard deviation of 30 randomly selected solutions, $coef = 0.99$, and $k = 0$. Let the current solution be represented by α_k and a neighboring solution brepresented by α'.
2. While ile (time limit is not reached *and* $k < iterlim$ *and* $nonopt < nonopt\_limit$
   a. $nonopt = nonopt + 1$.
   b. for $(iter = 1 : iterlength)$

i. Obtain a neighboring solution α'. and let $f(\alpha')$ note the length of the associated tour and let $\Delta = f(\alpha') - f(\alpha_k)$
ii. Let $\alpha_{k+1} = \alpha'$ if $\Delta < 0$ or $R(0,1) < e^{\frac{-\Delta}{T_k}}$ where $R(0,1)$ denotes a random number generated within the (0, 1) interval, else $\alpha_{k+1} = \alpha_k$
iii. If α' is better than the best solution $nonopt = 0$

c. $T_{k+1} = T_k \times coef$
d. $k = k + 1$

The initial temperature selection (i.e., $T_0 = -\frac{s_C}{\ln(0.9)}$) makes it possible to accept, at the initial iterations of the algorithm, the neighboring solution with approximately 90% probability even if it is worse than the current solution. Note that as the number of iterations grows, the value of the temperature reduces (due to $T_{k+1} = T_k \times coef$) and the probability of accepting worse solutions decreases and converges to 0 toward the end of the algorithm. In order to generate a neighboring solution with 50% probability, we randomly select a single arc and change its current pattern to one of the other suitable patterns, and with 50% probability, we randomly select two arcs and change each of their assigned patterns. Finally, *nonopt* counts the number of consecutive iterations in which the best solution does not improve and if it reaches to *nonopt_limit*, which is 10, the algorithm terminates. We referred to the work by Reeves (1993) during the selection of initial temperature and selection of other parameter values.

Once the periodicity pattern is assigned to each arc, one has to solve an HCPP instance for each day of the planning horizon in order to obtain the tour length $f(\alpha')$ for a candidate neighboring solution α'. Smaller instances of HCPP can be optimally solved by commercial mixed-integer linear programming solvers like Gurobi (2021). However, once again we need here practical heuristic solution strategies to tackle moderate and large sized HCPP instances. Luckily, there are, as mentioned in the first section, several good heuristic strategies developed in the literature for the solution of HCPP instances. We adapt here the Dror et al.'s layer algorithm.

Layer algorithm includes a polynomial number of operations and optimally solves HCPP with linear precedence relationships. If the arcs belonging to each hierarchy are connected and if there is, for each hierarchy (except the highest hierarchy), at least one incident node of the hierarchy that is also incident to a higher hierarchy, then the instance is said to have linear precedence relationships. All instances considered in this study are constructed so that they have linear precedence relationships. However, since the periodic service requirements of the arcs depend on the selected pattern, i.e., the value of α_{ijr} variables, each arc does not have to be passed each day. Hence, for a given day, we have to solve an HCPP instance in which we are to find a tour that serves a subset of arcs the selected pattern dictates. Those arcs that waits to be serviced for a given day do not necessarily possess a linear precedence relationship even if the original network has that property. Therefore, the HCPP instances to be solved for each day may not have the linear precedence relationships. Moreover, we allow passage through arcs without service probably for a lesser cost than the service cost, which makes our problem even more different. Namely, service order must follow the hierarchies while passage without service is made when needed without following the hierarchies. The layer algorithm does not guarantee optimality for our HCPP instances due to the above-mentioned reasons, but it is still able to provide a probably sub-optimal feasible solution implying that it can be used as a heuristic procedure.

The famous blossom method of Edmonds (1965) is run as a sub-procedure of the layer algorithm to find the shortest path between two nodes of the graph. However, we do not require service costs to be symmetric which makes blossom method inapplicable for our case. Instead, we construct an integer programming formulation and use it as a sub-procedure within the layer algorithm in order to find the shortest paths between any two nodes. This method shares some similarity with the Windy Layer Algorithm (WLA) of Keskin et al. (2021). However, our integer formulations are different as our cost structure does not have their variable cost property and also we allow here traversing arcs without service, contrary to the approach followed by Keskin et al. We go over the details in the sequel.

We name the subgraph induced by the set of arcs of hierarchy h as $G_h = (N_h, A_h)$ where N_h is the incident nodes of A_h. We also define F_q as the union of the first q sets, i.e., $F_q = \bigcup_{h=1}^{q} G_h$. Now, if node $i \in N_h$ is incident to an arc of F_{h-1}, it is defined as an entry node of G_h. Moreover, we define Y_h to represent the set of entry nodes of G_h. Mathematically, Y_h is equal to $\left(\bigcup_{q=1}^{h-1} N_q\right) \bigcap N_h$. Finally, we define Y_1 and $Y_{|H|+1}$ both equal to $\{i_1\}$ where i_1 represents the depot where the tour starts and ends. Now, a new graph, say $G' = (N', A')$, with node set $N' = \bigcup_{h=1}^{|H|+1} Y_h$ and arc set $A' = \{(i,j) : i \in Y_h, j \in Y_{h+1} \text{ for } h = 1, \ldots, |H|\}$ is defined. It should be noted that a node may exist in more than one of the sets $Y_h, h = 1, \ldots, |H| + 1$. Namely, a node can be an entry node for more than one hierarchy. In this case, the node existing in more than one entry node set should be treated as a different node each time. The resulting bipartite graph G' is illustrated in Fig. 2.

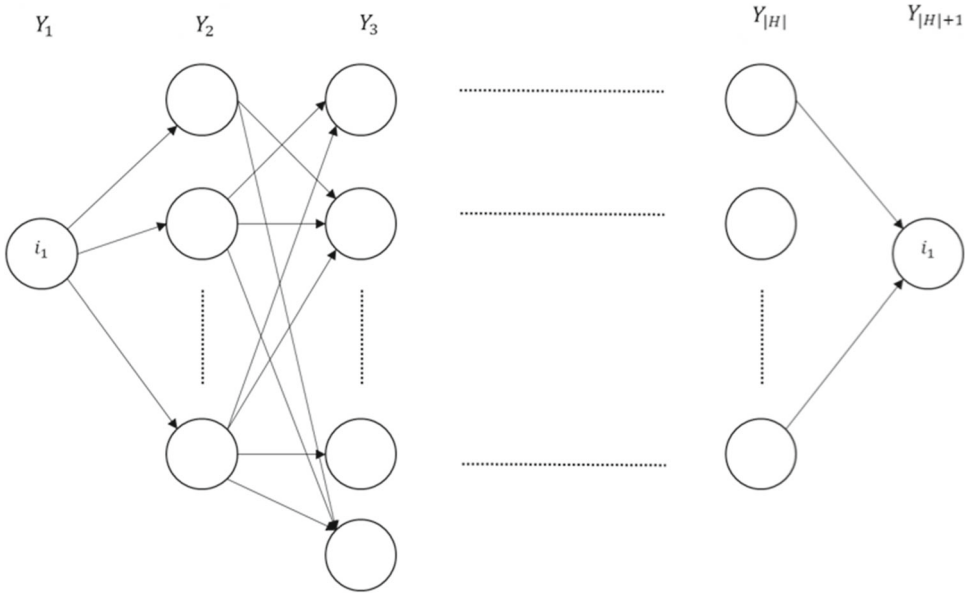

Fig. 2 The $(k+1)$ bipartite graph G' (Taken from Dror et al. 1987)

We remind here that we are to solve an HCPP for each day, and we do not have to pass through each arc every day but only the arcs required by the values of the pattern selection variables α_{ijr} that result from the outer SA algorithm. Going from i to j in G' through an arc $(i,j) \in A'$ such that $i \in Y_h$ and $j \in Y_{h+1}$ for some $h = 1, \ldots, |H|$ actually means that we start the route from i and end with j and in the meantime, each arc belonging to A_h which is assigned to that specific day (according to α_{ijr} values) should be served with minimum possible cost. Moreover, we should make sure not serving any arc from lower hierarchy classes (but traversing through an arc without service is permitted when needed) and the arc cost belonging to $(i,j) \in A'$ is taken as the length of the above-mentioned route. The length of the route is calculated by the help of an integer programming formulation.

3.2 Shortest path identification within HH

Let $P(i,j,h,t)$ denotes the problem of finding the shortest path starting from node i, ending at node j that serves every arc of A_h which are assigned to day t without serving any arc from lower hierarchies. We define x^1_{uv} and x^2_{uv} as binary variables indicating, respectively, whether or not arc (u,v) is served through the direction from u to v and traversed without service through the direction from u to v, in the solution of $P(i,j,h,t)$. We report the formulation of the problem $P(i,j,h,t)$ with these new set of variables in the following.

$$\min \quad \sum_{(u,v)\in A} \left(c^1_{uv}x^1_{uv} + c^2_{uv}x^2_{uv}\right) \tag{23}$$

s.t.

$$x^1_{uv} + x^1_{vu} \geq \sum_{r\in K_{uv}} a_{rt}\alpha_{ijr} \qquad (u,v) \in A_h, u < v \tag{24}$$

$$\sum_{v:(u,v)\in A} \left(x^1_{uv} + x^2_{uv}\right) - \sum_{v:(v,u)\in A} \left(x^1_{vu} + x^2_{vu}\right) = 0 \qquad u \in V: \quad u \neq i,j \tag{25}$$

$$\sum_{v:(i,v)\in A} \left(x^1_{iv} + x^2_{iv}\right) - \sum_{v:(v,i)\in A} \left(x^1_{vi} + x^2_{vi}\right) = 1 \tag{26}$$

$$\sum_{v:(j,v)\in A} \left(x^1_{jv} + x^2_{jv}\right) - \sum_{v:(v,j)\in A} \left(x^1_{vj} + x^2_{vj}\right) = -1 \tag{27}$$

$$x^1_{uv} \leq 0 \qquad (u,v) \in A_{h'}, h' = h+1, \ldots, |H| \tag{28}$$

$$x^1_{uv}, x^2_{uv} \in \{0,1\} \qquad (u,v) \in A \tag{29}$$

$P(i, j, h, t)$ minimizes the total service and traversal cost through expression (23). Note that c^1_{uv} represents the service cost of arc $(u,v) \in A$ while c^2_{uv} represents the traversal

(without service) cost of arc $(u, v) \in A$. By constraint (24) we make sure that each arc from A_h assigned to day t by the selection of α_{ijr} values coming from the outer SA algorithm is served at least once regardless of the direction. Note that α_{ijr} are not decision variables but rather act as parameters since their values are already known and fixed by SA beforehand. Constraints (25–27) are flow balance constraints. More specifically, for a node different from i and j, the number of ingoing arcs should be equal to the number of outgoing arcs and that is stated by constraint (25). As the tour starts from node i for $P(i,j,h,t)$, the number of ingoing arcs to node i should be one less than the number of outgoing arcs from node i which is achieved by constraint (26). On the other hand, as the tour ends at node j for $P(i,j,h,t)$, the total number of ingoing arcs should be one more than the total number of outgoing arcs for node j and this requirement is satisfied by the help of constraint (27). Constraint (28) avoids serving arcs belonging to lower-level hierarchies while permitting traversing without service. Finally, defining x^1_{uv} and x^2_{uv} as binary variables is ensured by constraint (29).

Arc costs of $G' = (N', A')$ can be calculated by solving $P(i,j,h,t)$ for each $(i,j) \in A'$. Then, an algorithm like Dijkstra's label algorithm (Dijkstra 1959) can be incorporated to find the shortest path from $i_1 \in Y_1$ to $i_1 \in Y_{|H|+1}$. The shortest path from $i_1 \in Y_1$ to $i_1 \in Y_{|H|+1}$ gives the solution that we seek for day t. Lengths of the tours found for each day $t \in T$ are then summed up to find the total cost associated for a given α, namely, $f(\alpha)$.

3.3 Illustrative example

We now illustrate the layer algorithm on the small example shown in Fig. 3 and we will highlight the steps to identify a sub-optimal solution (not necessarily the optimal even for a 6 day problem).

Suppose the first hierarchy consists of the red arcs including (1, 2) and (2, 4), and they have to be visited every day. The second hierarchy includes the green arcs (2, 3) and (2, 5), and their periodicity is two days implying that they are to be visited once in every two days. Finally, blue arcs (1, 4) and (4, 5) form the third hierarchy, and they have to be visited once in every three days. Service costs are written near each arc, and traversal costs of each arc without service is considered equal to one fifth of its service cost. Suppose there are 6 days in the planning horizon. There are 6 different possible patterns: the first pattern requires visiting the arcs every day, the second and the third patterns require the arcs to be visited once in every two days and the fourth, fifth and sixth patterns requires the arcs to be visited once in every three days. Hence, the first pattern is suitable for the first hierarchy, the second and third patterns are suitable for the second hierarchy, and the remaining patterns are suitable for the third hierarchy arcs. Relationship between patterns and days of the planning horizon is captured through a_{rl} parameter, as reported below.

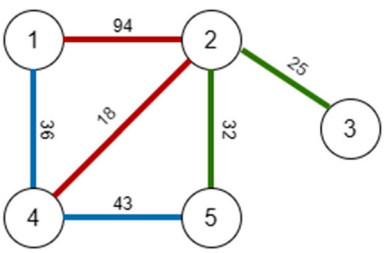

Fig. 3 A toy example for layer algorithm

$$a_{rl} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

We coded the mathematical model of P-HCPP in C# and run the commercial solver Gurobi for this toy example to obtain the following optimal solution. Arcs (1, 2) and (2, 4) are assigned to the first pattern, arcs (2, 3) and (2, 5) are assigned to the second pattern, and finally, the arcs (1, 4) and (4, 5) are assigned to the fourth pattern. That is, all arcs are assigned to the first possible pattern in the optimal solution. Hence, α_{121}, α_{241}, α_{232}, α_{252}, α_{144}, α_{454} values are set to 1 while all the other α_{ijr} values are set to 0 in the optimal solution. The tours and the associated costs obtained in the optimal solution for each day are such as:

Day 1 Tour: 1–2–4–2–3–2–5–4–1; Cost=94+18+3.6+25+5+32+43+36=256.6
Day 2 Tour: 1–2–4–1; Cost=94+18+7.2=119.2
Day 3 Tour: 1–2–4–2–3–2–5–4–1; Cost=94+18+3.6+25+5+32+8.6+7.2=193.4
Day 4 Tour: 1–2–4–5–4–1; Cost=94+18+43+8.6+36=199.6
Day 5 Tour: 1–2–4–2–3–2–5–4–1; Cost=94+18+3.6+25+5+32+8.6+7.2=193.4
Day 6 Tour: 1–2–4–1; Cost=94+18+7.2=119.2

One may notice from the optimal solution that services are made following the hierarchy order and traversal costs without service are taken as one fifth of the service costs. The total cost of the optimal solution for 6 days is 1081.4. Now, let's elaborate the solution that the layer algorithm can find for this toy example. First of all, the outer SA algorithm is initiated so that every arc is assigned to the first suitable pattern, which is the optimal solution as identified by Gurobi, and reported previously. Hence, the

values of the α_{ijr} variables obtained from the SA algorithm is the optimal α_{ijr} values. However, the tours obtained from the layer algorithm still deviates from the optimal tours given above.

First of all, we define Y_1 and Y_4 to be equal to $\{i_1 = 1\}$. We also define Y_2, which is the set of entry nodes of hierarchy 2, which only consists of node 2 since it is the only node that is incident to the arcs of hierarchy 2 and hierarchy 1. Moreover, Y_3. consists of nodes 1, 4 and 5 since these nodes are incident to the arcs of hierarchy 3 and to at least one of the higher hierarchies which are hierarchy 1 and 2. Now, the sketch of the $G' = (N', A')$ constructed for day 1 with node set $A' = \bigcup_{h=1}^{4} Y_h$ and arc set $A' = \{(i,j) : i \in Y_h, j \in Y_{h+1} \text{ for } h = 1, \ldots, 3\}$ is given in Fig. 4.

The numbers written near to each arc $(i,j) \in A'$ are the lengths of the tours that serves all the arcs of the related hierarchy starting from node i and ending at node j. For instance, 115.6 is the length of the route on the original network that starts from node 1 that serves arcs of hierarchy 1 and ends at node 2. That route can be written as 1–2–4–2 with total cost 115.6 = 94 + 18 + 3.6. Hence, the arc $(1,2)$ in graph G' represents the tour original graph G.

Similarly, 77.8 is the length of the route serving arcs of hierarchy 2 that starts from node 2 and ends at node 1. The route is 2–3–2–4–5–1 and its total cost can be calculated as 77.8 = 25 + 5 + 32 + 8.6 + 7.2. Here, the arc $(2,1)$ in graph G' represents the route 2–3–2–4–5–1 in the original graph G. The other costs can be calculated in a similar manner. The total cost of day 1 is equal to the *length of the shortest path* from the first node of G', which is node 1, to the last node of G', which is again node 1, and it is equal to 256.6 = 115.6 + 62 + 79. Note that this value is equal to the day 1 solution value of the optimal solution. However, on the other days, we do not have to serve all arcs while layer algorithm always incorporates entry nodes assuming that each hierarchy will be served, and that causes layer algorithm to deviate from the optimal solution. For instance, we re-construct the $G' = (N', A')$ for day 4, as shown in Fig. 5, during which only arcs of hierarchy 1 and 3 are to be served.

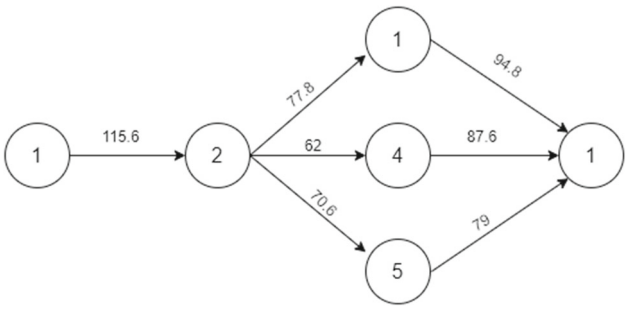

Fig. 4 Graph $G' = (N', A')$ corresponding to day 1 for the toy example

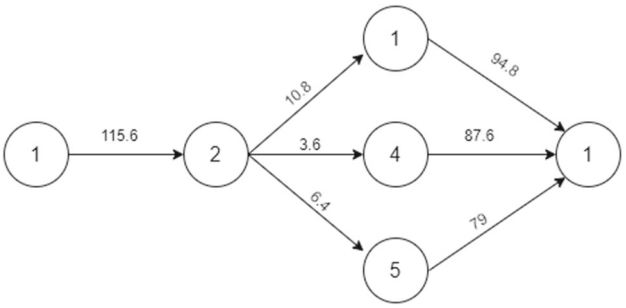

Fig. 5 Graph $G' = (N', A')$ corresponding to day 4 for the toy example

Note that we do not have to serve arcs of hierarchy 2 on day 4 (since arcs of hierarchy 2 are served on days 1, 3 and 5 due to the assigned periodicity pattern) which reduces the costs of the routes starting from node 2 in the second phase of the method. To illustrate how we calculate the costs let's focus on the route starting from node 2 and ending at node 1. The route is 2–4–1, and its cost is only 10.8 = 3.6 + 7.2 as the only aim is to head to node 1 without having to serve the arcs of hierarchy 2. Total cost of day 4 is the length of the shortest path from the beginning node to the ending node in G', and it is equal to 201 which is slightly larger than the day 4 value of the optimal solution which is 199.6. Besides, the total costs produced by the layer algorithm for days 2 and 6 are both 126.4 which is slightly larger than the costs of the optimal solution for the same days. In total, the cost of the solution for all the 6 days that layer algorithm finds is 1097.2 while the optimal objective function value is 1081.4. This is a clear indication of the fact that the layer algorithm does not necessarily produce optimal solution for our instances. However, it is much faster than the commercial solver and is able to produce good quality solutions as the instance size gets larger, as we illustrate in the next section.

4 Experimental results

In this section, we first describe how we generate the test problems for P-HCPP and how the values of the parameters of the P-HCPP are selected. Later, we illustrate the efficiency and the accuracy of the Hybrid Heuristic (HH) on the generated test instances.

4.1 Instance generation and parameter value selection

4 sets of problems with 10, 20, 30 and 40 nodes with 3 different arc densities are generated, and, for each of them, 4 different number of hierarchies are assumed which produces 48 different P-HCPP instances. Three different

number of arcs are calculated as $\left\lceil \frac{n\times(n-1)}{3} \right\rceil$, $\left\lceil \frac{n\times(n-1)}{5} \right\rceil$, $\left\lceil \frac{n\times(n-1)}{7} \right\rceil$, respectively, where n stands for the number of nodes. For instance, there are three instances with $\left\lceil \frac{10\times9}{3} \right\rceil = 30$, $\left\lceil \frac{10\times9}{5} \right\rceil = 18$, $\left\lceil \frac{10\times9}{7} \right\rceil = 13$ number of arcs for the instances with 10 nodes. We randomly assign each arc to one of the hierarchies so that arcs of each hierarchy are connected, and the generated instances have the linear precedence property. That is, for each hierarchy, there is at least one incident node of the hierarchy which is also incident to one of the higher hierarchies. We incorporate 4 number of hierarchies as 2, 3, 4 and 5 for each instance.

Service costs of the arcs are randomly generated as $c_{ij}^{1} = 30 + 70 \times rand(0,1)$ for all $(i,j) \in A$ where $rand(0,1)$ is a random real number from the $(0,1)$ interval. This will make the service costs to be random real numbers from the interval $(30,100)$ for all arcs. Besides, traversal costs without service are equal to one fifth of the service costs for each arc.

On the other hand, a period is assigned to each hierarchy along all the planning horizon, chosen to be 30 days. If the period is 3 for a hierarchy, it means that the arcs of that hierarchy must be visited once in every 3 days. Periods are assigned in such a way that higher hierarchies have lower periods, and period can be at most 7 for any hierarchy. After assigning the periods, possible patterns are generated accordingly for each hierarchy, and the values of the a_{rl} parameters are calculated for each pattern, and for each day in the planning horizon.

4.2 Accuracy and efficiency of HH on small instances

In this section, we assess the performance of the HH by comparing the objective function values corresponding to the solutions found by HH and those by the state-of-the-art MILP commercial solver Gurobi (2021) for the generated instances. We use C# language and Visual Studio environment for coding both methods and all the experiments for the generated instances are carried out using a single Intel i5-4570 core. We let Gurobi and HH run for at most 5 h for each of the instances. If Gurobi is able to find the optimal solution or if the HH converges to its best solution before 5 h, then they report the solution found and immediately starts running the next instance without waiting until the end of 5 h. We are aware that 5 h is a long time, and some real implementations would require less computation times. However, we still set such computation time limit for several reasons which are: (i) the resulting solution gives a complete tour for a long planning horizon, which is 30 days, implying that one may choose to bear 5 h computation time only once at the beginning of each month for a better quality solution for the whole month, (ii) the problem is characterized by a high level of complexity, and Gurobi is unable to find even feasible solutions for most of the instances when a computation time limit of less than 5 h is selected. In order to obtain a comparison basis for at least the smallest instances, having 10 nodes, we decide to extend the computation time, (iii) HH runs an SA algorithm at the outer part and is able to converge to its best solution especially for the relatively smaller instances. However, as the size of the instances gets larger, SA begins to terminate before convergence since the solution time of the layer algorithm (that is run at the inner level) requires more and more time. In order to avoid early termination of the SA for relatively larger instances, the limit of 5 h seems very appropriate.

Despite the allotted high amount of computation time, Gurobi does not produce feasible solutions for 7 out of the 12 smallest instances having 10 nodes and for none of the instances with 20 nodes. Hence, we choose not to run Gurobi for the larger instances having 30 and 40 nodes. On the other hand, the HH is able to produce feasible solutions for all instances. Hence, we split the report of the results into two parts. In the first part, we report the instances with 10 and 20 nodes for which both Gurobi and HH methods are run for in Table 2.

We specify the number of nodes and number of arcs in the first and second columns of Table 2. The number of hierarchies are given in the third column. The objective function values, i.e., the total tour length for 30 days, found by Gurobi and HH are respectively given in column 4 and column 5. Finally, the corresponding computation times are respectively given in columns 6 and column 7. As previously stated, Gurobi is unable to produce feasible solutions for 7 instances with 10 nodes, which are instances 10–13–3, 10–18–3, 10–18–4, 10–8–5, 10–30–3, 10–30–4 and 10–30–5 (we use here the nodes-arcs-hierarchies notation). On the other hand, Gurobi is unable to find any feasible solution for instances with 20 nodes. Phrase 'NA' in column 4 of Table 2 means that after the computation time limit, no feasible solution is found by Gurobi, while phrase 'OOM' is the abbreviation for "out of memory" implying that the instance is too large for the computer memory and that the solver immediately halts without reaching the end of the time limit.

The results of Table 2 show that for only two instances, which are 10–13–4 and 10–13–5, Gurobi finds solutions having smaller costs than those of HH. On the contrary, the costs of the solutions identified by HH is better than those of Gurobi for the 3 instances 10–13–2, 10–18–2 and 10–30–2. We summarize the same results for all 10-node instances in Fig. 6 (that highlights once again how HH obtained good quality solutions whereas Gurobi fails to find any feasible solution for 7 instances). As stated before,

Table 2 Performance of Gurobi and the HH for small instances

Node	Arc	Hierarchy	Solution		Time	
			Gurobi	HH	Gurobi	HH
10	13	2	9749.80	7752.00	18,004.87	136.81
		3	NA	17,494.20	18,000.07	157.91
		4	8378.20	9193.00	18,000.29	86.96
		5	14,297.80	14,488.80	18,000.29	231.97
	18	2	31,641.20	31,409.80	18,000.51	100.01
		3	NA	21,706.00	18,000.19	314.78
		4	NA	16,372.20	18,000.21	418.59
		5	NA	25,377.00	18,000.17	222.13
	30	2	39,219.60	17,263.60	18,001.46	243.93
		3	NA	18,757.20	18,000.51	1581.42
		4	NA	36,324.40	18,000.51	1733.24
		5	NA	35,494.40	18,000.56	964.20
20	55	2	NA	31,212.40	18,003.58	735.04
		3	NA	23,092.20	18,003.22	3119.82
		4	NA	66,465.80	18,003.50	5870.41
		5	NA	59,898.40	18,003.17	6882.55
	76	2	OOM	24,552.40	NA	1694.16
		3	OOM	84,324.00	NA	9134.96
		4	OOM	59,189.40	NA	7131.55
		5	OOM	39,900.40	NA	14,148.82
	127	2	OOM	151,994.40	NA	1421.25
		3	OOM	174,778.60	NA	18,055.23
		4	OOM	99,028.00	NA	18,184.72
		5	OOM	145,798.40	NA	18,078.49

NA no feasible solution found, *OOM* out of memory

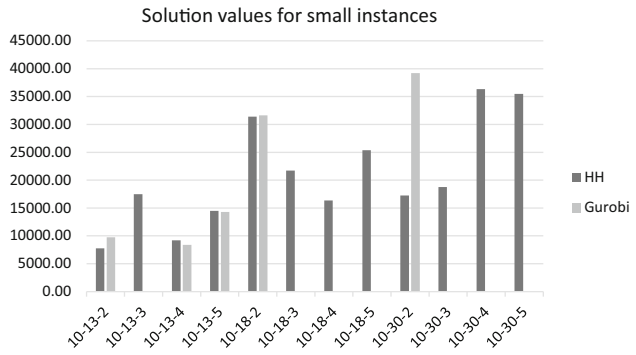

Fig. 6 Solution values for small instances with 10 nodes

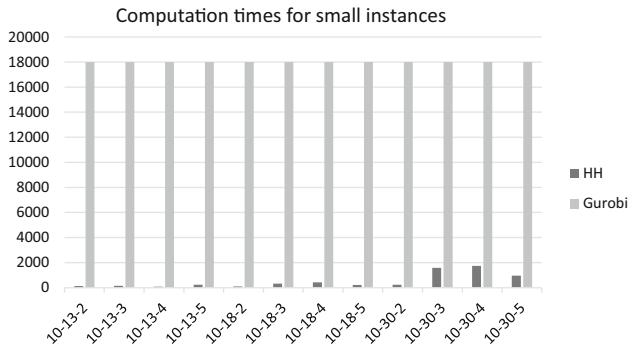

Fig. 7 Computational Times Spent by HH and Gurobi for Small Problems

no feasible solution is provided by Gurobi for all instances with 20 nodes.

Furthermore, one may also observe from Table 2 that the computation time spent by the HH is much smaller than the computation time employed by Gurobi for all 10-node instances. Indeed, Gurobi uses all the allotted time limit, which is 18,000 s, while the HH employs at most 1733.24 s. As the sizes of the instances gets larger (i.e., node numbers reaching 20), the computation time used by HH gets also larger reaching 18,000 s for some of the 20-node instances. On the other hand, Gurobi terminates early for most of the 20-node instances due to the 'out of memory' error. The comparison of the computation times spent by both methods for instances with 10 nodes is also summarized in Fig. 7, whereas the computation times are clearly incomparable for all 20-node instances.

Finally, Fig. 8 presents a summary of solution and time comparison of Gurobi and HH for the small 10-node problems. The figure includes the average costs of the solutions found and the computation times spent by both methods. Figure 8 clearly shows that the computation times of HH are much less than the time spent by Gurobi. On the other hand, the average cost of the solutions found by HH for all 10-node instances is higher than the average cost of Gurobi solutions. This is caused by the fact that HH is able to find solutions for relatively larger 10-node instances having naturally more costs. A fairer comparison can be made by comparing both methods over only the instances for which Gurobi provides feasible solutions. Those instances are named as "Gurobi instances" in the figure, and it is clear that the average cost of the HH solutions over Gurobi instances is less than the average cost of the solutions found by Gurobi. Therefore, it can be claimed that, for those instances, HH is able to find better quality solutions in much smaller amount of time compared to the commercial solver Gurobi.

4.3 Accuracy and efficiency of HH on large instances

The second set of experiments, whose results are reported in Table 3, are devoted to the results of HH for all instances reaching up to 40 nodes, 520 arcs and 5 hierarchies. The results of HH for instances with 10 and 20 nodes are already reported in Table 2, but they are also kept here in order not to harm the integrity of the HH results.

The objective function values of the solutions and the computation times related to HH are reported in columns 4 and 9; and in columns 5 and 10, respectively. These values are also depicted in Figs. 9 and 10, respectively.

One may extract from Table 3 and Figs. 9 and 10 that the cost of the solutions tends to increase as the problem size gets larger and consequently the computation times of HH gets larger and reaches to the allotted 5 h for the relatively larger instances. Moreover, the computation time gets even larger than 5 h for the largest instance having 40 nodes and 520 arcs. Since the stopping condition can be checked at the beginning of each iteration of the HH, and as the inner iterations take relatively large amount of time for large instances, the allotted computation time may already have been surpassed at the check point. If the computation time of an instance exceeds the allotted 5 h substantially, this can be interpreted that the instance size gets too large to be solved by HH in the allowed computation time. This is not surprising, given the high complexity characterizing the problem under exam and also the long-time horizon considered in our instances. Hence, we decided not to test the HH on even larger instances. However, much larger instances, can be decomposed into several sub-networks having each 40 or less nodes that can be solved simultaneously. This can be an interesting line of research that can be investigated in future.

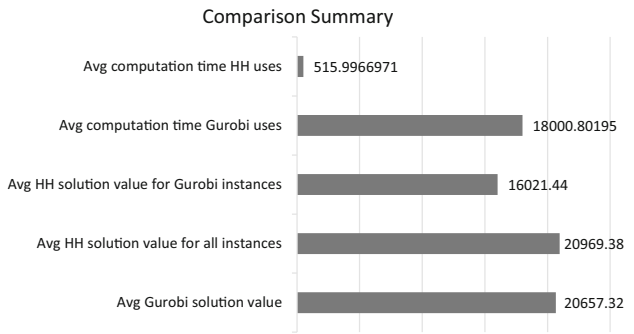

Fig. 8 Comparison of Computational Times and Solution Values of 10-node Instances

5 Concluding remarks

In this study, we developed a mixed integer linear programming formulation of the Periodic Hierarchical Postman Problem which has never been covered in the literature before. Since the developed model is characterized by so many binary variables, commercial solvers, like Gurobi, are not able to solve large-scale problems. Hence, we proposed a Hybrid Heuristic in which a simulated annealing algorithm combined with an adaptation of layer algorithm run in a nested manner. At the outer phase, the simulated annealing identifies the patterns which are more suitable with the periodicity requirements of the arcs. Then the adaptation of the layer algorithm is run for each day and for given pattern decision coming from the outer SA algorithm at the inner phase. In the layer algorithm, instead of employing the blossom algorithm, that finds the minimum length route between two arcs while obeying service requirements of the arcs without violating hierarchy restrictions, we constructed and solved an integer programming model to achieve the same goal. The reason is that the Blossom algorithm does not suit our model settings since our cost structure is not necessarily symmetric, as blossom algorithm requires. Our developed method HH, after being tested on several instances, has shown to perform much better than the commercial solver Gurobi especially for moderate and large sized instances. Moreover, despite that a relatively high computation time was allowed, Gurobi was unable to produce any feasible solution for instances having 20 or more nodes, whereas HH produced good quality solutions for all instances, including even relatively large ones, in a reasonable amount of time, on the average.

Table 3 Performance of the HH for all instances

Node	Arc	Hierarchy	Solution	Time	Node	Arc	Hierarchy	Solution	Time
10	13	2	7752.00	136.81	30	125	2	42,492.60	18,007.33
		3	17,494.20	157.91			3	44,813.60	18,008.01
		4	9193.00	86.96			4	86,396.60	18,058.74
		5	14,488.80	231.97			5	135,462.20	18,058.34
	18	2	31,409.80	100.01		174	2	209,590.00	1785.61
		3	21,706.00	314.78			3	70,878.60	18,121.26
		4	16,372.20	418.59			4	111,718.40	18,212.53
		5	25,377.00	222.13			5	237,047.60	18,048.79
	30	2	17,263.60	243.93		290	2	91,088.00	18,013.48
		3	18,757.20	1581.42			3	131,750.80	18,177.04
		4	36,324.40	1733.24			4	263,492.60	18,030.58
		5	35,494.40	964.20			5	375,338.20	18,029.67
20	55	2	31,212.40	735.04	40	223	2	148,152.60	18,017.70
		3	23,092.20	3119.82			3	241,499.20	18,250.23
		4	66,465.80	5870.41			4	238,127.80	18,243.13
		5	59,898.40	6882.55			5	226,911.40	18,247.49
	76	2	24,552.40	1694.16		312	2	168,189.80	18,037.88
		3	84,324.00	9134.96			3	407,068.00	18,106.85
		4	59,189.40	7131.55			4	292,748.60	18,348.93
		5	39,900.40	14,148.82			5	316,115.00	18,811.36
	127	2	151,994.40	1421.25		520	2	409,259.40	18,028.69
		3	174,778.60	18,055.23			3	269,588.60	18,164.85
		4	99,028.00	18,184.72			4	549,096.40	19,069.27
		5	145,798.40	18,078.49			5	283,819.60	19,186.59

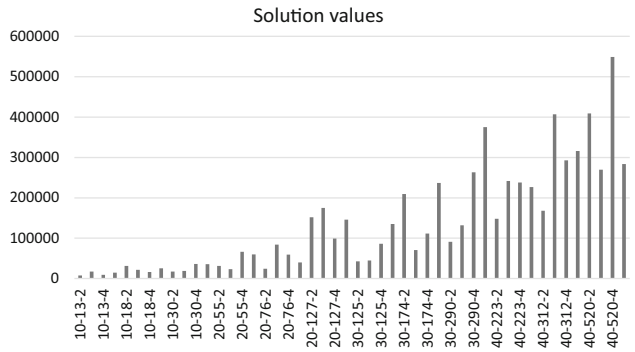

Fig. 9 Objective Function Values for the Solutions Found by the HH

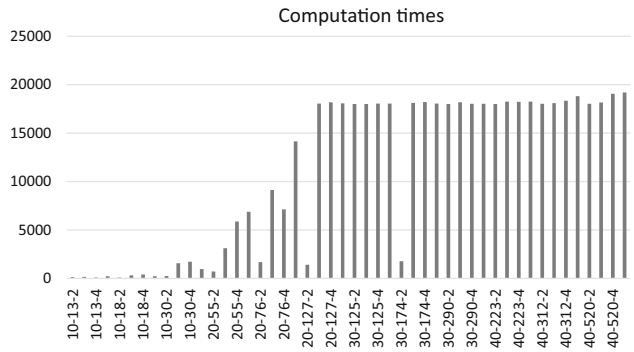

Fig. 10 Computation Times Spent by the HH

This work can be extended along several ways. First, the periodic variant of the windy or rural hierarchical postman problem versions can be defined and analyzed. Another avenue of research consists in studying the fuzzy or the stochastic versions of the problem (see Kaveh et al. 2021) and its dynamic aspect (see Ghiani et al. 2007). Also, another line of study is decomposing large-scale instances that can arise in real-life applications into sub-problems and solving them independently, and then testing the efficiency of such decomposition approach. Finally, an interesting direction of work is to extend the model so that multiple vehicles and/or multiple depots are incorporated into the problem and allowing some arcs to be visited with some delays (Leggieri et al. 2007 and 2010).

Funding Open access funding provided by Università del Salento within the CRUI-CARE Agreement.

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

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

Al-Hinai N, Triki C (2020) A two-level evolutionary algorithm for solving the petrol station replenishment problem with periodicity constraints and service choice. Ann Oper Res 286(1):325–350

Batista GV, Scarpin CT, Pécora JE, Ruiz A (2019) A new ant colony optimization algorithm to solve the periodic capacitated arc routing problem with continuous moves. Math Probl Eng 2019

Beltrami EJ, Bodin LD (1974) Networks and vehicle routing for municipal waste collection. Networks 4:65–94

Benavent E, Corberán Á, Laganà D, Vocaturo F (2019) The periodic rural postman problem with irregular services on mixed graphs. Eur J Oper Res 276(3):826–839

Bommisetty D, Dessouky M, Jacobs L (1998) Scheduling collection of recyclable material at Northern Illinois University Campus using a two-phase algorithm. Comput Ind Eng 35(34):435–438

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

Campbell AM, Wilson JH (2014) Forty years of periodic vehicle routing. Networks 63:2–15

Chao I-M, Golden BL, Wasil EA (1995) A new heuristic for the period traveling salesman problem. Comput Oper Res 22:553–565

Chen Y, Hao JK (2018) Two phased hybrid local search for the periodic capacitated arc routing problem. Eur J Oper Res 264(1):55–65

Chen B, Qu R, Bai R, Laesanklang W (2020) A variable neighborhood search algorithm with reinforcement learning for a real-life periodic vehicle routing problem with time windows and open routes. RAIRO-Oper Res 54(5):1467–1494

Christofides N, Beasley JE (1984) The period routing problem. Networks 14:237–256

Chu F, Labadi N, Prins C (2005) Heuristics for the periodic capacitated arc routing problem. J Intell Manuf 16(2):243–251

Chu F, Labadi N, Prins C (2006) A scatter search for the periodic capacitated arc routing problem. Eur J Oper Res 169(2):586–605

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

Corberán A, Prins C (2010) Recent results on arc routing problems: an annotated bibliography. Networks 56:50–69

Cordeau J-F, Gendreau M, Laporte G (1997) A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 30:105–119

Damodaran P, Krishnamurthi M, Srihari K (2008) Lower bounds for hierarchical Chinese postman problem. Int J Ind Eng 15:36–44

Delahaye D, Chaimatanan S, Mongeau M (2019) Simulated annealing: from basics to applications. Handbook of metaheuristics. Springer, Cham, pp 1–35

Diabat A, Abdallah T, Le T (2016) A hybrid tabu search based heuristic for the periodic distribution inventory problem with perishable goods. Ann Oper Res 242(2):373–398

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

Dror M (ed) (2000) Arc routing: theory, solutions and applications. Kluwer Academic Publishers, Norwell, Massachusetts

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

Edmonds J (1965) Paths, trees, and flowers. Can J Math 17(3):449–467

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

Eiselt HA, Gendreau M, Laporte G (1995b) Arc routing problems, part II: The rural postman problem. Oper Res 43(3):399–414

Francis P, Smilowitz K, Tzur M (2006) The period vehicle routing problem with service choice. Transp Sci 40(4):439–454

Fröhlich GE, Doerner KF, Gansterer M (2020) Secure and efficient routing on nodes, edges, and arcs of simple-graphs and of multi-graphs. Networks 76(4):431–450

Ghiani G, Laporte G (2000) A branch-and-cut algorithm for the undirected rural postman problem. Math Program 87(3):467–481

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

Ghiani G, Quaranta A, Triki C (2007) New policies for the dynamic traveling salesman problem. Optim Methods Softw 22(6):971–983

Gurobi optimizer 9.0 (2021) High-end libraries for math programming. http://www.gurobi.com/. Accessed Mar 2021

Huang SH, Lin TH (2014) Using ant colony optimization to solve periodic arc routing problem with refill points. J Ind Prod Eng 31(7):441–451

Kaveh F, Tavakkoli-Moghaddam R, Triki C, Rahimi Y, Jamili A (2021) A new bi-objective model of the urban public transportation hub network design under uncertainty. Ann Oper Res 296(1):131–162

Keskin ME, Yılmaz M, Triki C (2021) Solving the hierarchical windy postman problem with variable service costs using a math-heuristic algorithm. Submitted for publication

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

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

Lacomme P, Prins C, Ramdane-Cherif W (2005) Evolutionary algorithms for periodic arc routing problems. Eur J Oper Res 165(2):535–553

Leggieri V, Haouari M, Layeb S, Triki C (2007) The steiner tree problem with delays: a tight compact formulation and reduction procedures. Technical report, University of Salento, Lecce

Leggieri V, Mohamed H, Chefi T (2010) An exact algorithm for the Steiner tree problem with delays. Electron Notes Discrete Math 36:223–230

Liu DQ (1988) Snow clearing vehicle routing: the postman problem in a hierarchical network. MSc. Thesis, University of Manitoba

Mei Y, Tang K, Yao X (2011) A memetic algorithm for periodic capacitated arc routing problem. IEEE Trans Syst Man Cybern Part B Cybern 41(6):1654–1667

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

Oliveira JD, Scarpin CT (2020) A relax-and-fix decomposition strategy based on adjacent nodes applied to the periodic capacitated arc routing problem (PCARP). IEEE Lat Am Trans 18(03):573–580

Paletta G (1992) A multiperiod traveling salesman problem: heuristic algorithms. Comput Oper Res 19:789–795

Paletta G, Triki C (2002) Solving the asymmetric traveling salesman problem with periodic constraints. Networks 44(1):31–37

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

Reeves CR (1993) Modern heuristic techniques for combinatorial problems. Wiley, New Jersey

Riquelme-Rodríguez JP, Gamache M, Langevin A (2014a) Periodic capacitated arc-routing problem with inventory constraints. J Oper Res Soc 65(12):1840–1852

Riquelme-Rodríguez JP, Langevin A, Gamache M (2014b) Adaptive large neighborhood search for the periodic capacitated arc routing problem with inventory constraints. Networks 64(2):125–139

Riquelme-Rodríguez JP, Gamache M, Langevin A (2016) Location arc routing problem with inventory constraints. Comput Oper Res 76:84–94

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

Shahmanzari M, Aksen D, Salhi S (2020) Formulation and a two-phase matheuristic for the roaming salesman problem: application to election logistics. Eur J Oper Res 280(2):656–670

Taslimi M, Batta R, Kwon C (2020) Medical waste collection considering transportation and storage risk. Comput Oper Res 120:104966

Thomaz DV, Loch GV, Scarpin CT, Schenekemberg CM (2018) A mathematical model for the periodic capacitated arc routing problem with time windows. IEEE Lat Am Trans 16(10):2567–2573

Tirkolaee EB, Mahdavi I, Esfahani MMS (2018) A robust periodic capacitated arc routing problem for urban waste collection considering drivers and crew's working time. Waste Manag 76:138–146

Tirkolaee EB, Goli A, Pahlevan M, Malekalipour Kordestanizadeh R (2019) A robust bi-objective multi-trip periodic capacitated arc routing problem for urban waste collection using a multi-objective invasive weed optimization. Waste Manag Res 37(11):1089–1101

Triki C (2013) Solution methods for the periodic petrol station replenishment problem. J Eng Res 10(2):69–77

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

Wang X, Wasil E (2020) On the road to better routes: Five decades of published research on the vehicle routing problem. Networks 77(1):66–87

Wolsey LA (1998) Integer programming, vol 42. Wiley, New York

Zhang Y, Mei Y, Tang K, Jiang K (2017) Memetic algorithm with route decomposing for periodic capacitated arc routing problem. Appl Soft Comput 52:1130–1142

Zhou W, You X, Fan W (2020) A mixed integer linear programming method for simultaneous multi-periodic train timetabling and routing on a high-speed rail network. Sustainability 12(3):1131