On the Hierarchical Directed Capacitated Arc Routing Problem

Minh Hoàng Hà[a], Thu Huong Dang[b], Van Phuc Nguyen[a], Trung Thanh Nguyen[a], André Langevin[c]

[a] *ORLab, Faculty of Computer Science, Phenikaa University, Hanoi, Vietnam*
[b] *Department of Management Science, Lancaster University, Lancaster LA1 4YX, UK*
[c] *Department of Mathematics and Industrial Engineering and CIRRELT, École Polytechnique de Montréal, C.P. 6079, Succursale Centre-ville, Montréal, Qué. Canada H3C 3A7*

Abstract

The Hierarchical Directed Capacitated Arc Routing Problems (HDCARP) is a variant of the Capacitated Arc Routing Problems (CARPs), in which the arcs in a graph are partitioned into clusters. However, unlike traditional CARPs that aim to minimise total time, the HDCARP focuses on minimizing the maximum completion time of each priority class in a hierarchical fashion. Practical applications of the HDCARP include snow plowing, salt spreading, street cleaning, and waste collection. In this study, we explore two variants of the HDCARP. The key difference between these variants lies in the consideration of precedence relations between clusters within routes. We propose MILP formulations and matheuristics for both HDCARP variants. The MILP formulations enable us to find optimal solutions for small-scale instances and evaluate the quality of matheuristics. Our matheuristics are based on decomposing the problem into multiple sub-problems, resulting in faster running time for large-scale instances. We conduct extensive computational experiments to assess the performance of these approaches and present our findings.

Keywords: Arc routing problem; Hierarchical objective; Mixed-integer linear programming, Matheuristic.

1. Introduction

The well-known *Chinese Postman Problems* (CPPs) are a special kind of *Arc Routing Problems* (ARPs), in which all roads should be traversed at least once by a single vehicle, assuming unlimited capacity. [17, 19, 22]. While the CPPs can be solved in polynomial time [6, 10, 18], most ARPs of interest are NP-hard. The readers are referred to [14] for recent results on ARPs. In the CPP, road segments are assigned equal priorities. However, in practice, to optimise route scheduling or improve service efficiency, road are usually categorised into classes based on their importance [9, 25, 30, 31, 33, 34, 35, 38]. For instance, in disaster relief operations, routing problems are crucial for tasks like delivering medical aid, transporting supplies, and distributing food. The affected areas are categorized based on damage intensity, urgency of needs, and road conditions.

The Hierarchical Chinese Postman Problem (HCPP) is an NP-hard variant of the CPP, where edges are grouped into priority classes and a precedence relationship determines their

Email address: hoang.haminh@phenikaa-uni.edu.vn (Minh Hoàng Hà)

traversal order. The objective is to find the shortest route that starts and ends at a depot (see [16]). The *Hierarchical Rural Postman Problems* (HRPPs) are the generalization of the HCPP where only a subset of road segments require a service (see [4, 12, 13]). Typical applications of HCPPs and HRPPs include flame cutting, street cleaning, garbage collection, salt spreading, and snow plowing (e.g., [4, 8, 16, 21, 24, 25, 28]).

While the precedence relationship is commonly encountered in real-life scenarios, there are only a limited number of studies that specifically address ARPs incorporating service hierarchy. Moreover, most research efforts have focused on the HCPPs and HRPPs assuming unlimited vehicle capacity, which is unrealistic for real-world scenarios. However, constraints on vehicle capacity are typically critical in various applications. For example, in snow removal, routes must be planned to ensure that the total amount of de-icing materials loaded on a vehicle does not exceed its capacity (see [3]).

In this study, we examine the *Hierarchical Directed Capacitated Arc Routing Problem* (HDCARP), which extends the HRPPs by incorporating practical aspects such as one-way road segments, multiple vehicles, and capacity constraints. To be specific, the HDCARP aims to determine a set of routes for a fleet of identical vehicles, subject to the following conditions: (i) each vehicle must start and end its route at the depot; (ii) each required road must be serviced exactly once; (iii) the total demand on each route must not exceed the vehicle capacity, and (iv) the order in which the roads are serviced on each route must respect the specified precedence relation.

The objective of the HDCARP is to first minimise the maximum completion time of the first priority class, followed by the second priority class, and so on. The maximum completion time of a class is defined as the minimum time required for all vehicles to complete servicing all roads within that class. This objective is referred to as the *hierarchical objective*, as described in [8, 30]. The hierarchical objective is particularly suitable for practical applications where roads are categorized based on their priority. It ensures that roads with higher priority are serviced before proceeding to lower priority roads. Therefore, this objective aligns with the concept of precedence constraints.

There are two variants of precedence constraints studied in the literature. The predominant focus is on the *linear precedence relations*, which establish a unique lexicographical ordering for all classes within a route [8, 13, 16, 26, 30]. This variant arises when roads belonging to a class can only be serviced after all higher priority roads have been successfully serviced. Another variant of precedence constraints that has been studied in the literature is the *general precedence relations* [16, 30]. In this variant, all roads of high-priority class must be serviced before those of low-priority class. However, medium-priority roads are allowed to be serviced before or after certain high-priority and low-priority roads. There are several studies [30, 39] in the literature that do not impose precedence constraints between any pair of classes. Instead, the ordering of classes is determined solely by the hierarchical objective. This is referred to as *class upgrading* possibility, which is beneficial for improving the service quality of low priority classes and reducing the total completion time. The linear precedence and upgrading possibility have received more attention in the literature, primarily due to their practical applications and relevance [32, 33, 34]. Consequently, we consider these variants in this paper.

This paper has several contributions. First, we explore the HDCARP, a new problem that generalizes both the HCPPs and the HRPPs. We study both linear precedence constraints and the upgrading possibility, resulting in two HDCARP variants. Second, we provide *Mixed Integer Linear Programming* (MILP) formulations for both variants. We point out a special case that is not adequately addressed by existing formulations in the literature on ARPs with a

hierarchical objective. Additionally, we propose a new approach for expressing the hierarchical objective. Third, we improve the computational time by developing efficient matheuristics for both variants that decompose the original problem into several sub-problems. Finally, we present some extensive computational results on randomly created instances.

The rest of the paper is structured as follows. Section 2 contains a brief literature review. Section 3 provides a formal description and mathematical formulation for each HDCARP variant. Section 4 presents MILP-based matheuristics and Section 5 presents the computational results. Finally, some conclusions and future research directions are given in Section 6.

2. Literature review

In this section, we briefly review the relevant literature. We discuss the HCPPs in Subsection 2.1, the HRPPs in Subsection 2.2, and other ARPs with hierarchical services in Subsection 2.3.

2.1. Hierarchical chinese postman problems

To our knowledge, the first paper that formally addressed the HCPP was Dror et al. [16]. The authors showed that the HCPP is generally NP-hard. However, when all subgraphs induced by the classes are connected and the precedence relations are linear, the HCPP can be solved in polynomial time, via a matching problem and a series of shortest path problems. Improved versions of this algorithm, suitable for large-scale instances, was given in [21, 25, 37]. More recently, Afanasev et al. [1] proved that the HCPP with connected classes is generally NP-hard. The authors proposed a 5/3-approximation algorithm in polynomial time for the HCPP with linear precedence relations.

Several algorithms have been proposed for the HCPP with linear precedence relations in general cases. Lemieux and Gampagna [26] proposed a simple heuristic based on Euler circuits for the case of two priority classes. Cabral et al. [8] converted the HCPP to the equivalent RPP and applied branch-and-cut procedures to solve it. Çodur & Yılmaz [11] formulated the HCPP as a MILP and proposed two metaheuristics based on genetic algorithm and hybrid simulated annealing for large-scale instances. Damodaran et al. [15] used the optimal solution to the CPP as a lower bound for the HCPP. The authors also proposed a heuristic algorithm to enhance this lower bound within a short computational time.

Regarding directed graphs, Alfa and Liu [4] addressed the *Directed HCPPs* (HDCPPs) with general precedence relations. They introduced a new constraint, requiring the service of a higher priority class to be started and completed before that of a lower priority class. They proposed a three-phase heuristic approach to address this problem, which involved connecting classes, balancing non-symmetric nodes, and identifying feasible routes. A dynamic programming algorithm was later presented in [20] for HDCPPs with general precedence relations and class connectivity.

Perrier et al. [30] addressed the problem of *Mixed Multi-vehicle* HCPPs (m-HMCPP) with class upgrading possibilities. The m-HMCPP is the generalisation of the HCPP where both edges and arcs can be present, and multiple vehicles with unlimited capacity are available. They first performed a graph transformation and then formulated the m-HMCPP as a MILP model. The model was adaptable to incorporate additional constraints and different situations. Furthermore, the authors introduced two constructive heuristics based on decomposing the problem into sub-problems. For a more on the HCPPs, see the surveys [33, 34].

2.2. Hierarchical rural postman problems

Quirion-Blais et al. [36] proposed an adaptive large neighborhood search metaheuristic for the multi-vehicle HRPP with general precedence relations. The metaheuristic is suitable for large-scale instances and can be customized to accommodate operational constraints.

The concept of the Hierarchical Mixed RPP (HMRPP) was introduced by Perrier et al. [29] in 2006 and further studied in 2017 [12, 13, 35]. Comlombi et al. [12] proposed a mathematical programming formulation, a matheuristic, and a Tabu Search for the HMRPP with linear precedence relations. The matheuristic solves a series of Mixed Rural Postman Problems for each class. Comlombi et al. [13] provided polyhedral results and introduced facet-inducing inequalities based on the formulation in [12]. Quirion-Blais et al. [35] focused on the multi-vehicle version and developed an adaptive large neighborhood metaheuristic.

2.3. Other ARPs with hiearchy services

Recently, Ahabchane et al. [2] introduced the *Hierarchical Mixed Capacitated General Routing Problem* (HMCGRP). The HMCGRP is a generalisation of the HMRPP in which both nodes and edges may require service. Notably, this work is the first and only one in the literature to consider capacity constraint and demand uncertainty. In their approach, road hierarchies are modelled with time-dependent costs. For small-scale instances, a robust formulation with graph transformation was developed, while for large-scale instances, a two-phase metaheuristic based on simulated annealing and ruin-and-recreate strategies was proposed.

Paper	Lexicographic Objective	Precedence Relation	Upgrading Allowed	Multiple Vehicles	Capacity Constraint	Data Type	Graph Type
Dror et al. [16]	-	X	-	-	-	Deterministic	(Un)Directed
Cabral et al. [8]	X	X	-	-	-	Deterministic	Undirected
Comlombi et al. [12, 13]	-	X	-	-	-	Deterministic	Mixed
Perrier et al. [30]	X	X	X	X	-	Deterministic	Directed
Quirion-Blais et al. [36]	X	-	X	X	-	Deterministic	Mixed
Quirion-Blais et al. [35]	X	X	-	X	-	Deterministic	Mixed
Ahabchane et al. [2]	-	-	X	X	X	Uncertain	Mixed
This paper	X	X	X	X	X	Deterministic	Directed

Table 1: Synthesis of studies related to routing problems with service hierarchy.

Table 1 situates our research problem within the context of studies on the service hierarchy in the ARP literature. We extend the research direction of [8, 12, 13, 16] by considering multiple vehicles and capacity constraints. Our main objective is to analyze and compare the results of our proposed algorithms to understand the computational difficulty of solving these variants. This study is the first to report such results for capacitated ARPs with service hierarchy.

3. Mathematical formulations

In this section, we present the problems and mathematical formulations. First, in Subsection 3.1, we formally describe two HDCARP variants, along with notations, terminology and graph transformation. Then, in Subsections 3.2 and 3.3, we present the mathematical formulations for both variants.

3.1. Problem definition and graph transformation

We are given a strongly connected directed graph $G = (V, A)$, where V is the vertex set, and A is the set of (directed) arcs. The vertices are denoted by $v_0, v_1, \cdots, v_n$, and vertex v_0

is called the *depot*. This graph represents a road network, in which road junctions and key landmarks are represented by vertices, one-way road segments are represented by arcs, and two-way road segments are represented by a pair of directed arcs, one in each direction. We are also given a set $A_r \subseteq A$ of *required arcs*. We call $A_{nr} = A \setminus A_r$ the set of non-required arcs. The set of required arcs A_r is partitioned into p pairwise disjoint classes $A_r^1, A_r^2, \cdots, A_r^p$ ($A_r = A_r^1 \cup A_r^2 \cup ... \cup A_r^p$, and $A_r^h \cap A_r^k = \emptyset$ for $h \neq k$ and $h, k \in \{1, 2, \cdots, p\}$). For convenience, we denote $P = \{1, 2, \cdots, p\}$. Let V_t^k be the set of tail vertices of arcs A_r^k for each $k \in P$. Let V_t denote $V_t^1 \cup V_t^2 \cup ... \cup V_t^p$.

Figure 1 illustrates the concept of classes and notations. Required and non-required arcs are drawn by thick solid and dashed lines, respectively. The class numbers are indicated on the required arcs. Nodes in V_t are represented by hollow circles. One can check that $V_t^1 = \{v_0, v_2, v_3, v_4\}$, $V_t^2 = \{v_1\}$, and $V_t^3 = \{v_3, v_4\}$.

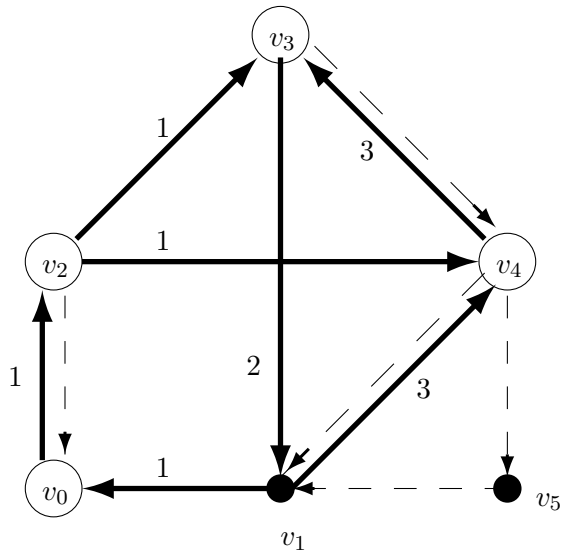


Figure 1: Graph G with priority classes

Each arc $a_i \in A$ has a positive *traversal time* d_i. Each required arc $a_i \in A_R$ is associated with a positive *demand* q_i, *servicing time* s_i, and *priority level* p_i. A fleet of identical vehicles M is located at the depot, each with positive capacity Q. Hence, the load of each vehicle must not exceed Q at any time. In the HDCARP, a vehicle must depart from the depot, service some required arcs while respecting the capacity constraint, the linear precedence relations between classes, and must return to the depot. Each required arc must be serviced by exactly one vehicle. Traversing an arc without servicing is called *deadheading*.

In this article, we consider both linear precedence relations and class upgrading possibility resulting in two variants of the HDCARP, called the HDCARP-P and HDCARP-U, respectively. (The "-P" suffix recalls the original HDCARP with linear precedence relations, whereas the "-U" suffix is to remind us of class upgrading possibility). The only difference between two variants is whether low priority arcs is allowed to be served before higher priority arcs. Specifically, in the HDCARP-P, arcs in class k can only be traversed after completing the service of all arcs in class $k-1$. In contrast, in the HDCARP-U, vehicles can traverse arcs in class k before completing the service of all arcs in class $k-1$.

We now consider the HDCARP instance shown in Figure 1. Assume that each arc has a unit traversal time, each required arc has a unit servicing time and a unit demand, and two identical vehicles are located at v_0 with a capacity of $Q = 4$. Let's examine a feasible solution for the HDCARP-P shown in Figure 2(a). The first route serves three required arcs of class 1, while the second route sequentially serves remaining required arcs according to the linear precedence relations. The completion time for class 1 is 4 on route 1 and 2 on route 2, resulting in a maximum completion time of 4 for class 1. Similarly, one can check that the maximum completion times for class 2 and 3 are 4, and 6, respectively. Note that this solution is also feasible for the HDCARP-U. Figure 2(b) illustrates another feasible solution for the HDCARP-U, where the required arc in class 2 is serviced before the one in class 1 in route 1. It can be verified that the maximum completion times of class 1, 2, and 3 are $4, 3$, and 5, respectively. If upgrades are permitted, the solution shown in Figure 2(b) is considered better than the solution in Figure 2(a) in terms of completing classes at the earliest possible time (and the total time). However, the solution in Figure 2(b) is infeasible for the HDCARP-P.

This suggests that the HDCARP-U can improve the maximum completion time of each class in the HDCARP-P's solution, especially if the majority of time is spent deadheading. This is further confirmed in Section 5. However, the main drawback of the HDCARP-U is the unknown order in which classes are traversed on each route, making it more challenging to solve.

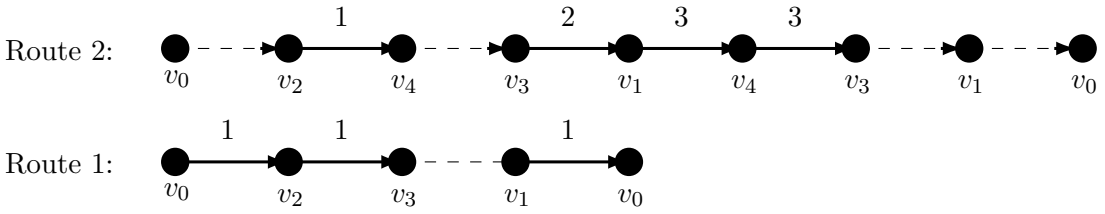
We consider the hierarchical objective for both HDCARP-P and HDCARP-U. A common way to express this objective in the literature is the weighted sum of the maximum completion times of each priority class [8, 13, 35, 36]. In other words, the weighted sum $\sum_{k=1}^{p} M_k T_k$ is minimised, where T_k is the maximum completion time of the class k, and M_k are coefficients such that $M_1 \gg M_2 \gg ... \gg M_p$. It can be seen that handling the large constants M_k $(k \in P)$ in the objective function can be difficult as often happening in integer programming. To get around this issue, we use the lexicographic optimization approach to model the hierarchical objective. Simply put, the approach starts with the minimization of the first objective, then among the possible optima minimises the second objective and so on, until all p objective functions are minimised. We call sequence $(T_1, \cdots, T_p)$ the *maximum completion time sequence*. To compare any two solutions, we use the lexicographic order to compare their maximum completion time sequences. More precisely, the solution is considered better if the first different element in its maximum completion time sequence is smaller.

When dealing with the HDCARPs, we observe special cases where a low priority class may have a shorter maximum completion time than a higher priotiry class. These could happen in both non-upgrade and upgrade variants. However, these cases are excluded in any formulation for multiple-vehicle HCPPs or HRPPs in the literature. For example, the ones in [30, 35] defined that $T_1 \leq T_2 \leq \cdots \leq T_p$.

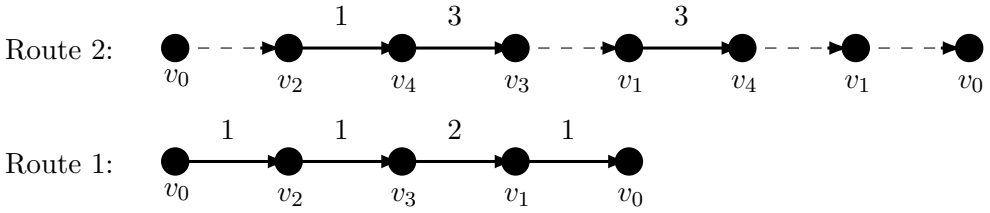
These special cases are best illustrated through the following HDCARP instance shown in Figure 1. Now consider a feasible solution of the HDCARP-P in Figure 3. One can check that $T_1 = 4$, $T_2 = 3$ and $T_3 = 5$. It is clear that $T_2 < T_1$ even though class 2 has a lower priority than class 1. We would like to highlight that our MILP formulations and matheuristics are capable of capturing and addressing such special cases.

For what follows, it is helpful to define an auxiliary directed multigraph, which we denote by $G' = (V', A')$. Our transformation is inspired by the one in [30] where a "dummy" vertex is introduced to ensure connectivity between classes on each route. However, we propose a slightly different way to transform graph G to the sparser multigraph and utilise the dummy node to address the special cases, which will become clear.

The multigraph is constructed as follows: Initially, $G' = (V', A')$ is a duplicate of G. Then,



a, A feasible solution for the HDCARP-P



b, A feasible solution for the HDCARP-U

Figure 2: Feasible solutions for the HDCARP instances in Figure 1a.

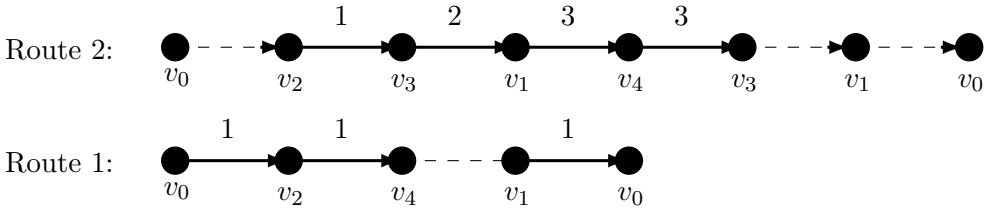


Figure 3: A feasible solution for the HDCARP-P instances in Figure 1b.

we add a dummy node v_0' to the node set V' ($V' = \{v_0'\} \cup V$). This dummy node serves as an intermediate node when transitioning between different classes on each route. For instance, when a vehicle completes the service of certain required arcs in class $k \in P$, it is required to visit node v_0' before starting the service of required arcs in another class $k' \in P$ ($k' \neq k$). Moreover, the node visited immediately after v_0' must be the same as the node visited right before v_0'. Every vehicle must make a visit to v_0' both before departing from the depot and after completing the service of the last required arcs on its assigned route.

Next, we construct two sets of dummy arcs A_f and A_t. For each vertex $v \in V_t \cup \{v_0\}$, we add the arcs $\{v, v_0'\}$ and $\{v_0', v\}$ to A_t and A_f, respectively. These arcs have zero traversal time. Finally, for each arc $a \in A_f \cup A_t$, we add arc a to A'. In other words, $A' = A \cup A_f \cup A_t$, which means $|A'| = |A| + 2|V_t| + 2$.

Figure 4 represents how the above-mentioned transformation works for the graph G in Figure 1, with dummy arcs represented as dotted lines. Furthermore, Figure 5(a) and 5(b)

show the routes in G' corresponding to the feasible solution of the HDCARP-P in Figure 2(a) and the feasible solution of the HDCARP-U in Figure 2(b), respectively.

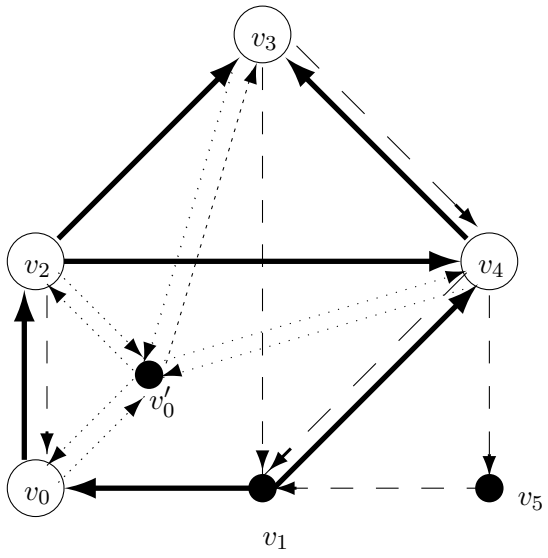


Figure 4: Auxiliary directed multigraph G' for the HDCARP instance in Figure 1

For convenience, we use some additional notations. Given a set $S \subseteq V'$, $A(S)$ denotes the set of arcs with both end-nodes in S, and $\delta(S)$ denotes the set of arcs with exactly one end-node in S. We denote $\delta^+(S)$ and $\delta^-(S)$ as the sets of arcs in G' leaving and entering S, respectively. Let's denote $A_r(S)$ as $A(S) \cap A_r$, and similarly for $\delta_r(S)$, $\delta_r^+(S)$, and $\delta_r^-(S)$. For each class $k \in \{1, 2, \cdots, p\}$, we also let $A_r^k(S)$ denote $A(S) \cap A_r^k$, and similarly for $\delta_r^k(S)$, $\delta_r^{+k}(S)$ and $\delta_r^{-k}(S)$. For simplicity, we sometimes write $\delta(v)$ instead of $\delta(\{v\})$. We also write $\delta_k(S), \delta_k^+(S)$, and $\delta_k^-(S)$ instead of $\delta_r^k(S)$, $\delta_r^{+k}(S)$, and $\delta_r^{-k}(S)$, respectively.

3.2. MILP model for the HDCARP-P

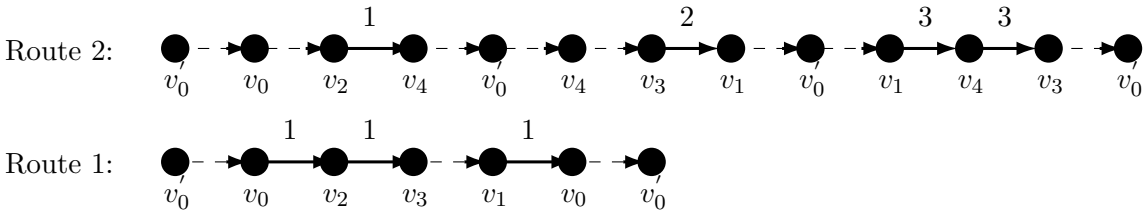
Our MILP model uses the following variables:

- Binary variables x_a^m indicate whether vehicle $m \in M$ services arc $a \in A_r$.
- Integer variables y_{ak}^m count the number of times vehicle $m \in M$ deadheads through arc $a \in A'$ in class $k \in P$ in its chosen path.
- Non-negative real variables t_k^m represent the service completion time of class $k \in P$ on route $m \in M$ (assigned to vehicle m).
- Binary variables r_k^m indicate whether vehicle $m \in M$ services any required arcs in class $k \in P$.
- A non-negative real variable T_k represent the service completion time of class $k \in P$.

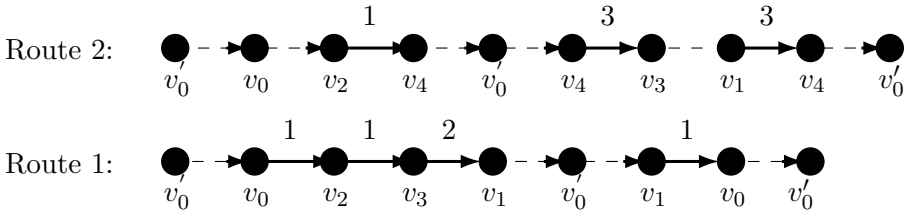
The MILP is then as follows:

$$\underset{k=1,\ldots,p}{\text{lex-min}} \quad T_k \tag{1}$$

$$\text{subject to} \quad T_k \geq t_k^m - N(1 - r_k^m) \qquad m \in M; k \in P \tag{2}$$



(a) The routes in G' corresponding to the solution in Figure 2(a).



(b) The routes in G' corresponding to the solution in Figure 2(b).

Figure 5: An illustration of routes in G'

$$t_k^m = t_{k-1}^m + \sum_{a \in A_r^k} s_a x_a^m + \sum_{a \in A} d_a y_{ak}^m \qquad m \in M; k \in P \tag{3}$$

$$t_0^m = 0 \qquad m \in M \tag{4}$$

$$\sum_{a \in A_r^k} x_a^m \leq |A_r^k| r_k^m \qquad m \in M; k \in P \tag{5}$$

$$y^m_{\{v'_0, v_0\}1} = 1 \qquad m \in M \tag{6}$$

$$\sum_{a \in A_f} y_{ak}^m = 1 \qquad m \in M; k \in P \tag{7}$$

$$y^m_{\{v'_0, v_i\}k} = y^m_{\{v_i, v'_0\}k-1} \qquad m \in M; k \in P \setminus \{1\}; v_i \in \bigcup_{h=1}^{k-1} V_t^h \cup \{v_0\} \tag{8}$$

$$\sum_{m \in M} x_a^m = 1 \qquad k \in P; a \in A_r^k \tag{9}$$

$$\sum_{k=1}^{p} \sum_{a \in A_r^k} q_a x_a^m \leq Q \qquad m \in M \tag{10}$$

$$\sum_{a \in \delta_k^+(v_i)} x_a^m + \sum_{a \in \delta^+(v_i)} y_{ak}^m = \sum_{a \in \delta_k^-(v_i)} x_a^m + \sum_{a \in \delta^-(v_i)} y_{ak}^m \qquad m \in M; k \in P; v_i \in V' \tag{11}$$

$$\sum_{a \in \delta_k^+(S)} x_a^m + \sum_{a \in \delta^+(S)} y_{ak}^m \geq x_b^m \qquad m \in M; k \in P; S \subseteq V \setminus \{v_0\}; \forall b \in A_r^k(S) \tag{12}$$

$$x_a^m \in \{0,1\} \qquad m \in M; k \in P; a \in A_r^k \tag{13}$$

$$y_{ak}^m \in \mathbb{N} \qquad m \in M; k \in P; a \in A' \tag{14}$$

$$t_k^m \in \mathbb{R}^+ \qquad m \in M; k \in P \tag{15}$$

$$r_k^m \in \{0,1\} \qquad m \in M; k \in P \tag{16}$$

$$T_k \in \mathbb{R}^+ \qquad k \in P \tag{17}$$

The objective function (1) represents the hierarchical objective, while constraints (2) ensure that the maximum completion time of class k is greater than or equal to the completion time of that class on any route. Here, N is a suitably large number. Constraints (3) and (4) are time-conservation constraints, ensuring route connectivity. Constraints (5) limit the number of arcs in class k that can be serviced on each route. Constraints (6), (7), and (8) define class transitions on each route using node v_0'. Constraints (9) and (10) guarantee that each required arc is serviced exactly once and that vehicle capacity is not exceeded. Constraints (11) ensure that the number of times a vehicle departs from a node is equal to the number of times it arrives at that node for each class. Constraints (12) state that if route m services arc b, it must traverse any arc cutset that separates b from the depot. The remaining constraints define the domains of the variables.

3.3. MILP model for the HDCARP-U

In the HDCARP-U, where class upgrading is allowed, the specific order in which classes are traversed within each route is unknown. To address this, we introduce the concept of *hierarchy level*. The hierarchy level indicates the completion of a class's service within each route. This concept is illustrated in Figure 2(b). In route 1, three required arcs $\{v_0, v_2\}$, $\{v_2, v_3\}$, and $\{v_3, v_1\}$ are in hierarchy level 1, marking the service completion of class 2. The required arc $\{v_1, v_0\}$ is in hierarchy level 2, marking the service completion of class 1. Hierarchy level 3 does not have any required arcs. Similarly, in route 2, hierarchy levels 1 and 2 indicate the completion of servicing classes 1 and 3, respectively, while hierarchy level 3 does not contain any required arcs. It's important to note that in each route, the number of hierarchy levels is equal to the number of classes, including the possibility of empty sets. To represent transitions between hierarchy levels, we use the dummy node v_0'. To avoid confusion, we use the notation h to represent the hierarchy level and k to represent the class.

The MILP model for the HDCARP-U, similar to HDCARP-P, makes use of five variable types: x, r, y, t, and T. While the definitions of the last variable type remain unchanged, the first four variable types have been redefined as follows:

- Binary variables x_{ah}^m indicate whether vehicle $m \in M$ services the required arc $a \in A_r$ in hierarchy level $h \in P$.
- Integer variables y_{ah}^m count the number of times vehicle $m \in M$ deadheads through arc $a \in A'$ in hierarchy level $h \in P$ in its chosen path.
- Non-negative real variables t_h^m represent the time at which vehicle $m \in M$ completes servicing all required arcs in hierarchy level $h \in P$.
- Binary variables r_{kh}^m indicate whether hierarchy $h \in P$ contains any required arcs in class k on route $m \in M$ (assigned to vehicle m).

The MILP model for the HDCARP-U is described as follows:

$$\underset{k=1,\ldots,p}{\text{lex-min}} \quad T_k \tag{18}$$

$$\text{subject to} \quad T_k \geq t_h^m - N(1 - r_{kh}^m) \qquad m \in M; h, k \in P \tag{19}$$

$$t_h^m = t_{h-1}^m + \sum_{a \in A_r} s_a x_{ah}^m + \sum_{a \in A} d_a y_{ah}^m \qquad m \in M; h \in P \tag{20}$$

$$t_0^m = 0 \qquad m \in M \tag{21}$$

$$\sum_{a \in A_r^k} x_{ah}^m \leq |A_r^k| r_{kh}^m \qquad m \in M; h, k \in P \tag{22}$$

$$y_{\{v_0', v_0\}1}^m = 1 \qquad m \in M \tag{23}$$

$$\sum_{a \in A_f} y_{ah}^m = 1 \qquad m \in M; h \in P \tag{24}$$

$$y_{\{v_0', v_i\}h}^m = y_{\{v_i, v_0'\}h-1}^m \qquad m \in M; h \in P \setminus \{1\}; v_i \in V_t \cup \{v_0\} \tag{25}$$

$$\sum_{m \in M} \sum_{h=1}^{p} x_{ah}^m = 1 \qquad a \in A_r \tag{26}$$

$$\sum_{h=1}^{p} \sum_{a \in A_r} q_a x_{ah}^m \leq Q \qquad m \in M \tag{27}$$

$$\sum_{a \in \delta_r^+(v_i)} x_{ah}^m + \sum_{a \in \delta^+(v_i)} y_{ah}^m = \sum_{a \in \delta_r^-(v_i)} x_{ah}^m + \sum_{a \in \delta^-(v_i)} y_{ah}^m \qquad m \in M; h \in P; v_i \in V' \tag{28}$$

$$\sum_{a \in \delta_r^+(S)} x_{ah}^m + \sum_{a \in \delta^+(S)} y_{ah}^m \geq x_{bh}^m \qquad m \in M; h \in P; S \subseteq V \setminus \{v_0\}; \forall b \in A_r(S) \tag{29}$$

$$x_{ah}^m \in \{0, 1\} \qquad m \in M; h \in P; a \in A_r \tag{30}$$

$$y_{ah}^m \in \mathbb{N} \qquad m \in M; h \in P; a \in A' \tag{31}$$

$$t_h^m \in \mathbb{R}^+ \qquad m \in M; h \in P \tag{32}$$

$$r_{kh}^m \in \{0, 1\} \qquad m \in M; h, k \in P \tag{33}$$

$$T_k \in \mathbb{R}^+ \qquad k \in P \tag{34}$$

Constraints (19) ensure that the maximum completion time of a class is greater than or equal to the completion time of any hierarchy level on any route that includes at least one required arc of that class. Constraints (20), (26), and (29) are analogous to constraints (3), (9), and (12) of the HDCARP-P model, respectively, with the exception that required arcs can be serviced at any hierarchy level. Constraints (22) limit the maximum number of required arcs of class k that can be serviced in a single hierarchy level on each route. The remaining constraints and objective function are straightforward and therefore we omit it for brevity.

Since the HDCARP-U is an upgraded version of HDCARP-P, any feasible solution for the HDCARP-U is also feasible for the HDCARP-P. Therefore, the optimal maximum completion time sequence of the HDCARP-U is considered to be lexicographically better to that of the HDCARP-P. However, solving the HDCARP-U model can be more challenging and time-consuming due to the larger solution space.

3.4. Branch-and-cut algorithms

We develop branch-and-cut algorithms to solve the MILP formulations. In our algorithms, the models are first solved without the connectivity constraints ((12) for the HDCARP-P and (29 for the HDCARP-U). We proceed by searching for violated connectivity inequalities. Any violated constraints are then included in the current MILP, which is subsequently reoptimised. This process is repeated until all the connectivity constraints are satisfied. If there are fractional variables, we branch to generate two new sub-problems. If all the variables are integer, we explore another sub-problem. In order to identify violated connectivity inequalities, both heuristic and exact procedures can be used. Though exact method can be done by solving min-cut problems, it is quite time consuming as shown in [23]. Hence, a simple heuristic is developed that first computes the connected components, then defines the connectivity constraints for each component. For more details about the approach to handle the connectivity constraints, we refer the reader to [23].

Our branch-and-cut algorithms are built around CPLEX 12.10 with the Concert Technology library. We turn off all CPLEX solver's cuts and set all the CPLEX parameters to their default values.

4. MILP-based matheuristics

In this section, we propose matheuristics for both variants based on MILPs. These matheuristics are specially tailored to provide good solutions within a reasonable time limit. The details are outlined in Algorithm 1. The idea is to decompose the original problems into p sub-problems, each focused on constructing partial routes for a subset of A_r. The kth sub-problem ($k \in \{1, \cdots, p\}$) keeps track of the flow of time t_k^m and the remaining capacity l_k^m for each vehicle $m \in M$. These values are subsequently used in the $(k+1)$th sub-problem. Each sub-problem consists of two objective functions, prioritised as follows: (1) Minimise the maximum duration of partial routes, and (2) Minimise the total duration of the partial routes. The second objective is introduced with the aim of potentially decreasing the value of the primary objective for the subsequent sub-problem.

This approach offers the advantage of reducing the problem size so that it can be tackled more easily and quickly. The sub-problems and their MILP formulation are described in the subsequent subsections.

Algorithm 1 MILP-based matheuristic

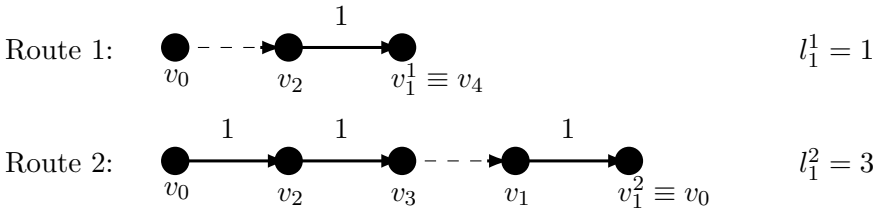
Set $t_0^m = l_0^m = 0$ for all $m \in M$
for $k = 1, \cdots, p$ **do**
  Set up the MILP the kth sub-problem
  Solve the MILP with the branch-and-cut algorithm in Section 3.4
  Update t_k^m and l_k^m for all $m \in M$
  Update T_k
end
output: T_k for all $k \in P$

4.1. Sub-problems for the HDCARP-P

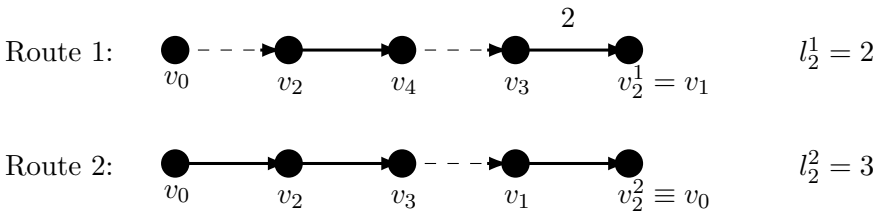
The HDCARP-P is decomposed into p sub-problems, each aimed at finding a set of partial routes for $|M|$ vehicles to service each class. Specifically, let's consider the scenario where $k-1$ sub-problems have already been solved ($1 \leq k \leq p$), and $|M|$ partial routes have been created for $|M|$ vehicles to service all required arcs in $A_r^1 \cup \cdots A_r^{k-1}$. Assuming each partial route m takes t_m^{k-1} units of time, vehicle m is currently at node v_m^{k-1} with a load of l_m^{k-1}. The goal of the kth sub-problem is to identify a set of $|M|$ new partial routes that service each required arc in A_r^k exactly once, while satisfying the following conditions: (1) The partial route for vehicle m must begin from node v_m^{k-1}, and (2) The total load of the partial route for vehicle m must not exceed the maximum remaining load capacity, which is $Q - l_m^{k-1}$. These new partial routes are then connected to the existing ones based on linear precedence relations. As mentioned earlier, this sub-problem consists of two objective functions. The primary objective is to minimise the service completion time T_k for class k, while the secondary objective is to minimise the total travelling time of the partial routes ($\sum_{m \in M} t_m^k$).

Figure 6 shows the solutions for two sub-problems of the HDCARP instance in Figure 1. Figure 6(a) illustrates one feasible solution for the first sub-problem. It consists of two partial

routes assigned to two vehicles, servicing required arcs in class 1. The first vehicle is currently located at node v_4 after $t_1^1 = 2$ units of time, carrying a load of $l_1^1 = 1$ unit. The second vehicle returns to node v_0 after $t_1^2 = 4$ units of time, carrying a load of $l_1^2 = 3$ units. Figure 6(b) presents a solution after solving the first two sub-problems for class 1 and 2. The single required arc in the second class is serviced by the first vehicle. After an additional two units of time ($t_2^1 = 4$), the first vehicle is currently located at node v_1 and carries a demand of $l_2^1 = 2$ units.



a, Two partial routes after solving 1st sub-problem for A_r^1



b, Two partial routes obtained after solving two sub-problems for $A_r^1 \cup A_r^2$

Figure 6: An illustration of sub-problems for the HDCARP-P instance in Figure 1.

We will now present the MILP formulation for the kth sub-problem, using the same notation as in the MILP model for the HDCARP-P described in Section 3.2 for consistency. However, since in the kth sub-problem only arcs in A_r^k are required to be serviced, we will omit the "class" index from the variables y, t, and r for simplicity. Specifically, the MILP model for the kth sub-problem uses the following variables:

- For $a \in A_r^k$ and $m \in M$, let $x_a^m \in \{0, 1\}$ be a variable indicating whether vehicle m services arc a.
- For $a \in A'$ and $m \in M$, let $y_a^m \in \mathbb{Z}^+$ be a variable representing the number of times vehicle m deadheads the arc a.
- For $m \in M$, let $r_m \in \{0, 1\}$ be a variable indicating whether vehicle m services any arc in A_r^k.
- For $m \in M$, let $t_k^m \in \mathbb{R}^+$ represent the partial route duration for vehicle m.
- Let $T_k \in \mathbb{R}^+$ be a variable representing the maximum completion time of class k.

The MILP for the kth sub-problem is as follows:

$$\text{lex-min} \quad T_k, \sum_{m\in M} t_k^m \tag{35}$$

$$\text{subject to} \quad T_k \geq t_k^m - N(1-r_m) \qquad m \in M \tag{36}$$

$$t_k^m = t_{k-1}^m + \sum_{a\in A_r^k} s_a x_a^m + \sum_{a\in A} d_a y_a^m \qquad m \in M \tag{37}$$

$$\sum_{a\in A_r^k} x_a^m \leq |A_r^k| r_m \qquad m \in M \tag{38}$$

$$y^m_{\{v_0',v_m^k\},k+1} = 1 \qquad m \in M \tag{39}$$

$$\sum_{a\in A_f} y_a^m = 1 \qquad m \in M \tag{40}$$

$$\sum_{m\in M} x_a^m = 1 \qquad a \in A_r^k \tag{41}$$

$$\sum_{a\in A_r^k} q_a x_a^m \leq Q - l_k^m \qquad m \in M \tag{42}$$

$$\sum_{a\in\delta_k^+(v)} x_a^m + \sum_{a\in\delta^+(v)} y_a^m = \sum_{a\in\delta_k^-(v)} x_a^m + \sum_{a\in\delta^-(v)} y_a^m \qquad m \in M, v \in V' \tag{43}$$

$$\sum_{a\in\delta_k^+(S)} x_a^m + \sum_{a\in\delta^+(S)} y_a^m \geq x_b^m \qquad m \in M, S \subseteq V\setminus\{0\}, b \in A_r^k(S) \tag{44}$$

$$x_a^m \in \{0,1\} \qquad m \in M, a \in A_r^k \tag{45}$$

$$y_a^m \in \mathbb{N} \qquad m \in M, a \in A' \tag{46}$$

$$t_k^m \in \mathbb{R}^+ \qquad m \in M \tag{47}$$

$$r_m \in \{0,1\} \qquad m \in M \tag{48}$$

$$T_k \in \mathbb{R}^+ \tag{49}$$

The objective function (35) minimises the maximum completion time of class k and then minimises the total duration of the partial routes. Constraints (36) and (37) define the maximum completion time of class k and the completion time of class k on route m. Constraints (39-40) ensure that each vehicle continues its route from node v_m^k. Constraints (42) ensure that the remaining capacity of the vehicles is not exceeded. The remaining constraints are trivial.

It can be observed that the MILP formulation for the kth subproblem contains only $|M|(|A_r^k|+|A|+2|V_t|+p+3)+1$ variables, while the MILP formulation for the HDCARP-P in Section 3.2 has nearly p times the number of variables, specifically $|M|\Big(|A_r|+p(|A|+2|V_t|+4)\Big)+p$ variables. Moreover, the subproblem objective has 2 hierarchical levels, while the HDCARP-P objective has k hierarchical levels. This suggests that the proposed matheuristic, denoted as MB1, is expected to be faster than solving optimally the MILP formulation for the HDCARP-P, especially when p is large.

It is worth noting that each sub-problem may have different optimal solutions, but due to the time restriction, only one solution is used to generate the solution for the original problem.

4.2. Sub-problems for the HDCARP-U

Similarly, the HDCARP-U is decomposed into p sub-problems. However, unlike in the HDCARP-P, we cannot treat each class separately in the HDCARP-U, as the order of classes on each route is unknown. Therefore, we need to redefine $p-1$ sub-problems, except for the first one. The resulting matheuristic is denoted as MB2-U.

Consider the scenario where $k-1$ sub-problems have been solved ($1 \leq k \leq p$), and all $|M|$ vehicles have successfully completed servicing the required arcs from classes 1 to $k-1$. At this stage, we have detailed information about the assignment of required arcs to hierarchy levels on each route. For each vehicle $m \in M$ and each hierarchy level $1 \leq h \leq k-1$, let A_r^{hm} denote the set of required arcs serviced by vehicle m in hierarchy level h. Additionally, we also know the completion time of hierarchy level h on route m (assigned to vehicle m), denoted as t_h^m, and the current load of each vehicle m at time t_h^m, denoted as l_h^m. We define the maximum completion time of hierarchy level h across all routes as C_h, given by $C_h = \max_{m \in M}\{t_h^m\}$.

The kth sub-problem aims to determine a set of $|M|$ partial routes that serve classes from 1 to k. This is subject to several conditions. Firstly, the total load of each partial route must not exceed the vehicle capacity. Secondly, the hierarchy levels and routes assigned to the required arcs from classes 1 to $k-1$ remain unchanged. Thirdly, for $1 \leq h \leq k-1$, the maximum completion time of hierarchy level h on each route should not exceed C_h. The first condition is straightforward. The second condition ensures that hierarchy level k marks the completion of class k on every route, even though some required arcs in class k are allowed to be serviced at any hierarchy level between 1 and $k-1$. This restriction may increase the value of the hierarchical objective, but it significantly reduces the solution space, thereby improving the running time of MB2-U. Additionally, the third constraint is introduced to achieve the hierarchical objective of the HDCARP-U.

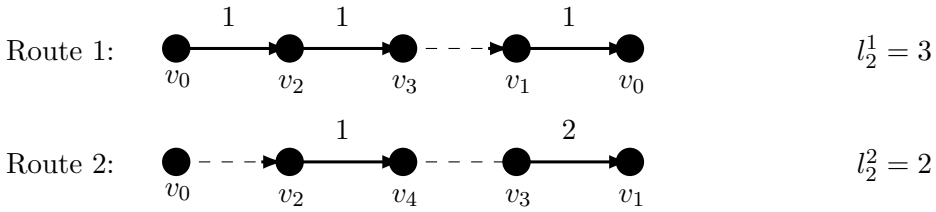
It can be seen that the kth sub-problem not only assigns the required arcs in class k to the existing partial routes but also provides flexibility in rearranging the order of required arcs within each class on existing partial routes. Similar to the HDCARP-P, this sub-problem has two objective functions. The primary objective is to minimise the maximum completion time T_k for class k, while the secondary objective is to minimise the total travel time of the partial routes.

Figure 7 shows the solutions for two sub-problems of the HDCARP instance in Figure 1. Figure 7(a) presents one feasible solution after solving the second sub-problem. It contains two partial routes servicing all required arcs in $A_r^1 \cup A_r^2$. The first partial route currently ends at node v_0 and carries a load of $l_2^1 = 3$ after $t_1^1 = 4$, while the second partial route ends at node v_1 and carries a load of $l_2^2 = 2$ after $t_1^2 = 4$. In this solution, the completion time for both class 1 and class 2 is 4 or $T_1 = T_2 = 4$. Figure 7(b) illustrates one feasible solution for the third sub-problem. By servicing the required arc v_4, v_3 of class 3 before class 2, the maximum completion time of class 3 is only 5, while the maximum completion time of class 2 remains the same.

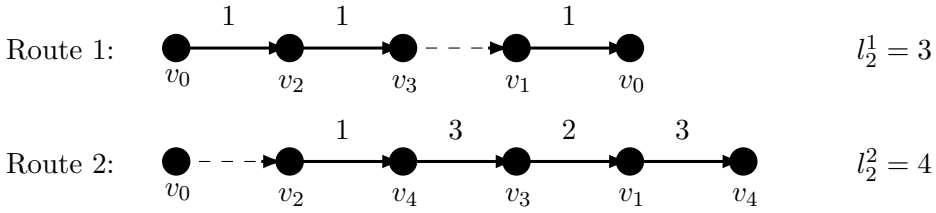
Before presenting the formulation, let us introduce some variables.

- For $h \in \{1, \cdots, k\}$, $m \in M$, and $a \in A_r^1 \cup \cdots \cup A_r^k$, let $x_{ah}^m \in \{0, 1\}$ be a variable taking value of 1 if and only if vehicle m serves the required arc a in hierarchy h.
- For $h \in \{1, \cdots, k\}$, $m \in M$, and $a \in A$, let y_{ah}^m be a variable representing the number of times that vehicle m deadheads the arc a in hierarchy h.
- For $m \in M$ and $h \in \{1, \cdots, k\}$, let t_h^m be a variable representing the time at which the vehicle m finishes servicing required arcs in hierarchy h.
- For $m \in M$, let $r^m \in \{0, 1\}$ be a variable indicating whether vehicle m services any arc in class k.

The formulation of sub-problem k is then as follow:



a, Two partial routes after solving the second sub-problem



b, Two partial routes obtained after solving the third sub-problem

Figure 7: An illustration of sub-problems for the HDCARP-U instance in Figure 1.

$$\text{lex-min} \quad T^k, \sum_{m\in M} t_k^m \tag{50}$$

$$\text{subject to} \quad \sum_{a\in A_r^k} x_{ak}^m \le |A_r^k| r^m \qquad m \in M \tag{51}$$

$$T^k \ge t_k^m - N(1-r^m) \qquad m \in M \tag{52}$$

$$C_h \ge t_h^m \qquad m \in M; h = 1, ..., k-1 \tag{53}$$

$$x_{ah}^m = 1 \qquad m \in M; h = 1, ..., k-1; a \in A_r^{hm} \tag{54}$$

$$t_{h+1}^m - t_h^m = \sum_{a\in A_r^{h+1,m}} s_a + \sum_{a\in A_r^k} s_a x_{a,h+1}^m + \sum_{a\in A} d_a y_{a,h+1}^m \qquad m \in M; h = 1, ..., k-1 \tag{55}$$

$$y_{(v_0' v_0)1}^m = 1 \qquad m \in M \tag{56}$$

$$\sum_{a\in A_f} y_{ah}^m = 1 \qquad m \in M; h = 1, ..., k-1 \tag{57}$$

$$y_{\{v_0', v_i\},h+1}^m = y_{\{v_i, v_0'\},h}^m \qquad m \in M; h = 1, ..., k-1; v_i \in V_t \cup \{v_0\} \tag{58}$$

$$\sum_{m\in M} \sum_{h=1}^{k} x_{ah}^m = 1 \qquad a \in A_r^k \tag{59}$$

$$\sum_{h=1}^{k} \sum_{a\in A_r^k} q_a x_{ah}^m \le Q - l_{k-1}^m \qquad m \in M \tag{60}$$

$$\sum_{a\in\delta_r^+(v)} x_{ah}^m + \sum_{a\in\delta^+(v)} y_{ah}^m = \sum_{a\in\delta_r^-(v)} x_{ah}^m + \sum_{a\in\delta^-(v)} y_{ah}^m \qquad m \in M; h = 1, ..., k; v \in V' \tag{61}$$

$$\sum_{a\in\delta_r^+(S)} x_{ah}^m + \sum_{a\in\delta^+(S)} y_{ah}^m \ge x_{bk}^m \qquad m \in M; h = 1, ..., k; S \subseteq V \setminus \{v_0\}; b \in A_r(S) \tag{62}$$

$$x_{ah}^m \in \{0, 1\} \qquad m \in M; h = 1, ..., k; a \in A_r \tag{63}$$

$$y_{ah}^{m} \in \mathbb{N} \qquad m \in M; h = 1, ..., k; a \in A' \tag{64}$$
$$t_{h}^{m} \in \mathbb{R}^{+} \qquad m \in M; h = 1, ..., k \tag{65}$$
$$r_{m} \in \{0, 1\} \qquad m \in M \tag{66}$$
$$T^{k+1} \in \mathbb{R}^{+} \tag{67}$$

Constraints (51) indicate whether route m services any required arcs in class k. If this is the case, constraints (52) define the maximum completion time of class k. Constraints (53) set an upper bound on the servicing time of each vehicle for each hierarchy level from 1 to $k-1$. Constraints (54) guarantee that required arcs in A_r^{hm} must be serviced in hierarchy h by vehicle m. Constraints (55) define the completion time of hierarchy level $h+1$ ($1 \leq h \leq k-1$) on each route, given that the set $A_r^{h+1,m}$ could be extended by including some required arcs in A_r^k. The objective function and the remaining constraints are trivial.

The MB2-U matheursitic can be adapted to the HDCARP-P by omitting the secondary objective in each sub-problem. Due to the linear precedence relations in the HDCARP-P, we do not allow the required arcs of class k to be serviced in any hierarchy $h \leq k-1$. To do so, we simply modify the range of hierarchy levels considered in constraints (55) and (59) in the MILP formulations. Furthermore, after solving the $k-1$ sub-problems, we only need to track which required arcs of class $k-1$ were serviced on which routes, rather than all classes from 1 to $k-1$. Subsequently, the kth sub-problem aims to assign required arcs of class k to existing partial routes while allowing required arcs of class $k-1$ to be rearranged. This approach reduces the deadheading time connecting class $k-1$ and class k, potentially resulting in a reduction in T^k while keeping T^{k-1} unchanged. The resulting methodology is referred to as *MB2-R*.

5. Computational Experiments

In this section, we present the results of some computational experiments. To fairly compare the performance of the different approaches, we set the time limit of an hour for solving the full MILP-U or MILP-R models in one pass, and 10 minutes for solving each sub-problem in matheuristic MB1 or MB2 for every instances. For all of our computational experiments, we used a laptop with an AMD Ryzen 7 3700X Processor at 3.6GHz with 32GB RAM. All algorithms were implemented in C/C++.

5.1. Instance generation

Now, we explain how we created our artificial HDCARP instances to closely imitate real road network structures. Note that road networks are often planar and, even if not, they could be made planar by deleting a very small number of road segments. In most of planar road networks, the average number of road segments incident to each node usually ranges from 1.5 to 3 [7, 6]. Moreover, distances in road network can be reasonably approximated by multiplying Euclidean distances with a suitable constant [5].

Our procedure to construct the graph $G = (V, A)$ is inspired by methods introduced in [23, 27], as follows:

- We generate randomly n vertices inside a unit square, where n is a multiple of 10 between 10 and 100. We denote the set of these vertices by V. These vertices have known coordinates. The first vertex is set to be the depot.
- Then, we construct the minimum Hamiltonian circuit passing through each node exactly once. We add all arcs in the giant circuit to the set A according to the direction in which they are traversed. The resulting graph G is now strongly connected.

- Additional arcs are introduced randomly into the graph following these conditions: (1) The total number of arcs is equal to nd ($|A| = nd$), where $d \in \{1.5, 2, 2.5, 3\}$ represents the average number of arcs incident on each node in graph G; (2) The length of the added arcs are not excessively long; (3) None of the arcs intersect or cross each other.
- For each combination of n and d, five different version of graphs were generated.
- The arc set A is randomly divided into four separate sets: A_r^1, A_r^2, A_r^3, and A_{nr}. Each of the first three sets contains $\left\lfloor \frac{|A|}{4} \right\rfloor$ required arcs and corresponds to priority classes. The last set, A_{nr}, consists of non-required arcs. This means $A_r = A_r^1 \cup A_r^2 \cup A_r^3$ and $P = \{1, 2, 3\}$.
- The cost of each arc a is calculated as the Euclidean distance multiplied by a constant factor of 100. This cost is denoted as $tCost_a$.
- The deadheading cost d_a of each arc $a \in A$ is set as $\max 1, [tCost_a + 0.5]$ to ensure it is always at least 1, regardless of the value of $tCost_a$.

  each arc $a \in A_r$, its demand q_a is calculated as $\lfloor d_a \times 0.5 + 0.5 \rfloor$, and its servicing cost s_a is set to twice its deadheading cost, i.e., $s_a = 2d_a$.
- There are 3 homogeneous vehicles with capacity $Q = \sum_{a \in A_r} q_a/3 + 0.5$, based at a depot.

We name the created instances in the form *harp-n-m_i*, where n is the number of vertices, m is the number of arcs, and $i(= 1, ..., 5)$ is the instance's version. For example, harp-50-75_2 shows the second instance with 50 vertices and 75 arcs. Detailed benchmark instances and results of our experiments are available at http://orlab.com.vn/home/download.

5.2. Experimental results on the small instances

Instance		MILP-R			MB1				MB2-R						
|V|	|A|	#opt	Gap	Time	B	E	W	Time	B	E	W	B_{MB1}	E_{MB1}	W_{MB1}	Time
10	15	5	0.00	0.13	0	4	1	0.18	0	4	1	0	5	0	0.08
10	20	5	0.00	0.28	0	2	3	0.19	0	3	2	1	4	0	0.10
10	25	5	0.00	0.52	0	2	3	0.20	0	3	2	1	4	0	0.11
10	30	5	0.00	1.26	0	3	2	0.18	0	2	3	1	1	3	0.13
20	30	5	0.00	0.82	0	1	4	0.20	0	2	3	1	1	3	0.12
20	40	5	0.00	2.93	0	4	1	0.25	0	3	2	0	4	1	0.18
20	50	5	0.00	6.15	0	3	2	0.29	0	4	1	1	3	1	0.27
20	60	5	0.00	77.20	0	5	0	0.45	0	4	1	0	4	1	0.49
30	45	5	0.00	1.65	0	3	2	0.24	0	3	2	0	4	1	0.16
30	60	5	0.00	7.36	0	4	1	0.35	0	1	4	0	2	3	0.33
30	75	5	0.00	120.15	0	4	1	0.87	0	3	2	1	3	1	0.88
30	90	4	0.30	949.25	1	0	4	10.58	1	1	3	2	2	1	6.64
40	60	5	0.00	8.50	0	3	2	0.33	0	3	2	1	4	0	0.26
40	80	5	0.00	69.93	0	3	2	0.66	0	4	1	1	4	0	0.78
40	100	3	0.59	1943.27	0	5	0	7.52	0	4	1	0	4	1	5.38
40	120	1	2.89	3594.37	3	0	2	75.39	2	1	2	2	0	3	44.50
50	75	5	0.00	17.86	0	2	3	0.46	0	2	3	0	5	0	0.44
50	100	5	0.00	220.37	0	1	4	1.79	0	1	4	1	3	1	1.85
50	125	3	3.94	2761.86	2	2	1	11.98	2	2	1	1	2	2	10.28
50	150	1	6.22	2578.93	4	0	1	157.17	4	0	1	3	2	0	50.01

Table 2: HDCARP-P's results on small instances

This subsection starts with comparing the exact algorithm presented in Section 3 with two metaheuristics, MB1 and MB2-R, for the HDCARP-P on 100 instances with up to 50 nodes. The summarised results can be found in Table 2. The first two columns indicate the number of vertices and arcs in the instances. For the exact algorithm, we show the number of instances optimally solved (out of 5 versions) (opt), as well as the average percent gap (Gap) between the upper and lower bounds obtained using our branch-and-cut algorithm. In the case of each metaheuristic, we present the number of instances where it produces better (B), equal (E), or worse (W) solutions compared to the exact algorithm. Furthermore, we are interested in comparing the two metaheuristics. Hence, we show the number of instances in which MB2-R performs better (B_{MB1}), the same (E_{MB1}), or worse (W_{MB1}) than MB1. Finally, for each approach, we provide the average running time (Time) in seconds.

The exact method solves optimally 87 out of 100 instances, including nearly all instances with fewer than 100 arcs and 8 out of 10 instances with 100 arcs. For the remaining instances, the optimal gap never exceeds 6.22%. It is worth noting that our MILP formulation successfully captures a special case highlighted in Subsection 3.1. For instance, in the optimal solution of the harp-10-15_5 instance, the maximum completion times for classes 1, 2, and 3 are 249, 444, and 441, respectively.

The MB1 and MB2-R matheuristics have found solutions that are at least as good as the exact algorithm for 61 and 59 instances, respectively. Closer inspection of the output showed that the improved solutions obtained by the MB1 and MB2-R matheuristics were observed in instances with a minimum of 90 arcs. Additionally, the running time of both matheuristics was significantly shorter compared to the exact algorithm. These findings suggest that the MB1 and MB2-R matheuristics are especially promising for large-scale instances.

The MB2-R outperforms the MB1 in terms of solution quality for 17 instances. This confirms that the MB2-R has succeeded in reducing deadheading time between consecutive classes on each route. Additionally, the MB2-R runs faster than the MB1 in almost all instances. A possible explanation for this is that the MB2-R has a single objective function in each subproblem, while the MB1 has a bi-objective function.

Next, we compare the exact algorithm and the matheuristic (MB2-U) for the HDCARP-U. The results are summarised in Table 3. The exact algorithm has achieved optimal solutions for 67 instances, with most of them having fewer than 80 arcs. For instances that could not be solved optimally, the MB2 matheuristic successfully produced feasible solutions within a short running time.

In summary, the exact approaches can solve small-scale instances optimally with up to 90 arcs, but they consume significant computational time. On the other hand, the matheuristics provide reasonably good solutions within a short computing time for all instances in this experiment.

5.3. Comparing matheuristics on the large instances

We conducted experiments on larger instances using our matheuristics. These instances range from 60 to 100 nodes. The results are presented in Tables 4 and 5. The first columns of the tables show the name of each instance. The next four columns show the maximum completion time of each class, denoted by T_1, T_2 and T_3, and the running time (in seconds) for MB1. For the MB2-R or MB2-U, we also show the gap in the maximum completion time for each class, namely Gap_1, Gap_2 and Gap_3, and the running time, measured in seconds. For $i \in \{1, 2, 3\}$, the gap Gap_i is calculated as $100 \times (T'_i - T_i)/T_i$, where T'_i and T_i represent the maximum completion times of class i obtained by the MB1 and MB2 (MB2-R or MB2-

Instance		MILP-U			MB2-U			
|V|	|A|	#opt	Gap	Time	B	E	W	Time
10	15	5	0.00	9.90	0	4	1	0.20
10	20	5	0.00	40.57	0	4	1	0.23
10	25	5	0.00	63.07	0	1	4	0.26
10	30	5	0.00	62.59	0	1	4	0.23
20	30	5	0.00	123.43	0	0	5	0.29
20	40	5	0.00	319.64	0	3	2	0.46
20	50	5	0.00	480.07	0	3	2	0.64
20	60	4	0.00	1712.85	1	3	1	0.99
30	45	5	0.00	136.81	0	2	3	0.43
30	60	5	0.00	620.45	0	4	1	0.77
30	75	4	0.83	2591.48	0	5	0	1.77
30	90	1	0.00	1795.47	4	0	1	13.15
40	60	5	0.00	1559.93	0	4	1	0.83
40	80	4	0.00	1507.41	1	3	1	1.67
40	100	0	-	-	5	0	0	11.37
40	120	0	-	-	5	0	0	144.59
50	75	4	0.00	2016.94	1	2	2	1.21
50	100	0	45.89	3600.03	5	0	0	5.51
50	125	0	-	-	5	0	0	26.21
50	150	0	-	-	5	0	0	165.25

Table 3: HDCARP-U's results on small instances

U), respectively. A negative gap indicates improvement gained by using MB2 matheuristic, while a positive gap indicates a deterioration in solution quality. A dash (-) indicates that the solving process either did not terminate within the time limit or encountered memory problems. Instances that can be improved by either MB2-R or MB2-U are highlighted in bold.

The performance of the MB1 and MB2-R matheuristics for the HDCARP-P problem is comparable. The MB1 outperforms MB2-R for 23 instances, while MB2-R improves upon the solutions found by MB1 for 26 instances.

Although MB2-U fails to find feasible solutions for 5 instances, it significantly improves the hierarchical objective for 43 instances compared to the other two matheuristics. This suggests that the HDCARP-U is effective in reducing the completion time of high priority classes. Regarding running times, ther is no obvious pattern.

5.4. Total time comparison of the HDCARP-P and HDCARP-U

To provide a comprehensive analysis, we further investigated the total servicing time of all routes for each HDCARP variant to examine the impact of class upgrading. To conduct this experiment, we used the best solutions obtained from either MB1 or MB2-R for the HDCARP-P and compared them with the solutions from MB2-U for the HDCARP-U.

Figure 8 compares the average total servicing time between both HDCARP variants for different combinations of $|V|$ and d. It can be seen that the HDCARP-U successfully reduces the total servicing time, except for instances with $|V| = 20, d = 2.5$ and $|V| = 40, d = 3.0$.

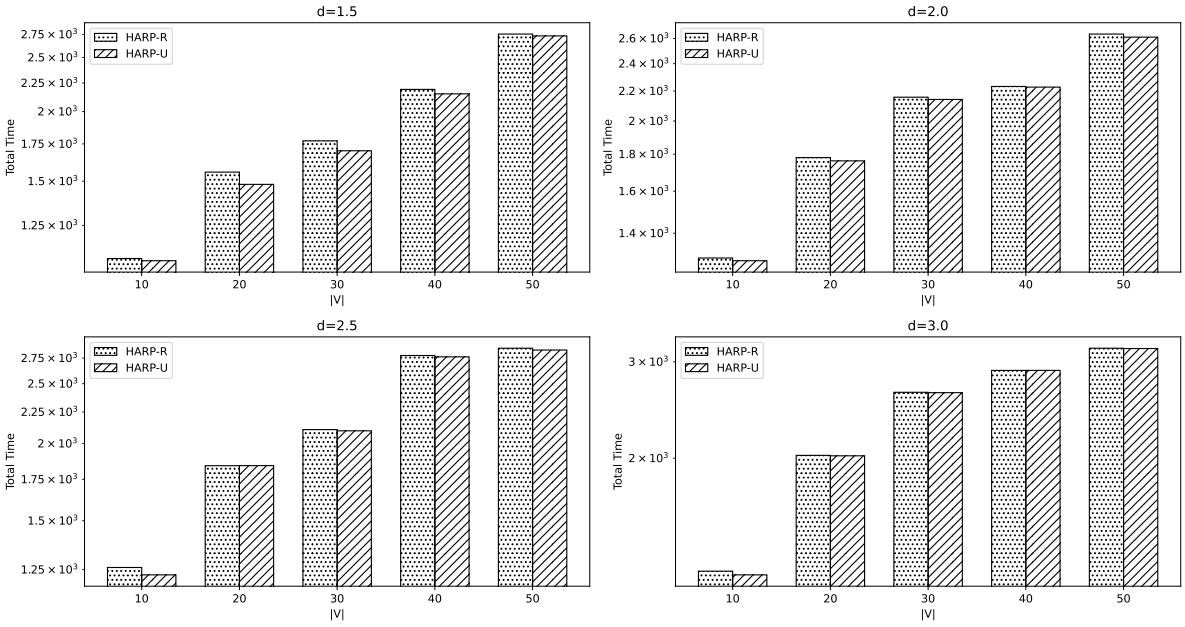


Figure 8: Total time on small instances

Figure 9 shows the same for larger instances. The HDCARP-U only improves the total servicing time compared to the HDCARP-P for $d = 1.5$ and $d = 2.0$. This is because the limitation on the hierarchy level of classes in each sub-problem in MB2-U restricts the solution space. Surprisingly, for $d = 3$ and $|V| \geq 80$, the HDCARP-U performs worse than the HDCARP-P. A possible explanation is that when $|A|$ increases, each sub-problems in MB2-U also increases in size, making it challenging to obtain good feasible solutions.

Instance	MB1				MB2-R				MB2-U			
	T_1	T_2	T_3	Time	Gap_1	Gap_2	Gap_3	Time	Gap_1	Gap_2	Gap_3	Time
harp-60-90_1	455	891	1255	0.39	0	0	0	0.53	0	0.56	-0.48	1.25
harp-60-90_2	478	758	1079	0.48	0	0.40	0.28	0.47	0	0.26	1.48	2.18
harp-60-90_3	525	886	1277	0.52	0	0	0	0.45	0	0	0	1.62
harp-60-90_4	547	1066	1517	0.44	0	0	0	0.44	0	0	-5.14	2.90
harp-60-90_5	507	916	1296	0.62	0	0	0	0.63	0	0	0	1.56
harp-60-120_1	616	1065	1471	7.86	0	0	0	5.54	0	-1.03	-0.34	7.79
harp-60-120_2	466	1132	1750	5.69	0	0	0	5.07	0	0	-0.23	14.92
harp-60-120_3	596	1279	1702	5.58	0	0	0	5.10	0	-1.49	-1.01	14.27
harp-60-120_4	439	916	1322	11.28	0	0	0	8.17	0	-0.33	0.45	14.75
harp-60-120_5	478	870	1278	5.26	0	0	0	5.23	0	0.11	0.08	8.92
harp-60-150_1	579	1114	1584	605.38	0	0	0	607.12	0	0.54	-0.19	627.00
harp-60-150_2	556	1123	1595	10.02	0	0	0.06	11.14	0	0	-0.13	15.25
harp-60-150_3	548	1128	1714	76.02	0	0	0	25.17	0	0	0	148.57
harp-60-150_4	456	1090	1591	17.77	0	-0.09	-0.06	7.47	0	-0.09	-0.06	18.14
harp-60-150_5	621	1131	1714	42.53	0	0	0	25.66	0	0	0	47.08
harp-60-180_1	695	1310	1876	1800.43	0	3.05	0.59	1231.67	-0.43	0	-0.05	1216.37
harp-60-180_2	593	1186	1788	1103.22	0	0	0.34	1154.68	0	0	-0.56	675.48
harp-60-180_3	630	1219	1775	227.71	0	0	0	131.18	0	0	0	125.78
harp-60-180_4	593	1202	1689	241.14	0	0	-0.06	131.51	0	0	0	306.01
harp-60-180_5	573	1214	1833	1265.09	0	-1.65	-1.09	659.53	0	-1.65	-1.09	684.91
harp-70-105_1	531	998	1580	1.04	0	0	0	2.25	0	0	0	4.25
harp-70-105_2	547	1120	1555	0.75	0	0	0	0.57	0	-5.45	-3.92	2.12
harp-70-105_3	492	1018	1515	0.81	0	0.39	0.20	0.83	0	0.10	0.20	3.01
harp-70-105_4	525	946	1491	0.48	0	0	0	0.54	0	-0.21	-10.13	1.19
harp-70-105_5	537	966	1281	0.44	0	0	0.08	0.64	0	-1.76	1.56	1.99
harp-70-140_1	682	1223	1680	14.33	0	0	-0.42	8.96	0	0	-0.42	12.52
harp-70-140_2	425	831	1272	11.02	0	0	0	12.30	0	0	-0.08	22.88
harp-70-140_3	580	1162	1743	6.2	0	0	0	8.78	0	0	-0.98	10.68
harp-70-140_4	469	974	1460	3.03	0	8.01	6.23	10.67	0	0	-2.05	10.09
harp-70-140_5	522	980	1509	1.9	0	0	0	3.07	0	0	-0.13	6.76
harp-70-175_1	595	1151	1816	22.55	0	0	0	47.76	0	0	0	19.79
harp-70-175_2	527	1021	1568	12.37	0	0	0	35.22	0	0	0	23.37
harp-70-175_3	523	995	1601	194.68	0	0	0.06	213.13	0	-0.60	-0.37	128.73
harp-70-175_4	620	1233	1767	120	0	0	0	60.22	0	0.08	-1.98	121.02
harp-70-175_5	565	1092	1559	18.57	0	0	0	15.49	0	-2.93	-0.58	36.29
harp-70-210_1	691	1419	2332	612.82	0	0	0	612.27	0	0	1.07	628.24
harp-70-210_2	633	1242	1896	519.66	0	0	0	222.21	0	-0.16	-0.37	439.60
harp-70-210_3	639	1226	1792	661.27	0	0	0	639.36	-0.16	0	-0.50	746.29
harp-70-210_4	576	1147	1911	1214.75	0	-0.35	-4.66	1242.09	1.04	-0.70	-6.44	1233.83
harp-70-210_5	719	1355	2017	271.48	0	0	0.05	343.04	0	0	0.05	342.03
harp-80-120_1	692	1368	1768	2.78	0	0	0.85	2.70	0	0	-2.60	8.33
harp-80-120_2	649	1159	1592	4.48	0	0	0.06	4.81	0	-0.86	-4.27	3.43
harp-80-120_3	556	990	1467	4.08	0	0	0	4.05	0	0	0	18.63
harp-80-120_4	576	967	1450	1.17	0	0	-0.14	2.07	0	-1.14	-2.34	4.32
harp-80-120_5	520	1073	1574	2.81	0	0	0	5.08	0	0	0	7.48
harp-80-160_1	574	1067	1711	46.64	0	0	0	28.40	0	0	0	38.93
harp-80-160_2	644	1109	1686	3.81	0	-0.36	-0.18	3.71	0	-0.36	-2.43	7.12
harp-80-160_3	528	1032	1553	610.31	0	-0.10	-0.26	290.73	0	0	-0.06	613.60
harp-80-160_4	589	1061	1586	611.35	0	0	-0.13	379.28	0	0.19	-6.68	620.23
harp-80-160_5	575	1047	1513	32.92	0	0	0	28.39	0	-2.39	-1.65	34.90

Table 4: The results of matheuristics on large instances

Instance	MB1				MB2-R				MB2-U			
	T_1	T_2	T_3	Time	Gap_1	Gap_2	Gap_3	Time	Gap_1	Gap_2	Gap_3	Time
harp-80-200_1	683	1360	1975	724.59	0	6.40	2.33	955.16	0	0.22	0.61	1140.40
harp-80-200_2	807	1391	2052	737.04	0	1.22	0.58	1293.48	-	-	-	-
harp-80-200_3	634	1276	1874	29.89	0	0	0	33.05	0	0	-0.75	91.01
harp-80-200_4	624	1142	1652	209.45	0	0	0	159.57	0	0	0.06	244.39
harp-80-200_5	880	1483	2232	614.66	0	0	0	644.12	0	0.07	0.04	623.67
harp-80-240_1	651	1406	2006	1200.63	0	-0.36	-0.25	916.37	0	-0.36	0.55	915.60
harp-80-240_2	614	1222	1819	730.98	0	0	0.49	675.93	0	0	0.27	984.68
harp-80-240_3	713	1371	2122	1335.51	0	-1.31	0.42	1252.04	8.98	1.60	0.71	1804.57
harp-80-240_4	758	1360	2059	682.21	0	-1.84	-1.46	943.04	5.28	4.63	-2.96	1766
harp-80-240_5	675	1318	2094	641.22	0	0	0	137.72	0	-0.15	-0.05	654.93
harp-90-135_1	507	956	1325	1.62	0	-4.29	-3.85	2.21	0	-7.95	-7.02	10.81
harp-90-135_2	610	1160	1642	16.8	0	0	-0.30	17.64	0	0	-0.30	38.45
harp-90-135_3	655	1109	1570	600.97	0	0	0	605.97	0	0	-0.06	606.66
harp-90-135_4	571	1045	1548	2.81	0	0	0	3.04	0	-7.27	-3.62	10.69
harp-90-135_5	472	1062	1665	7.12	0	-7.06	-6.61	4.90	0	-1.13	-0.54	26.26
harp-90-180_1	666	1166	1800	119.42	0	0	-0.06	47.12	0	-3.09	-3.72	232.50
harp-90-180_2	631	1135	1705	615.07	0	0	0	612.30	0	0	0	628.46
harp-90-180_3	564	1109	1748	38.05	0	0	0	25.66	0	-2.52	-1.03	59.62
harp-90-180_4	608	1129	1612	12.21	0	0	0	14.82	0	-2.21	-0.87	22.19
harp-90-180_5	674	1299	1860	10.33	0	0	0	9.18	0	-0.31	-0.86	19.81
harp-90-225_1	615	1233	1837	639.10	0	0	0	732.47	-3.90	-7.38	-5.17	836.49
harp-90-225_2	663	1267	1925	112.32	0	0	0	31.19	0	0	0	89.87
harp-90-225_3	708	1454	2110	1573.62	0	-0.28	-2.09	1571.84	0	-0.41	-3.79	1067.52
harp-90-225_4	817	1461	2066	1801.67	0	-4.52	-5.18	1199.88	-3.30	-1.03	3.58	1800.60
harp-90-225_5	648	1449	2176	702.47	0	0	0.64	680.82	0	0	0	750.80
harp-90-270_1	717	1423	2374	1207.83	0	-0.14	-4.68	1239.75	2.79	0.42	-5.81	1802.97
harp-90-270_2	894	1610	2413	1472.61	0	-1.93	-2.86	892.05	-	-	-	-
harp-90-270_3	728	1421	2073	1059.94	0	-0.28	-0.24	1218.06	0.41	-0.35	-0.43	992.00
harp-90-270_4	727	1443	2299	1802.06	0	26.96	24.88	1801.49	-	-	-	-
harp-90-270_5	753	1471	2236	1218.44	0	0	1.92	1386.65	0	0	7.38	1409.45
harp-100-150_1	548	1146	1691	0.87	0	0	0.18	1.64	0	-0.09	-0.83	3.42
harp-100-150_2	593	1163	1677	12.93	0	0	0	12.35	0	0	0	11.57
harp-100-150_3	653	1543	1998	0.58	0	0	0	0.90	0	0	0	4.21
harp-100-150_4	644	1178	1857	439.12	0	0	0	724.60	0	0	-0.70	338.14
harp-100-150_5	645	1091	1686	2.04	0	0	0	3.40	0	-0.37	2.49	12.16
harp-100-200_1	770	1638	2555	1070.48	0	0.24	4.89	1202.26	0	-6.11	-3.41	1202.48
harp-100-200_2	671	1378	1956	685.72	0	0	0	672.551	0	0	0	644.87
harp-100-200_3	635	1200	1764	14.30	0	0	0	19.03	0	0	0	18.02
harp-100-200_4	623	1254	1840	640.72	0	-0.08	0.92	696.89	0	1.83	01.68	780.97
harp-100-200_5	675	1341	1990	50.77	0	0	0	39.35	0	0	0	77.39
harp-100-250_1	657	1223	1893	213.83	0	0	0.85	206.04	0	-0.65	0.37	404.87
harp-100-250_2	671	1245	1969	525.85	0	0	0	865.56	1.19	0.80	0.51	960.02
harp-100-250_3	693	1312	2088	980.74	0	0.08	1.01	866.94	0	-0.53	3.69	768.66
harp-100-250_4	742	1509	2187	1147.68	0	3.05	-2.79	1418.90	0	13.59	-5.08	1268.54
harp-100-250_5	740	1380	2102	655.52	0	0	-0.10	637.92	0	0	-0.14	688.47
harp-100-300_1	766	1551	2345	1642.50	0	0	2.17	1399.34	7.44	3.68	7.08	1615.73
harp-100-300_2	763	1578	2298	490.36	0	0	0	901.32	2.62	-0.51	0.78	1377.95
harp-100-300_3	842	1582	2446	1241.17	0	-1.33	0.29	1267.36	-0.24	-2.09	-0.61	1612.58
harp-100-300_4	863	1978	3002	1265.71	0	-4.65	-6.53	1801.09	-	-	-	-
harp-100-300_5	832	1667	2466	1398.80	0	-0.06	-0.08	1328.52	-	-	-	-

Table 5: The results of matheuristics on large instances (continued)

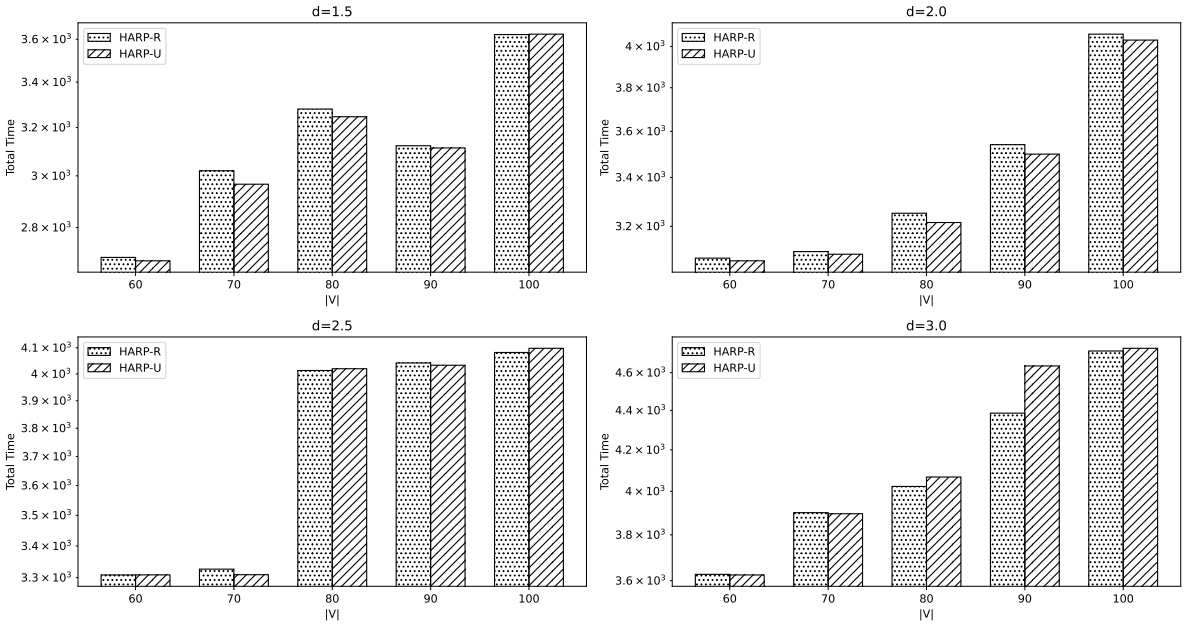


Figure 9: Total time on large instances

6. Conclusions

In this paper, we have studied the Hierarchical Directed Capacitated Arc Routing Problem (HDCARP) and its variants with and without class upgrading. We proposed the mathematical models and matheuristics for both variants and conducted extensive computational experiments. with and without class upgrading. The results indicate that the exact methods are only suitable for small instances, while the proposed matheuristics work rather well in most cases. Furthermore, a comparison between the two variants was performed to evaluate the impact of class upgrading option. It was found that allowing class upgrading successfully improves the completion time of high priority classes while slightly affecting the total servicing time in certain cases.

Our approaches could be fairly easily adapted to other HDCARPs, such as HDCARPs with multiple depots, heterogeneous vehicles or HDCARPs on mixed graphs.

We can think of three possible topics for future research. The first topic involves the development of fast heuristics to efficiently handle real-world instances because matheuristics proposed in this study are still time-consuming. The second is the development of a local search heuristic to further improve the solutions. Finally, the third topic involves the development of a matheuristic approach specifically designed for the HDCARP-U variant to improve the total completion time without any restrictions.

Acknowledgement

We thank Le Ba Luat (Phenikaa University) for performing a part of experiments. The manuscript of the paper was finished during the research stay of the authors Minh Hoàng Hà and Trung Thanh Nguyen at the Vietnamese Institute for Advanced Studies in Mathematics (VIASM). We wish to thank this institution for their kind hospitality and support.

References

[1] V.A. Afanasev, R. van Bevern, and O.Y. Tsidulko. The hierarchical chinese postman problem: the slightest disorder makes it hard, yet disconnectedness is manageable. *Operations Research Letters*, 49(2):270–277, 2021.

[2] C. Ahabchane, A. Langevin, and M. Trépanier. Robust optimization for the hierarchical mixed capacitated general routing problem applied to winter road maintenance. *Computers & Industrial Engineering*, 158:107396, 05 2021.

[3] C. Ahabchane, M. Trépanier, and A. Langevin. Street-segment-based salt and abrasive prediction for winter maintenance using machine learning and GIS. *Transactions in GIS*, 23(1):48–69, 2019.

[4] A.S. Alfa and D.Q. Liu. Postman routing problem in a hierarchical network. *Engineering Optimization*, 14(2):127–138, 1988.

[5] B. Boyacı, T.H. Dang, and A.N. Letchford. Vehicle routing on road networks: how good is Euclidean approximation? *Computers & Operations Research*, 129:105197, 2021.

[6] B. Boyacı, T.H. Dang, and A.N. Letchford. On matchings, T-joins and arc routing in road networks. *Networks*, 79(1):20 31, 2022.

[7] B. Boyacı, T.H. Dang, and A.N. Letchford. Fast upper and lower bounds for a large-scale real-world arc routing problem. *Networks*, 81(1):107–124, 2023.

[8] E.A. Cabral, M. Gendreau, G. Ghiani, and G. Laporte. Solving the hierarchical chinese postman problem as a rural postman problem. *European Journal of Operational Research*, 155(1):44–50, 2004.

[9] J.F. Campbell and A. Langevin. Operations management for urban snow removal and disposal. *Transportation Research Part A: Policy and Practice*, 29(5):359–370, 1995.

[10] N. Christofides. The optimum traversal of a graph. *Omega*, 1(6):719–732, 1973.

[11] M.K. Çodur and M. Yılmaz. A time-dependent hierarchical chinese postman problem. *Central European Journal of Operations Research*, pages 1–30, 2018.

[12] M. Colombi, A. Corberán, R. Mansini, I. Plana, and J.M. Sanchis. The hierarchical mixed rural postman problem. *Transportation Science*, 51(2):755–770, 2017.

[13] M. Colombi, A. Corberán, R. Mansini, I. Plana, and J.M. Sanchis. The hierarchical mixed rural postman problem: Polyhedral analysis and a branch-and-cut algorithm. *European Journal of Operational Research*, 257(1):1–12, 2017.

[14] Á. Corberán, R. Eglese, G. Hasle, I. Plana, and J. M. Sanchis. Arc routing problems: A review of the past, present, and future. *Networks*, 77(1):88–115, 2021.

[15] P. Damodaran, M. Krishnamurthi, and K. Srihari. Lower bounds for hierarchical chinese postman problem. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 15(1):36–44, 2008.

[16] M. Dror, H. Stern, and P. Trudeau. Postman tour on a graph with precedence relation on arcs. *Networks*, 17(3):283–1294, 1987.

[17] J. Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards B*, 69:125–130, 1965.

[18] J. Edmonds and E.L. Johnson. Matching, Euler tours and the Chinese postman. *Mathematical Programming*, 5(1):88–124, 1973.

[19] H.A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part i: The Chinese postman problem. *Operations Research*, 43(2):231–242, 1995.

[20] É. Gélinas. *Le probleme du postier chinois avec contraintes générales de préséance. M. Sc. A*. PhD thesis, Dissertation, École Polytechnique de Montréal, Canada, 1992.

[21] G. Ghiani and G. Improta. An algorithm for the hierarchical chinese postman problem. *Operations Research Letters*, 26(1):27–32, 2000.

[22] M.G. Guan. Graphic programming using odd or even points. *Chinese Mathematics*, 1:273–277, 1962.

[23] M.H. Ha, N. Bostel, A. Langevin, and L.M. Rousseau. Solving the close-enough arc routing problem. *Networks*, 63(1):107–118, 2014.

[24] E. Haslam and J.R. Wright. Application of routing technologies to rural snow and ice control. *Transportation Research Record*, 1304, 1991.

[25] P. Korteweg and T. Volgenant. On the hierarchical chinese postman problem with linear ordered classes. *European Journal of Operational Research*, 169(1):41–52, 2006.

[26] P.F. Lemieux and L. Gampagna. The snow ploughing problem solved by a graph theory algorithm. *Civil Engineering Systems*, 1(6):337–341, 1984.

[27] L.C. Lu, M.H. Ha, A. Langevin, and M. Gendreau. Optimizing road network daily maintenance operations with stochastic service and travel times. *Transportation Research Part E: Logistics and Transportation Review*, 64:88–102, 2014.

[28] U. Manber and S. Israni. Pierce point minimization and optimal torch path determination in flame cutting. *Journal of Manufacturing Systems*, 3(1):81–89, 1984.

[29] N. Perrier, A. Langevin A.Ãmaya, and G.Cormier. Improving snow removal operations using operations research: A case study. In *Proceedings of International Conference on Information Systems, Logistics and Supply Chain*. INSA de LYON Lyon, France, 2006.

[30] N. Perrier, A. Langevin, and C.A. Amaya. Vehicle routing for urban snow plowing operations. *Transportation Science*, 42(1):44–56, 2008.

[31] N. Perrier, A. Langevin, and J.F. Campbell. A survey of models and algorithms for winter road maintenance. part I: system design for spreading and plowing. *Computers & Operations Research*, 33(1):209–238, 2006a.

[32] N. Perrier, A. Langevin, and J.F. Campbell. A survey of models and algorithms for winter road maintenance. part II: system design for snow disposal. *Computers & Operations Research*, 33(1):239–262, 2006b.

[33] N. Perrier, A. Langevin, and J.F. Campbell. A survey of models and algorithms for winter road maintenance. part III: Vehicle routing and depot location for spreading. *Computers & Operations Research*, 34(1):211–257, 2007a.

[34] N. Perrier, A. Langevin, and J.F. Campbell. A survey of models and algorithms for winter road maintenance. part IV: Vehicle routing and fleet sizing for plowing and snow disposal. *Computers & Operations Research*, 34(1):258–294, 2007b.

[35] O. Quirion-Blais, A. Langevin, F. Lehuédé, O. Péton, and M. Trépanier. Solving the large-scale min-max K-rural postman problem for snow plowing. *Networks*, 70, 08 2017.

[36] O. Quirion-Blais, M. Trépanier, and A. Langevin. A case study of snow plow routing using an adaptive large hood search metaheuristic. *Transportation Letters*, 7:201–209, 09 2015.

[37] U.B. Sayata and N.P. Desai. An algorithm for hierarchical chinese postman problem using minimum spanning tree approach based on kruskal's algorithm. In *2015 IEEE International Advance Computing Conference (IACC)*, pages 222–227. IEEE, 2015.

[38] J.L. Sullivan, J. Dowds, D.C. Novak, D.M. Scott, and C. Ragsdale. Development and application of an iterative heuristic for roadway snow and ice control. *Transportation Research Part A: Policy and Practice*, 127:18–31, 2019.

[39] Jin-Yuan Wang and Jeff R Wright. Interactive design of service routes. *Journal of Transportation Engineering*, 120(6):897–913, 1994.