RAIRO-Oper. Res. 59 (2025) 163–191
https://doi.org/10.1051/ro/2024116

RAIRO Operations Research
www.rairo-ro.org

ROUND-TRIP HUB LOCATION PROBLEM

Omar Kemmar[1,*], Karim Bouamrane[1] and Shahin Gelareh[2]

Abstract. In this paper, we introduce a novel network design for the Hub Location Problem, inspired by the round-trip structure commonly used by transport service providers. Our design integrates spoke nodes assigned to a central hub node, creating round-trips where the hub node serves as the starting point, visits all assigned spoke nodes, and returns to the hub. To enhance transportation services and provide additional redundancy, we introduce a new type of nodes called runaway nodes to the network. The motivation for this research arises from two real-life cases encountered during consultancy projects, underscoring the necessity for an optimized network design in transportation services. To address the proposed problem, we introduce a mixed-integer linear programming (MIP) mathematical model. However, due to the problem's complexity, the feasibility of the MIP model is limited to small-scale instances. To tackle medium and large-scale instances, we introduce two hyper-heuristic approaches based on reinforcement learning. These hyper-heuristic approaches harness the power of reinforcement learning to guide the selection of low-level heuristics and improve solution quality. We conduct extensive computational experiments to evaluate the efficiency and effectiveness of the proposed approaches. The results of our experiments affirm the efficiency of the proposed hyper-heuristic approaches, showcasing their ability to discover high-quality solutions for the Hub Location Problem.

Mathematics Subject Classification. 68T20, 90C59, 90C27, 90B80, 90C35, 90C05, 90C11.

Received March 6, 2022. Accepted May 22, 2024.

1. Introduction

The Hub Location problem (HLP), underlies almost all the transportation networks including supply chain logistics, telecommunications, urban transportation, and postal delivery. Generally speaking, hub-and-spoke structures, like the one depicted in Figure 1a, are the foundation of transportation networks. Throughout our expertise in the sector, we dealt with a variety of networks of a different type where transportation is provided *via* round-trips rather than directly from the hub node to each spoke node (see Fig. 1b). The problem encountered is that in many cases this kind of structure does not correspond to the real needs and therefore must be improved.

Our expertise in the industry over the past five years, during which we completed numerous consulting projects, served as the inspiration for our work. Following, we present two examples from the industry where the structure Figure 1b has reached its limits.

Keywords. Hub location problem, liner shipping, runaway node, branch-and-cut, hyper-heuristic, reinforcement learning, Q-learning, A-learning.

[1] Laboratoire d'informatique d'Oran (LIO), Université Oran 1, BP 1524 EL Mnaouer Oran, Algeria.
[2] Département Réseaux et Télécommunications, Université d'Artois, F-62400 Béthune, France.
*Corresponding author: Omke1941@hotmail.com; kemmar.omar@edu.univ-oran1.dz

(a) **(b)**

FIGURE 1. Basic and round-trip HLP networks. (a) A basic hub-and-spoke service network. (b) A typical round-trip hub-and-spoke service network.

Liner shipping industry. Liner shipping is the basis of world trade. As world economic activities intensify, according to [34], about 80% of the world trade by volume and more than 70% of world trade in value is carried by sea. The liner shipping network design problem get a lot of interest from researchers. When the mega-vessels such as Emma Maersk of 18 000 TEU (twenty-foot equivalent) have emerged and deployed, the string managers who are in charge of designing seasonal round-trip service network in shipping companies, had some difficulties to exploit this such vessels and were worried that this vessels which are expensive to maintain to be underused (at least during some periods) without being able to benefits from their economies of scale. A much higher problem has been encountered by port operators, which is the risk of disruption. If a disruption happens and the delays go beyond the expected time it will cause dissatisfaction and cost a lot of money for both liners and ports due to the draught requirement. Moreover, many European ports did not have the capacity (not even potential for physical expansion) to serve this huge vessels in case of problem or disruption. One of the solution to reduce the impact of such disruptions is to introduce some redundancy in the network design such that a reasonable part of the risks could be mitigated. One way to achieve this is by making sure that the feeder round-trips do not get monopolized by only one hub port and provide the possibility of cross-rotation *escape* operation through some nominated ports. If the single hub fails or a spoke edge (due to strikes, threats, etc.), the operations are re-directed through a *runaway* node with a minimal overhead operational cost.

Humanitarian aids distribution. In a study we recently carried out related to distribution of humanitarian aids among the UNHCR-recognized displaced Syrians who fled the war and found refuge on Lebanon as the second hosting country according to the UNHCR[1]. A set of hub locations identified among the demand nodes are supplied by a central depot in Beirut suburbs. Cycles are formed starting from every hub locations, visiting a set of spoke demand points (the villages wherein refugees are residing) and returning to the depot on the same day (see Fig. 2). For some domestic reasons, it is very likely that a hub node becomes unavailable at anytime without any prior notice. This can even happen after it receives the supply for spoke nodes allocated to it in which case new supply must be sent from the main depot but *via* a different *runaway node*.

1.1. Literature review

The work in this paper is inspired from the uncapacitated single allocation hub location problem (USAHLP). In the following, we present the closely related studies. The works are ordered by publication date from the newest to the oldest.

In their work, Monemi *et al.* [28] presented a case study on the distribution of humanitarian aid. They proposed a formulation for the Multiperiod Hub Location Problem with Serial Demand (MPHLPSD) using Mixed

[1] https://data.unhcr.org/en/situations/syria.

Integer Programming (MIP). The researchers identified several valid inequalities and employed the benders decomposition approach as a solving method for handling large instances.

In their paper, Kemmar *et al.* [23] introduced a novel hub location problem structure, utilizing the term "runaway node" for the first time to the best of our knowledge. The researchers proposed a MIP formulation and also explored hyper-heuristic and variable neighborhood search approaches as alternative solving methods.

Danach *et al.* [14] addressed the single allocation hub location and routing problem, considering the constraint that the flow volume passing through an edge at the spoke level must not exceed the capacity of available transporters. The researchers proposed a MIP formulation, as well as utilizing Lagrangian relaxation and a hyper-heuristic as alternative solution methods.

The uncapacitated Single Allocation p-Hub Location Problem under the risk of hub disruptions was examined in [2]. The author proposed a MIP formulation and utilized the Particle Swarm Optimization (PSO) algorithm. The constructed networks in this research included a backup hub assigned to each individual demand, serving as a contingency plan in the event of disruptions.

Rostami *et al.* [32] introduced the reliable single allocation Hub Location Problem, which employs a two-stage formulation and utilizes the benders decomposition approach for solving large-scale instances. In this work, when a hub experiences a breakdown, the corresponding flow is rerouted through a designated single backup hub.

Zhong *et al.* [39] investigated the hierarchical hub location model and proposed a MIP formulation that incorporates hub capacity constraints. Additionally, a hybrid meta-heuristic approach combining tabu search and genetic algorithm was introduced.

The Ring Spur Assignment Problem was initially introduced by Carroll *et al.* [9] in the context of next-generation telecommunications networks. Monemi and Gelareh [27] proposed a 2-index integer programming (IP) formulation for this problem and developed a branch-and-cut algorithm that incorporates various classes of valid inequalities. Although there are certain differences, this problem shares some common features with the current research.

For the Bounded Cardinality Capacitated Hub Routing Problem (BCCHRP) with route capacity constraints, Gelareh *et al.* [22] proposed a 2-index MIP formulation. Additionally, they developed a branch-and-cut approach based on Benders decomposition.

Contreras *et al.* [12] presented a MIP formulation and a branch-and-cut algorithm for the Cycle Hub Location Problem (CHLP). To tackle large-scale instances of the CHLP, a greedy randomized adaptive search procedure (GRASP) was developed.

In [31], a two-level network was proposed, consisting of a backbone ring network at the upper level connecting the hub nodes, and an access ring at the lower level connecting the spoke nodes to the hub nodes. This pure location problem does not incorporate any flow, and a branch-and-cut algorithm was provided as a solution method.

Chaharsooghi *et al.* [11] addressed the reliable uncapacitated multiple allocation hub location problem under hub disruptions. They introduced a two-stage stochastic model and employed an adaptive large neighborhood search meta-heuristic approach. In this context, when a hub fails, the spokes allocated to that hub are either reallocated to other functioning hubs or a penalty is incurred if they do not receive any service due to high reallocation costs.

Mohammadi *et al.* [26] addressed the single allocation p-hub center-median problem under data uncertainty. The study proposed a bi-objective mixed-integer non-linear programming model and introduced an evolutionary algorithm as a solution approach. For readers interested in reliability network problems, the following studies are recommended: [7, 35, 38].

In the work by de Sá *et al.* [15], the Hub Line Location Problem (HLLP) for public transportation systems was studied. de Sá *et al.* [16] presented a benders decomposition algorithm and several metaheuristics as solution methods for this problem.

In [30], the hub location and routing problem was investigated, where the number of hubs is predetermined, and the capacity constraint is defined in terms of the number of spokes per cycle. The research proposed a MIP formulation and a branch-and-cut algorithm to address this problem.

Gelareh *et al.* [21] introduced a hub-and-spoke structure comprising a central hub cycle and spoke-level (feeder) cycles connected to each hub node. To solve the problem, they employed a Lagrangian decomposition approach combined with a Lagrangian heuristic.

Yang *et al.* [37] tackled the p-Hub Center Problem in fuzzy environments. They proposed a hybrid PSO algorithm that integrates PSO, genetic operators, and local search (LS) techniques.

Alumur *et al.* [1] investigated the hierarchical multi-modal hub location problem. They developed a MIP formulation that considers two types of hub nodes and hub edges for ground and air transportation, specifically for time definite delivery services.

Gelareh and Nickel [20] addressed the uncapacitated multiple allocation Hub Location Problem, focusing on urban transport and liner shipping network structures. The authors proposed a MIP formulation, utilizing a benders decomposition method and a greedy heuristic as solution approaches.

Çetiner *et al.* [10] proposed an iterative two-stage heuristic for hub location problems. Firstly, they locate the hub nodes, and then they design routes using heuristics based on the traveling salesman problem. They employed data from the Turkish postal delivery system as a case study.

Kim and O'Kelly [24] introduced a reliable p-Hub Location Problem for telecommunication networks, considering both single and multiple assignment scheme mathematical models.

In [3], a mathematical model was proposed to locate p helicopter pads and one facility (hospital) among n nodes, aiming to minimize either the total time (minisum) or the maximum time (minimax).

Yaman *et al.* [36] presented a minimax mathematical model for the latest arrival hub location problem in ground-based cargo delivery systems with stopovers.

For the capacitated single allocation hub location problem (CSAHLP), Carello [8] proposed a local search approach and various meta-heuristic algorithms. In this problem, hubs act as transit nodes and spokes as access nodes.

Ebery *et al.* [18] developed a branch-and-bound procedure with an efficient heuristic algorithm using shortest paths to obtain the upper bound for the capacitated multiple allocation hub location problem, specifically applied to postal networks.

Campbell [6] presented the first linear programming formulations for single/multiple allocation capacitated/uncapacitated hub location problems. Their work laid the foundation for subsequent research in this area. Skorin-Kapov *et al.* [33] improved the existing MIP formulation for the single allocation p-hub median problem (SApHMP).

Kuby and Gray [25] designed the least-cost single-hub air network with a predetermined hub location, which was not considered as part of the problem. They formulated it as a MIP.

O'Kelly [29] introduced the single allocation hub location problem (SAHLP). The author proposed a quadratic integer formulation where the number of hubs was an endogenous part of the problem, incorporating fixed costs. Campbell [5] introduced the first model for the multiple allocation problem.

Generally speaking, the major works in the literature on the Hub Location Problem primarily focus on standard network structures and rarely propose new ones. Additionally, the commonly employed non-exact solution procedures are predominantly heuristics or meta-heuristics, with the potential of hyper-heuristics remaining largely unexplored.

This paper presents a variation of our previous work in [23], sharing some similarities. However, there are distinct differences between this work and [23] as follows: in [23], the hub-level network is not necessarily a complete sub-graph and is an endogenous part of the problem, whereas in this study, it is a complete subgraph. Moreover, the aforementioned work incorporates a discount factor λ that represents economies of scale at the runaway connections level, whereas our study does not consider such a factor. Finally, the network structure proposed in [23] differs from the network structure proposed in this work.

The main features of the most related contributions in the literature are summarized in Table 1.

TABLE 1. A summary of the relevant contributions in the literature.

Work	Alloc.	Num. hubs	Objective	Capacity	Solution method
Kemmar *et al.* [23]	SA	Exogenous	Cost	No	MIP + VNS + Hyper-heuristic
Danach *et al.* [14]	SA	Exogenous	Time (transit + transshipment)	Yes	MIP + Lagrangian relaxation + Hyper-heuristic
Zhong *et al.* [39]	SA	Endogenous	Cost	Yes	MIP + Meta-heuristic
Contreras *et al.* [12]	SA	Exogenous	Cost	No	MIP + Branch-and-cut + Meta-heuristic
Rodriguez-Martín *et al.* [31]	SA	Endogenous	Cost	$\leq q$ spokes per access ring + $1 \leq$ access rings $\leq k$ per hub	MIP + Branch-and-cut
Gelareh *et al.* [22]	SA	$q = 3 \leq \ldots \leq p$	Time (transit + transshipment)	Yes	MIP + Branch-and-cut + Benders decomposition
Rodriguez-Martín *et al.* [30]	SA	Exogenous	Cost	$\leq q$ spokes per cycle	MIP + Branch-and-cut
Gelareh *et al.* [21]	SA	Exogenous	Cost	$\geq q$ spokes per cycle	MIP + Lagrangian decomposition based heuristic
Çetiner *et al.* [10]	MA	Endogenous	Cost + Num. of vehicles	No	Heuristic
Current work	SA	Exogenous	Cost	No	MIP + Two Hyper-heuristics

1.2. Contribution and scope

The problem addressed in this paper is derived from real-life experience, where the spoke-level network structure shares similarities with the one depicted in Figures 2 and 3. Our objective is to propose a novel hub and spoke network design that offers different perspectives from the one presented in Figure 2. The proposed network design, illustrated in Figure 3, introduces a third type of node called runaway nodes. These runaway nodes hold special attributes, bridging the characteristics of sophisticated hub nodes and simple spoke nodes. They can be allocated to two hubs (two cycles). By incorporating runaway nodes, redundancy is added to the network structure, allowing it to handle potential failures such as strikes or unforeseen threats at the hub nodes or the connections to/from the hub nodes. This redundancy provides alternative paths to fulfill the demands. This work builds upon our previous works, including [21,23], and is motivated by real-life cases.

We present the first exponential MIP formulation for this problem, along with two reinforcement learning-based hyper-heuristics that provide high-quality solutions for instances of varying sizes, including small, medium, and large. Through extensive computational experiments conducted on randomly generated instances, we demonstrate the computational efficiency of our proposed solution framework and the effectiveness of these approaches.

The rest of the paper is organized as follows. The problem description is given in Section 2. Section 3 presents a MIP formulation for the problem. Section 4 describes all the components of the hyper-heuristic approaches (initial solution, selection method, low-level heuristics and the main procedure). In Section 5 we report and

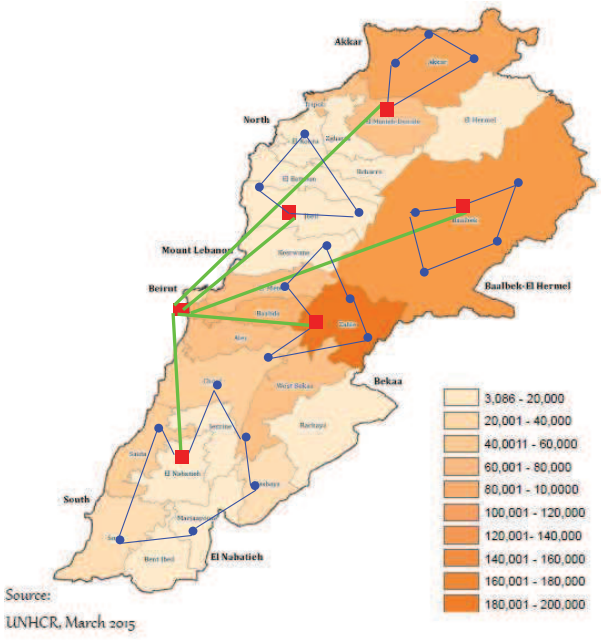


FIGURE 2. Network structure for a hub-and-spoke structure distribution network of humanitarian aids in Lebanon. Source: *Displaced Syrian distribution by CAZA, Lebanese Government*

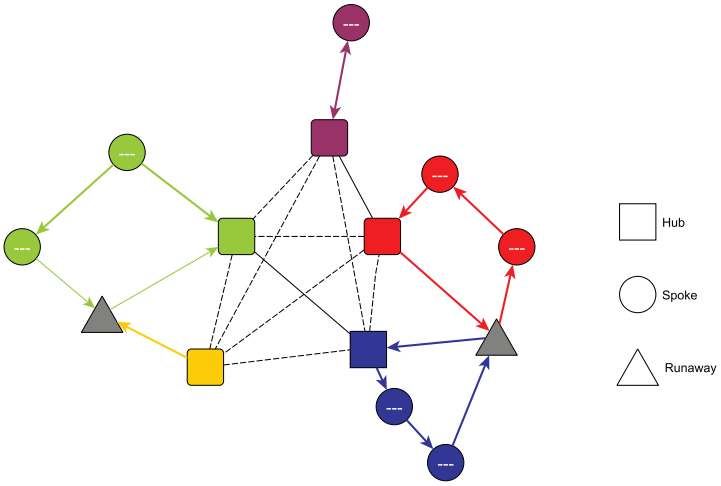


FIGURE 3. A typical network structure in a Runaway p-Hub location problem.

discuss the results of our computational experiments. The conclusion of our study and the possible future works are given in Section 6.

2. Problem statement

The Runaway p-Hub Location Problem (RpHLP), as depicted in Figure 3, can be formally described as follows:

Given a set of nodes V with $|V| = n$, a cost matrix C where c_{ij} represents the cost per unit of flow on the edge from node i to node j, and a flow/demand matrix W where w_{ij} denotes the flow from node i to node j. The fixed costs of establishing hub and runaway nodes are denoted by F_k and G_k, respectively. Additionally, I_{kl} represents the allocation cost of spoke node k to hub node l. The objective is to identify p hub nodes and construct the hub-level network using undirected hub edges to form a complete sub-graph. The remaining $n-p$ spoke nodes are allocated to different hub nodes among the p hubs. At the spoke level, nodes form directed cycles starting from the hub node, visiting all the allocated spoke nodes, and returning to the hub node. Each hub node must have at least one spoke node allocated to it. A runaway node is a special node selected from the spoke nodes, capable of being allocated to two different hub nodes and consequently two different cycles. This provides an additional access point outside the rotation, in addition to the one provided by the hub node. To capture the economies of scale at the hub-level network, a discount factor α is introduced. The objective is to determine the optimal structure that minimizes the total costs associated with establishing facilities and transportation on this structure.

3. Mixed-integer linear programming formulations

The variables of model are: $x_{ij} = 1$ $(\forall i \neq j)$, if node i is allocated to hub j, 0, otherwise; $x_{ii} = 1$ if i is a hub, 0, otherwise; $y_{ij} = 1$ $(\forall i \neq j)$, if there exists a spoke arc (i, j), 0 otherwise; $t_i = 1$, if i is runaway node, 0 otherwise; $w_{ijkl} = 1$, if the flow from i to j is routed *via* hub edge (k, l), 0 otherwise; $s_{ijkl} = 1$, if the flow from i to j is routed *via* spoke arc (k, l), 0 otherwise.

The Runaway p-Hub Location Problem (RpHLP) model can be stated as follows:

$$\min \sum_{i=1}^{n} \sum_{j=1,j\neq i}^{n} \sum_{k=1,k\neq j}^{n} \sum_{l,l\neq k,l\neq i}^{n} W_{ij}C_{kl}s_{ijkl} + \sum_{i=1}^{n} \sum_{j=1,j\neq i}^{n} \sum_{k=1,k\neq j}^{n} \sum_{l,l\neq k,l\neq i}^{n} f_{ij}\alpha C_{kl}w_{ijkl} + \sum_{k}^{n} \sum_{l,l\neq k}^{n} I_{lk}x_{lk} + \sum_{k=1}^{n} F_k x_{kk} + \sum_{k=1}^{n} G_k t_k \tag{1}$$

$$\text{s.t.} \sum_{j=1}^{n} x_{jj} = p \tag{2}$$

$$x_{ij} \leq x_{jj}, \qquad \forall i, j \in V \tag{3}$$

$$\sum_{j=1}^{n} x_{ij} = 1 + t_i, \qquad \forall i \in V \tag{4}$$

$$t_i + x_{ii} \leq 1, \qquad \forall i \in V \tag{5}$$

$$t_i + x_{ik} + x_{il} + t_j + x_{jk} + x_{jl} \leq 4, \qquad \forall i, j, k, l \in V, l \neq k, l \neq j, l \neq i, k \neq j, k \neq i, j > i \tag{6}$$

$$x_{ij} + t_i \leq 1 + y_{ij} + y_{ji}, \qquad \forall i, j, k, l \in V, l \neq k, j \neq i \tag{7}$$

$$y_{ij} + x_{ik} + x_{jl} \leq 2 + x_{jk} + x_{il}, \qquad \forall i, j, k, l \in V, l \neq k, j \neq i \tag{8}$$

$$y_{ij} + y_{ji} + x_{ki} + x_{kj} \leq 1 + x_{ii} + x_{jj}, \qquad \forall i, j, k \in V, k \neq j, k \neq i, j > i \tag{9}$$

$$\sum_{j=1, j\neq i}^{n} y_{ij} = 1 + t_i, \qquad \forall i \in V \tag{10}$$

$$\sum_{j=1, j\neq i}^{n} y_{ji} = 1 + t_i, \qquad \forall i \in V \tag{11}$$

$$\sum_{l=1, l\neq i}^{n} w_{ijil} + \sum_{l=1, l\neq i}^{n} s_{ijil} = 1, \qquad \forall i, j \in V, j \neq i \tag{12}$$

$$\sum_{l=1, l\neq j}^{n} w_{ijlj} + \sum_{l=1, l\neq j}^{n} s_{ijlj} = 1, \qquad \forall i, j \in V, j \neq i \tag{13}$$

$$\sum_{l=1, l\neq i, l\neq k}^{n} w_{ijkl} + \sum_{l=1, l\neq i, l\neq k}^{n} s_{ijkl} = \sum_{l=1, l\neq j, l\neq k}^{n} w_{ijlk} + \sum_{l=1, l\neq j, l\neq k}^{n} s_{ijlk}, \qquad \forall i, j, k \in V, k \neq i, k \neq j, j \neq i \tag{14}$$

$$s_{ijkl} \leq y_{kl}, \qquad \forall i, j, k, l \in V, l \neq k, l \neq i, k \neq j, j \neq i \tag{15}$$

$$w_{ijkl} \leq x_{kk}, \qquad \forall i, j, k, l \in V, l \neq k, l \neq i, k \neq j, j \neq i \tag{16}$$

$$w_{ijlk} \leq x_{kk}, \qquad \forall i, j, k, l \in V, l \neq k, l \neq j, k \neq i, j \neq i \tag{17}$$

$$x_{ij}, t_i, y_{kl}, \in \{0, 1\} \qquad \forall i, j, k, l \in V, l \neq k \tag{18}$$

$$w_{ijkl}, s_{ijkl} \in (0, 1), \qquad \forall i, j, k, l \in V, l \neq k, l \neq i, k \neq j, j \neq i. \tag{19}$$

The objective function minimizes the transportation costs, the hub and runaway setup costs and spoke allocation costs. The constraints (2) ensure that the number of hubs is equal to p. The constraints (3) assure that a node i will be allocated to node j only if the node j is a hub. The constraints (4) assure that a vertex i will be allocated to two hubs if i is a runaway, or to only one hub if i is a spoke. The constraints (5) assure that a node i can not be a hub and a runaway in the same time. The constraints (6) ensure that two runaways are not allocated to the same hub. The constraints (7) assure that if a runaway i is allocated to hub j then a direct link (incoming or outgoing) exists between i and j. The constraints (8) assure that a spoke arc links two spokes of the same hub (cycle). The constraints (9) assure that when the number of spokes allocated to a given hub are more than one, if the spoke arc y_{ij} exists then the spoke arc y_{ji} does not exist. The constraints (10) ensure that a spoke i can have at most one outgoing spoke arc if i is a spoke node and two spoke arcs if i is a runaway node. The constraints (11) ensure that a spoke i can have at most one incoming spoke arc if i is a spoke node and two spoke arcs if i is a runaway node. The constraints (12) assure that all the flow from node i to node j will leave the node i using the same edge il (a hub edge or a spoke arc). The constraints (13) assure that all the flow from node i to node j will reach the node j using the same edge lj (a hub edge or a spoke arc). The constraints (14) assure that every flow between two nodes i and j passing by an intermediate node l will leave this node. The constraints (15)–(17) assure that a flow between two nodes i and j will pass by the link kl (lk) (spoke arc or hub edge) if it exists. Constraints (18) are integrality constraints.

4. Hyper-heuristic for RpHLP

Due to the complexity of the problem, solving large-sized instances becomes impossible. Therefore, we propose two hyper-heuristic approaches to solve large-sized instances within a reasonable amount of time.

The term "hyper-heuristic" was first introduced by Denzinger *et al.* [17] to describe a protocol that combines multiple artificial intelligence methods. In the context of combinatorial optimization, the term was initially used

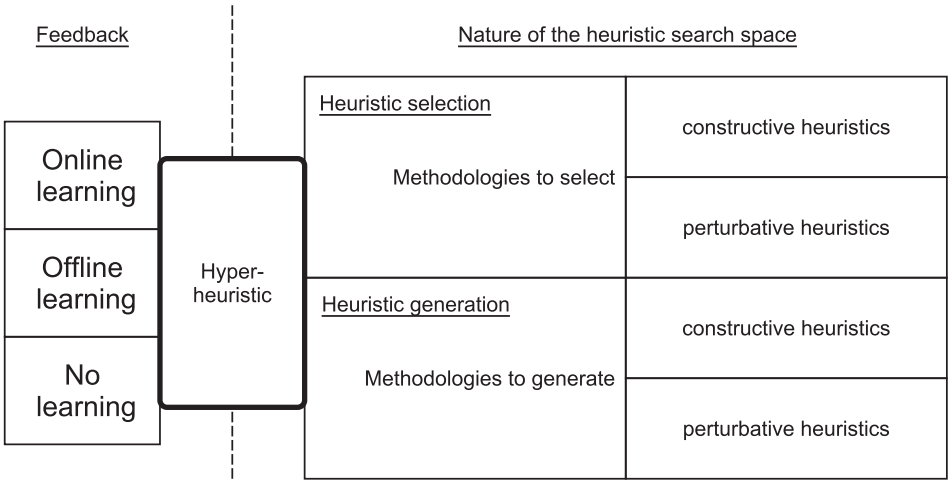


FIGURE 4. Hyper-heuristics classification.

in [13] as *heuristics to choose heuristics*. According to this definition, a hyper-heuristic is a high-level approach that, given a set of low-level heuristics and a specific problem instance, selects and applies the most suitable low-level heuristic at each milestone.

In conventional approaches, the search space generally refers to the space of solutions, aiming to identify the optimal solution. However, in hyper-heuristics, the search space encompasses the space of low-level heuristics, with the main objective being the identification, at each step, of the most effective low-level heuristic to apply for solution improvement. Figure 4 [4] offers a comprehensive overview and classification of existing hyper-heuristics.

In this study, we introduce two heuristic selection hyper-heuristics, based on an online learning scheme. The first hyper-heuristic utilizes a stochastic approach based on Q-learning, while the second hyper-heuristic employs a deterministic approach using a novel selection method called A-learning. In the subsequent discussion, we will explore various aspects that are common to both hyper-heuristics.

4.1. Initial solution

The initial solution plays a crucial role in most algorithms as it greatly influences the quality of the final solution obtained. A well-constructed initial solution increases the likelihood of finding a good solution, and in some cases, even the optimal solution. In the following, we outline the key steps involved in constructing the initial solution for our problem, as depicted in Figure 5.

(1) **Hub node location:** to determine the hub nodes, we aim to identify the p nodes that are closest to the center. This is achieved by calculating the average distances between each node and the remaining nodes, as shown in equation (20), where d_{ij} represents the distance between node i and node j. Subsequently, the p nodes with the lowest value of c are selected as our hub nodes.

$$c_i = \sum(d_{ij})/n - 1 \tag{20}$$

(2) **Runaway node location and allocation:** initially, we determine the number of runaways to be located, denoted as s, which is equal to $\lfloor p/2 \rfloor$. Next, we apply equation (20) to the remaining $n - p$ nodes. We then select s nodes with the lowest values and allocate each selected node to the two nearest hubs with no runaway allocated to them.

(3) **Spoke allocation:** each spoke node is assigned to the nearest hub node.

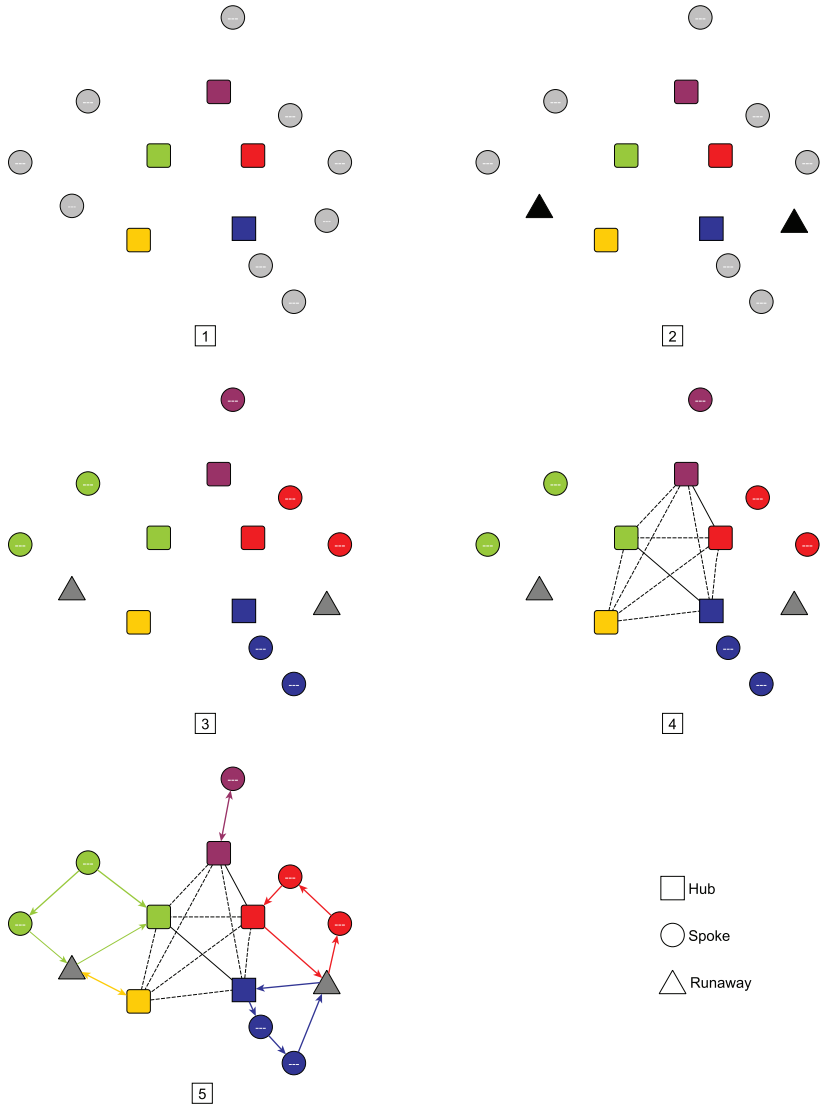


FIGURE 5. Initial solution construction.

(4) **Hub-level network:** the hub-level network is constructed as a complete graph, meaning that all the hub nodes are interconnected.
(5) **Spoke-level network:** the spoke-level network is formed by creating cycles using a greedy constructive heuristic. At each step, the nearest node is selected and connected to the existing structure.

4.2. Low-level heuristics

In the following section, we will define the neighborhood structures and the set of low-level heuristics utilized in our hyper-heuristics. Additionally, we will provide a brief explanation of how each of them operates and performs.

Network low-level heuristics. These heuristics aim to enhance the solution network connections (edges and arcs).

LINK-HUB: this low-level heuristic involves adding or removing a hub edge between each pair of hub nodes in the hub-level network. Adding a hub edge between two hub nodes, if it doesn't already exist, does not affect feasibility. However, deleting a hub edge can lead to infeasibility and disconnectedness in the hub-level network, in which case the solution is rejected.

NEW-CYCLE-LINKS: for the cycle attached to the hub node k:

- Destroy the spoke-level network by removing all the spoke arcs of the cycle k.
- Choose a node i as a start and use the nearest neighbor heuristic to create a solution for the traveling salesman problem (TSP).

Distribution low-level heuristics. These local search heuristics aim to improve each cycle by swapping nodes within the cycle without altering the number of nodes allocated to it.

SWAP-CYCLE-NODES: for each spoke node i allocated to cycle k:

- Find the nearest node j to the node i (j allocated to cycle k).
- Swap the positions of the spokes i and j on the spoke-level network.

SWAP-SPOKE-RUNAWAY: for each spoke node, we perform a swap operation with its corresponding runaway node, effectively transforming the spoke node into a runaway node and *vice versa*.

SWAP-SPOKE-HUB: we swap every spoke with its corresponding hub such that the spoke node becomes the hub and the hub node becomes a spoke node.

SWAP-DIFFERENT-CYCLE: for each spoke node i allocated to hub k:

- Find the nearest hub l to the hub k.
- Find a node among the spokes allocated to hub l, say spoke j, which is the nearest node to the spoke i.
- Swap the spoke i with the spoke j so that the spoke i takes the position of spoke j in the cycle attached to the hub node l and the other way around.

Structure low-level heuristics. The objective is to determine the optimal composition of clusters by reassigning nodes among them.

MOVE-SPOKE-NEAREST-CYCLE: for each spoke node i allocated to hub node k:

- Find the nearest hub l to the hub k.
- Find a node among spokes allocated to hub l, say spoke j, which is the nearest node to the spoke i.
- Remove spoke i from cycle attached to the hub node k.
- Add the spoke node i in cycle attached to the hub node l before/after the spoke j in the spoke-level network.

MOVE-SPOKE-NEAREST-NODE: this low-level heuristic identifies the nearest spoke node j in the entire network (not restricted to the nearest cycle to hub node k as in the previous low-level heuristic) for a given spoke node i allocated to hub node k. Once the spoke node j and its corresponding hub node l are determined:

- Removes the spoke node i from cycle attached to the hub node k.
- Add the spoke node i to cycle attached to the hub node l before/after the spoke node j in the spoke-level network.

ADD-RUNAWAY-NODE: for each two cycles with there respective hubs k and l without a runaway node:

- Find the nearest node i to the two hubs k and l.
- Add the spoke node i as a runaway node to cycles attached to the hub nodes k and l.

REMOVE-RUNAWAY-NODE: for each two cycles with a runaway node i.

- Remove runaway node i from the two cycles.
- Allocate runaway node i to the cycle with the nearest hub node as a spoke node.

4.3. Stochastic hyper-heuristic

In the following we give the main components of the stochastic approach.

4.3.1. Selection method

The selection method employed in this approach is known as Q-learning. Q-learning operates with an agent that repeatedly selects actions based on the current state and receives feedback according to the utility of the subsequent state. The utility value of the current state is updated using the following formula:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\Big(R + \gamma \max_a Q_t(s',a) - Q_t(s,a)\Big). \tag{21}$$

In this formula, the reward R is obtained after taking the action a in the state s. The learning rate α is a parameter between 0 and 1, where $\alpha = 0$ means the agent does not learn from the current experience, and $\alpha = 1$ means the agent only considers the most recent information. The discount factor γ is also between 0 and 1 and determines the importance of future rewards in the decision-making process. The term $\max Q_t(s',a)$ represents the maximum expected utility from the future state s'.

The Algorithm 1 describes the main steps of our Q-learning procedure.

Algorithm 1: Q-LEARNING.

Input: array $Q[S,A]$, precedent state s, precedent action a, γ, α
if $random(0,1) < \epsilon$ **then**
  $a \leftarrow$ random action
else
  $a \leftarrow \text{argmax}_z\, Q(s,z)$
Calculate reward R
get the new state s'
$Q[s,a] \leftarrow Q[s,a] + \alpha(R + \gamma \max_a Q[s',a] - Q[s,a])$
$s \leftarrow s'$

4.3.2. Move acceptance strategy

Depending on the move acceptance strategy employed, the solution obtained after applying a low-level heuristic may be either accepted or rejected. In our hyper-heuristic algorithm, we use an adaptive acceptance strategy. In this approach, the acceptance of a deteriorating move is contingent upon a probability of $1 - 1/C$, where C is incremented after each worsening move and is reset to 1 whenever a solution improvement occurs.

4.3.3. Hyper-heuristic procedure

Our hyper-heuristic algorithm begins by generating the initial solution based on the approach outlined in Section 4.1 and initializing the Q array. The selection of the low-level heuristic to be applied is determined using Q-learning. The best solution found, along with its objective value, is then stored in variables x' and min, respectively. If x' represents the best solution encountered thus far, it is stored in x^*, and its objective function value is stored in top, designating it as the incumbent solution x_i. Alternatively, x' is stored as the incumbent solution x_i with a probability of $1 - 1/C$, as shown in Section 4.3.2.

Our termination criteria are defined by either reaching the maximum number of consecutive non-improving moves, N_{limit}, or hitting the time limit, T_{limit}.

One can summarize the algorithm process in 4 main steps (see Algorithm 2). (1) The Q array is initialized to 0 at line 2. (2) The low-level heuristic with the best Q value is applied to the incumbent solution x_i at line 7. (3) The neighborhood solutions obtained are explored within the range of lines 11–16, and the best solution found is stored as the current solution x' at line 15. (4) If the current solution x' demonstrates an improvement over

TABLE 2. The parameters used in Algorithm 2.

Parameter	Definition
nbr_iter	The number of successive worsening moves
T_{limit}	The time limit
N_{limit}	The number of worsening moves tolerated
$sols$	An array of solutions
x'	The current solution
x_i	The incumbent solution
x^*	The best solution found

the best-found solution so far at line 17, we update the best-found solution, x^*, at line 18, and the incumbent solution x_i at line 19. Otherwise, if the current solution is not superior, it is accepted as the incumbent solution with a probability of $1 - 1/C$ at lines 25 and 26. Steps 2–4 are iterated until the termination criteria are met at line 6.

The parameters of our algorithm are presented in Table 2.

Algorithm 2: STOCHASTIC HYPER-HEURISTIC.

Input: Data
Output: The best solution found

```
1  x_i ← Initial_solution();
2  Q_initialization(0);
3  x* ← x_i
4  top ←← Eval(x*)
5  nbr_iter ← 0
6  while nbr_iter ≤ N_limit and time < T_limit do
7      sols ← select_heuristic(x_i) // based on Q-learning algorithm
8      x' ← sols(0)
9      min ← Eval(x')
10     i ← 1
11     while i ≤ nbr_sols do
12         cost ← Eval(sols(i))
13         if min > cost then
14             min ← cost
15             x' ← sols(i)
16         i ← i + 1
17     if min < top then
18         x* ← x'
19         x_i ← x'
20         top ← min
21         nbr_iter ← 0
22         C ← 1
23     else
24         C ← C + 1
25         if random(0, 1) < (1 − 1/C) then
26             x_i ← x'
27     nbr_iter ← nbr_iter + 1
28 return x*
```

4.4. Deterministic hyper-heuristic

In the following we give the main components of the proposed deterministic approach.

4.4.1. Selection method

The selection method employed in this approach is known as *A*-learning. *A*-learning utilizes an array of weights, denoted as W, where w_{ij} represents the utility value of applying the low-level heuristic j after the low-level heuristic i. The utility value is updated using the following formula:

$$w_{ij} = \left(\sum_{1}^{N_{ij}} \left(f^{\text{in}} - f^{\text{out}} \right) \right) / N_{ij}. \tag{22}$$

The weight (w_{ij}) is calculated as the negative sum (in the case of minimization) of the difference in objective function values after applying the low-level heuristic j, divided by the total number of times it has been called after the low-level heuristic i during the search. Here, f^{in} represents the current objective value, f^{out} is the objective value after applying the low-level heuristic j, and N_{ij} denotes the number of times the low-level heuristic j has been used after i. This weight provides an estimation of the average effectiveness of applying the low-level heuristic j following the low-level heuristic i. In other words, the selection of the low-level heuristic j is based on the prior application of the low-level heuristic i.

The selection procedure Algorithm 3 chooses the low-level heuristic j with the highest weight and nc_j less or equal to the parameter Rl (Rl is the maximum authorized consecutive calls) and increments the number of consecutive calls of the low-level heuristic, j, *i.e.* nc_j (line 11).

Algorithm 3: *A*-LEARNING.

Input: Array $W[L, L]$ of weights, i the last low-level heuristic used, a set $Nc = \{nc_1, nc_2, \ldots, nc_n\}$ the number of consecutive calls of each low-level heuristic and R_l the restriction on the number of consecutive calls.
Output: $z = \text{argmax}_j W(i, j)$

```
1  max ← −∞
2  z ← −1
3  for j ← 1 to n do
4      if w_ij > max and nc_j ≤ R_l then
5          if z > −1 then
6              nc_z ← 0
7          max ← w_ij
8          z ← j
9      else
10         nc_j ← 0
11 nc_z ← nc_z + 1
12 return j
```

4.4.2. Move acceptance strategy

The move acceptance strategy governs the algorithm's behavior in relation to each solution encountered. It determines whether a solution should be accepted or rejected based on the chosen criterion, which in turn affects the search trajectory and the ultimate outcome. In this approach, the random walk acceptance criteria is employed. This criteria accepts a solution regardless of its quality, introducing some randomness into the search path to prevent premature convergence.

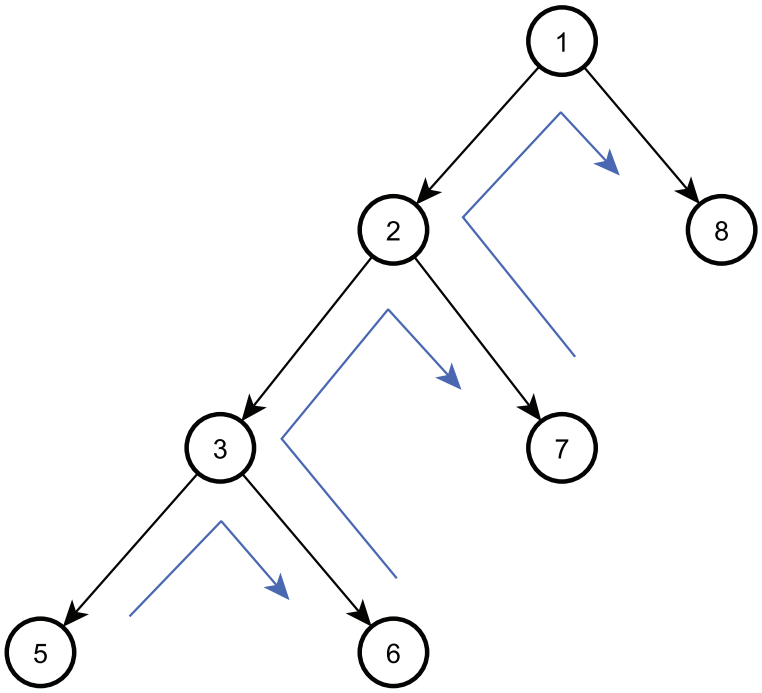


FIGURE 6. Backtracking on solution tree.

4.4.3. Backtracking mechanism

The final outcome of any approach is determined by a series of decisions made throughout the entire process. A poor decision can result in an unfavorable outcome. To mitigate this, incorporating a backtracking mechanism helps minimize the occurrence of bad decisions during the process. In our case, a decision corresponds to selecting a low-level heuristic to apply. If the algorithm becomes trapped in a local optimum, it initiates backtracking to a previously successful solution and employs a different low-level heuristic to explore an alternative path that may lead to a better outcome (refer to Fig. 6). In our proposed approach, the backtracking mechanism is triggered after a consecutive sequence of b worsening moves (indicating poor decisions) to revert to the context of the last favorable decision made. If a backtracking branch fails to yield an improved solution, it is removed using a last-in, first-out (LIFO) scheme.

4.4.4. Hyper-heuristic procedure

The proposed algorithm bears many similarities to the stochastic one. For instance, both algorithms utilize the same initial solution, stopping criteria, and terminology.

This algorithm initiates by generating the initial solution and initializing the weights array. Employing A-learning, the low-level heuristic to apply is selected. The best solution found, along with its objective value, is then recorded as x' and min, respectively. Subsequently, the weights array is updated accordingly.

If x' outperforms its parent solution in the backtracking solutions tree, it is stored as x_b and added to the tree. If x' represents the best solution discovered thus far, it is stored as x^*, and its objective function value is recorded as top.

Our termination criteria are defined by either reaching the maximum number of consecutive non-improving moves (N_{limit}) or reaching the time limit (T_{limit}).

The algorithm process can be summarized in 8 main steps (see Algorithm 4). (1) The weights array initialization (line 2) (the weights array is initialized to 0 similarly to Q-learning). (2) After each b non-improving moves

TABLE 3. The parameters used in Algorithm 4.

Parameter	Definition
b	Backtracking after b worsening moves
B_l	Backtracking array of solutions
nbr_iter	The number of successive worsening moves
T_{limit}	The time limit
N_{limit}	The number of worsening moves tolerated
$sols$	An array of solutions
x'	The current solution (incumbent)
x^*	The best solution found

(line 9) the algorithm backtracks on the solutions tree (line 10). (3) The low-level heuristic with the best weight is applied on the incumbent solution x' (line 12). (4) The neighborhood solutions obtained are explored (lines 16 to 21) and the best solution found is stored as the current incumbent solution x' (line 15). (5) The weights array is updated (line 22). (6) If the current solution x' demonstrates an improvement over its parent solution at line 23, we update the parent solution, x_b, at line 24, and add the incumbent solution to the backtracking tree at line 26. (7) If the current solution x' shows improvement over the best-found solution thus far at line 28, we update the best-found solution, x^*, at line 29. (8) Steps 2–7 are iterated until the termination criteria is met (line 8).

The parameters of our algorithm are presented in Table 3.

5. COMPUTATIONAL RESULTS

Due to the confidentiality, we are only able to report our results on the instances generated using the Australian Post (AP) dataset [19]. The fixed costs are generated based on the distance and flow to get realistic costs as in the following (see [20]):

- Hub nodes: $F_k = (\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} w_{ij}/\max_{i\neq k} d_{ik}) \times 10\text{e}^8$, where $\max_{i\neq k} d_{ik}$ is the distance between the most remote location to k. If this distance is a large number, the node k is far from the remaining nodes and therefore F_k is less expensive.
- Runaway nodes: $G_k = \frac{1}{2}(\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} w_{ij}/\max_{i\neq k} d_{ik}) \times 10\text{e}^8$. G_k is calculated in the same way as F_k but slightly cheaper than hub nodes.
- Hub edges: $I_{kl} = (\frac{w_{kl}/d_{kl}}{\max_{i,j\neq i} w_{ij}/d_{ij}}) \times 10\text{e}^7$. This function is based on the distance and flow to get the importance of each edge and its cost. A high cost I_{kl} means that the edge $k-l$ transits a high flow w_{kl} and(or) k is close to l.

The costs were generated according to the importance of each node and edge. The more a node or a edge is important, the higher the cost is.

The name of each instance is referred as $nwwpxx\_yy$ where ww is the number of nodes, xx is the number of hubs, yy is the hub-level discount factor α. Three combinations have been taken into account, namely, $(\alpha) \in \{0.6, 0.9, 1\}$ to avoid too many similar solutions and have a better approximation of real-life cases.

The algorithms were implemented in C++ and CPLEX 12.8.0 was used as MIP solver for solving our mathematical model for different instances. The experiments were performed on an Intel 2.10 GHz core i7 CPU with 8 GB RAM running on Windows 10.

In the tables below, MIP-RpHLP refers to the exact branch-and-cut method (solving with CPLEX), Q-RpHLP stands for the Q-learning based hyper-heuristic method, and A-RpHLP for the A-learning based hyper-heuristic.

Algorithm 4: DETERMINISTIC HYPER-HEURISTIC.

Input: Data instance, R_l, b, T_{limit} and N_{limit}
Output: The best solution found

$x' \leftarrow Initial\_solution()$;
$Weight\_initialization(x')$;
$x^* \leftarrow x'$
$top \leftarrow Eval(x^*)$
$B_l \leftarrow x'$
$topb \leftarrow top$
$nbr\_iter \leftarrow 1$
while $nbr\_iter \leq N_{limit}$ ***and*** $time < T_{limit}$ **do**
  if $(nbr\_iter\ mod(b) = 0)$ **then**
    $x' \leftarrow backtrack(B_l)$
    $topb \leftarrow Eval(x')$
  $sols \leftarrow select\_heuristic(x')$ // based on A-learning algorithm
  $x' \leftarrow sols(0)$ // the first solution of the array of solutions sols
  $min \leftarrow Eval(x')$
  $i \leftarrow 1$
  while $i \leq nbr\_sols$ **do**
    $cost \leftarrow Eval(sols(i))$
    if $min > cost$ **then**
      $min \leftarrow cost$
      $x' \leftarrow sols(i)$
    $i \leftarrow i + 1$
  $Update\_weights()$;
  if $min < topb$ **then**
    $x_b \leftarrow x'$
    $topb \leftarrow min$
    $B_l \leftarrow x'$
    $nbr\_iter \leftarrow 1$
    if $min < top$ **then**
      $x^* \leftarrow x'$
      $top \leftarrow min$
  $nbr\_iter \leftarrow nbr\_iter + 1$
return x^*

5.1. Branch-and-cut approach

In this section, we examine only the computational behavior of the MIP model when CPLEX is being used as a modern (equipped with callbacks) MIP solver. The time limit was set to 86 400 s (*i.e.* 1 day).

The results are reported in Table 4 where "Obj. Val." is the objective value of the best solution found. "NSols" represent the number of feasible solutions found; "E.T" stands for the execution time; the column "GAP" reports the relative GAP returned by CPLEX upon termination; "Nnodes" gives the number of nodes processed by CPLEX; "Status" reports the status of CPLEX when termination criteria is met; "best obj." reports the best lower bound found by CPLEX.

As shown in Table 4, we were able to find optimal solutions for the majority of instances with fewer than 20 nodes. However, for instances with 15–20 nodes, CPLEX requires more time to demonstrate optimality. When the number of nodes exceeds 20, CPLEX is no longer able to find a solution, and even finding a feasible solution becomes impossible.

In Figure 7a, it can be observed that there is no clear correlation between the value of α and the execution time of CPLEX. No distinct pattern indicating the impact of α on CPLEX time is evident. However, it is notable

TABLE 4. Computational results of MIP-RpHLP.

Instance	Obj. Val.	NSols.	E.T (s)	GAP (%)	Nnodes	Status	Best Obj.
n10p3_0.6	94 370 539.823	12	6524.164	0.000	10 268	Optimal	94 370 539.823
n10p3_0.9	100 232 452.414	12	7015.021	0.000	11 832	Optimal	100 232 452.414
n10p3_1	102 052 795.999	11	4637.816	0.000	11 609	Optimal	102 052 795.999
n10p4_0.6	75 603 981.388	9	556.627	0.000	1550	Optimal	75 603 981.388
n10p4_0.9	83 890 888.293	6	564.317	0.000	1729	Optimal	83 890 888.293
n10p4_1	86 427 920.364	5	562.695	0.000	2069	Optimal	86 427 920.364
n11p3_0.6	22 246 312.176	4	14 285.448	0.000	7041	Optimal	22 246 312.176
n11p3_0.9	23 790 349.013	8	13 122.398	0.000	7085	Optimal	23 790 349.013
n11p3_1	24 305 028.163	9	9967.636	0.000	8515	Optimal	24 305 028.163
n11p4_0.6	19 052 910.179	7	1400.671	0.000	1056	Optimal	19 052 910.179
n11p4_0.9	20 969 461.076	6	1890.965	0.000	1731	Optimal	20 969 461.076
n11p4_1	21 608 311.630	7	1508.857	0.000	1717	Optimal	21 608 311.630
n12p4_0.6	21 780 541.665	9	7173.923	0.000	2488	Optimal	21 780 541.665
n12p4_0.9	23 848 677.309	6	9134.528	0.000	3391	Optimal	23 848 677.309
n12p4_1	24 538 056.131	8	12 589.717	0.000	5455	Optimal	24 538 056.131
n13p5_0.6	24 092 707.617	7	12 354.186	0.000	3659	Optimal	24 092 707.617
n13p5_0.9	27 130 967.589	12	46 173.722	0.000	11 085	Optimal	27 130 967.589
n13p5_1	28 133 751.974	19	78 622.149	0.000	25 677	Optimal	28 133 751.974
n14p5_0.6	44 768 423.665	10	86 401.918	13.877	5699	Feasible	38 555 601.234
n14p5_0.9	49 494 648.277	4	86 400.810	10.442	6182	Feasible	44 325 978.222
n14p5_1	51 093 496.271	6	86 401.216	10.061	9132	Feasible	45 952 888.713
n14p6_0.6	40 683 530.506	6	83 989.004	0.000	10 991	Optimal	40 683 530.506
n14p6_0.9	45 440 775.224	9	75 306.612	0.000	10 423	Optimal	45 440 775.224
n14p6_1	46 570 856.057	6	34 812.511	0.000	7119	Optimal	46 570 856.057
n15p6_0.6	48 474 186.521	6	86 400.497	13.129	4174	Feasible	42 109 549.503
n15p6_0.9	55 836 218.389	10	86 401.277	10.442	4487	Feasible	50 005 559.109
n15p6_1	56 664 852.977	8	86 401.293	7.147	6762	Feasible	52 614 817.024
n15p7_0.6	44 728 640.566	11	85 727.167	0.000	15 143	Optimal	44 728 640.566
n15p7_0.9	52 430 892.433	6	86 404.491	2.931	12 137	Feasible	50 894 051.746
n15p7_1	53 989 330.833	6	64 291.489	0.000	16 610	Optimal	53 989 330.833
n20p7_0.6	99 264 715.533	3	86 400.903	44.014	129	Feasible	55 573 646.017
n20p7_0.9	94 171 712.183	2	86 400.295	24.378	180	Feasible	71 213 836.585
n20p7_1	108 681 110.389	2	86 402.338	30.054	335	Feasible	76 017 543.617
n25p7_0.6	–	–	–	–	–	Failed	–
n25p7_0.9	–	–	–	–	–	Failed	–
n25p7_1	–	–	–	–	–	Failed	–
n30p8_0.6	–	–	–	–	–	Failed	–
n30p8_0.9	–	–	–	–	–	Failed	–
n30p8_1	–	–	–	–	–	Failed	–
n35p8_0.6	–	–	–	–	–	Failed	–
n35p8_0.9	–	–	–	–	–	Failed	–
n35p8_1	–	–	–	–	–	Failed	–

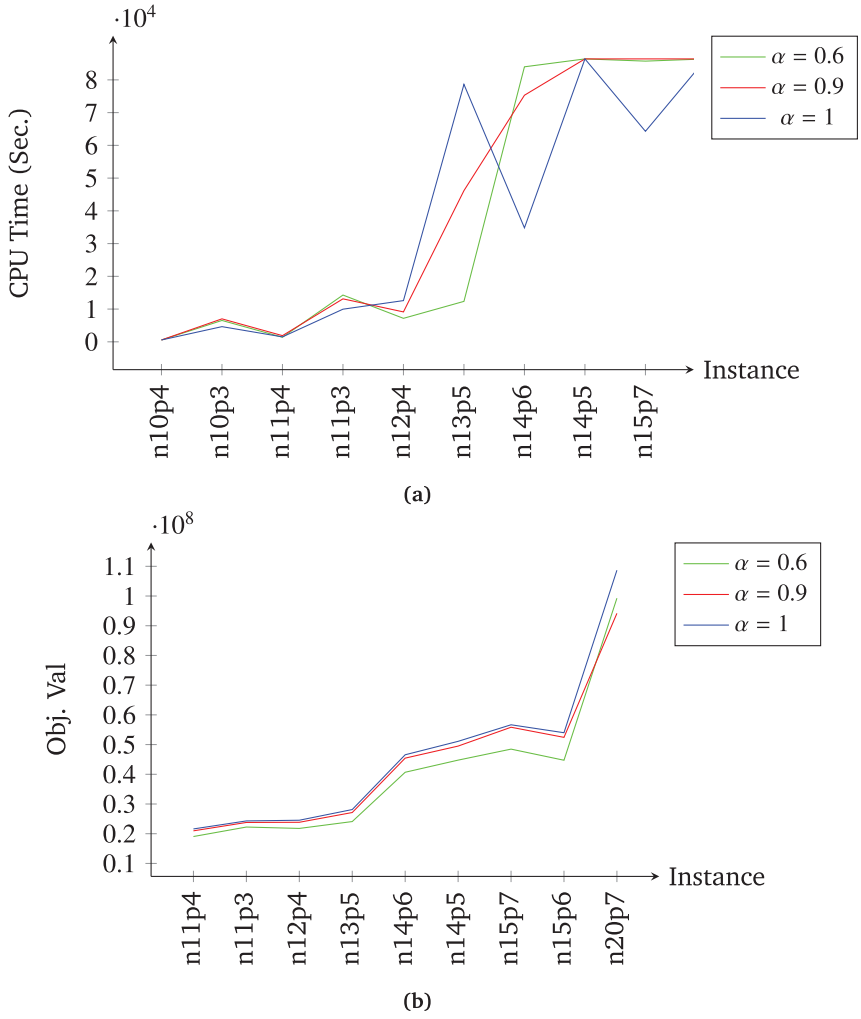


FIGURE 7. Numerical results of the MIP model. (a) MIP time. (b) Objective value comparison.

that for instances of the same size, lower values of p require longer computational times to prove optimality. Since we only obtained provably optimal solutions for instances up to size $n = 15$, Figure 7a displays data only up to this size.

In general, it can be observed that the objective value increases as the size of the instance grows. Moreover, for instances of the same size n, smaller values of p tend to result in higher objective values (as shown in Fig. 7b).

5.2. Hyper-heuristic approaches

The results of the hyper-heuristics are presented in Table 5 for small-sized instances and Table 6 for medium and large-sized instances. The columns "Best. Obj." and "Worst. Obj." represent the best and worst objective values obtained, respectively. The "Obj. Val." column displays the average objective value achieved throughout the entire process. The relative standard deviation is shown in the "RSD" column. "NSols" indicates the number of distinct feasible solutions encountered in each execution. "E.T" denotes the execution time per run. In Table 5, the "GAP" column presents the relative gap between the best-known solution of the hyper-heuristic and the

TABLE 5. Results of the hyper-heuristic algorithms for small size instances.

Instance	*Q*-R*p*HLP							*A*-R*p*HLP			
	Best. Obj.	Obj. Val.	Worst. Obj	RSD (%)	NSols.	E.T (s)	S-HH cplex GAP (%)	Best. Obj.	NSols.	E.T (s)	D-HH cplex GAP (%)
n10p3_0.6	94 370 539.823	95 170 929.746	97 707 238.509	0.849	13 200	0.411	0.000	94 370 539.823	30 627	T_l	0.000
n10p3_0.9	101 546 417.509	101 057 334.599	103 230 701.713	1.045	11 727	0.365	1.310	100 232 452.414	29 978	T_l	0.000
n10p3_1	102 052 795.999	103 035 275.182	106 062 450.459	0.925	12 810	0.405	0.000	102 052 795.999	25 866	T_l	0.000
n10p4_0.6	75 603 981.388	78 852 655.292	80 965 817.250	2.359	5274	0.199	0.000	75 603 981.388	20 797	T_l	0.000
n10p4_0.9	83 890 888.293	84 986 235.520	88 577 759.385	1.612	9793	0.321	0.000	85 486 063.967	14 994	30.062	1.901
n10p4_1	86 427 920.364	89 562 172.630	92 459 545.631	2.737	6185	0.208	0.000	86 427 920.364	125 181	54.741	0.000
n11p3_0.6	22 246 312.176	23 434 179.431	24 798 268.634	3.240	14 041	0.531	0.000	22 246 312.176	93 900	17.817	0.000
n11p3_0.9	23 790 349.013	24 418 950.386	25 157 934.774	1.897	14 753	0.566	0.000	23 790 349.013	29 928	T_l	0.000
n11p3_1	24 305 028.163	24 590 888.920	25 445 974.190	1.555	15 939	0.603	0.000	24 330 670.964	28 734	T_l	0.105
n11p4_0.6	19 052 910.179	19 890 982.163	21 709 702.553	5.083	11 908	0.438	0.000	19 052 910.179	45 455	15.305	0.000
n11p4_0.9	20 969 461.076	21 695 237.748	23 927 064.668	4.301	10 280	0.392	0.000	21 322 513.666	18 331	T_l	1.683
n11p4_1	21 608 311.630	22 641 224.957	23 461 874.780	3.121	8925	0.334	0.000	21 608 311.630	25 029	T_l	0.000
n12p4_0.6	21 780 541.665	23 099 759.157	23 717 447.021	2.203	13 347	0.584	0.000	21 780 541.665	16 211	7.394	0.000
n12p4_0.9	23 848 677.309	25 268 720.354	26 170 568.692	2.453	15 195	0.674	0.000	23 848 677.309	202 915	T_l	0.000
n12p4_1	24 538 056.131	26 242 328.298	27 677 612.312	2.560	12 146	0.528	0.000	24 538 056.131	24 041	T_l	0.000
n13p5_0.6	24 092 707.617	24 925 052.445	26 909 741.590	2.576	16 984	0.838	0.000	24 092 707.617	207 860	T_l	0.000
n13p5_0.9	27 130 967.589	28 764 210.506	30 783 868.279	3.869	12 958	0.657	0.000	27 951 474.936	219 686	T_l	3.024
n13p5_1	28 133 751.974	30 145 777.298	31 751 676.205	2.865	13 070	0.670	0.000	28 133 751.974	194 398	T_l	0.000
n14p5_0.6	44 768 423.665	46 478 818.112	48 513 324.370	1.620	30 371	1.804	0.000	45 166 742.924	152 950	T_l	0.889
n14p5_0.9	49 078 200.450	54 627 351.443	58 861 077.826	5.988	17 001	1.016	−0.841	48 751 617.078	208 930	T_l	−1.501
n14p5_1	49 805 861.909	54 784 990.208	59 916 126.625	5.213	16 683	1.019	−2.520	50 138 302.461	226 819	T_l	−1.869
n14p6_0.6	41 180 058.785	49 371 374.841	51 783 123.735	6.212	8639	0.518	1.220	41 858 055.911	120 193	T_l	2.886
n14p6_0.9	46 126 868.422	55 486 770.056	58 683 738.193	6.372	17 473	1.036	1.509	48 632 689.869	189 454	T_l	7.024
n14p6_1	47 380 126.950	59 595 368.586	60 931 435.953	4.432	15 359	0.928	1.737	49 958 975.995	190 085	T_l	7.275
n15p6_0.6	48 432 576.558	52 117 336.637	58 246 060.048	4.878	31 571	2.173	−0.085	48 284 015.066	194 625	T_l	−0.392
n15p6_0.9	55 036 369.859	60 305 057.868	68 206 688.231	4.009	27 361	1.909	−1.432	55 151 768.984	125 893	T_l	−1.225
n15p6_1	56 262 216.132	63 009 605.778	67 278 434.854	3.643	26 748	1.869	−0.710	56 803 831.313	157 085	51.219	0.245
n15p7_0.6	44 858 799.634	49 022 941.179	50 866 623.799	2.859	23 414	1.570	0.290	45 060 632.859	204 210	T_l	0.742
n15p7_0.9	53 052 573.045	57 641 892.575	62 548 338.424	2.962	21 136	1.460	1.185	52 430 892.433	139 929	T_l	0.000
n15p7_1	54 368 445.692	60 531 474.253	61 989 430.836	2.568	20 261	1.409	0.702	55 731 760.060	96 266	T_l	3.227
n20p7_0.6	78 734 372.468	80 979 334.510	91 621 539.362	3.021	52 778	6.919	−20.682	85 993 179.074	166 590	T_l	−13.369
n20p7_0.9	88 414 940.063	93 467 421.786	102 364 733.063	3.663	43 054	5.708	−6.113	89 517 407.283	192 197	T_l	−4.942
n20p7_1	92 866 010.903	97 831 149.827	104 536 878.695	2.707	43 803	5.898	−14.551	102 538 261.035	206 184	T_l	−5.652

TABLE 6. Results of the hyper-heuristics for medium and large size instances.

Instance	Q-RpHLP						A-RpHLP			
	Best. Obj.	Obj. Val.	Worst. Obj	RSD (%)	NSols.	E.T (s)	Best. Obj.	NSols.	E.T (s)	D-HH S-HH GAP (%)
n40p5_0.6	176 697 314.301	201 066 884.069	241 865 045.166	8.661	20425	18.930	165 968 233.811	114 010	T_l	−6.072
n40p5_0.9	186 346 883.331	213 403 650.641	254 392 909.633	9.052	25755	24.180	192 471 847.812	114 843	T_l	3.286
n40p5_1	189 633 584.379	221 337 835.081	261 576 500.064	8.844	21091	20.522	157 183 811.123	115 905	T_l	−17.111
n40p10_0.6	116 278 440.996	123 878 225.692	131 905 227.103	3.23	8209	8.189	108 796 329.780	115 471	T_l	−6.434
n40p10_0.9	129 269 005.238	138 172 560.433	148 292 140.241	3.715	6383	6.444	102 692 941.354	109 652	T_l	−20.558
n40p10_1	132 477 862.671	142 018 328.176	151 594 214.635	3.627	8905	8.959	108 259 760.721	110 719	T_l	−18.280
n40p20_0.6	64 204 608.601	66 357 635.159	68 389 588.887	1.414	6178	6.083	66 638 521.807	105 561	T_l	3.790
n40p20_0.9	79 406 057.531	80 864 866.018	82 519 707.558	0.787	4910	4.861	80 497 614.836	102 664	T_l	1.374
n40p20_1	83 913 326.169	85 538 320.392	87 214 866.414	0.939	4324	4.295	84 738 577.368	101 403	T_l	0.983
n50p5_0.6	206 983 179.517	261 534 413.213	300 867 579.694	10.889	34 286	61.178	169 701 596.159	63 858	T_l	−18.011
n50p5_0.9	217 231 890.111	264 301 569.952	304 552 694.664	9.594	37 235	66.014	221 537 945.963	66 494	T_l	1.982
n50p5_1	238 816 029.069	276 528 108.567	318 373 134.859	8.184	37 566	66.706	223 241 477.384	65 810	T_l	−6.521
n50p10_0.6	125 986 402.133	139 170 140.781	159 016 758.976	5.768	13 147	23.496	111 876 371.733	55 338	T_l	−11.199
n50p10_0.9	139 972 788.630	151 470 331.307	167 320 841.400	4.661	12 512	22.296	124 098 184.403	60 684	T_l	−11.341
n50p10_1	141 389 218.598	153 077 203.583	168 858 250.862	3.567	12 154	21.621	144 675 962.447	59 483	T_l	2.324
n50p20_0.6	70 484 204.807	72 929 160.102	78 934 694.735	2.252	8144	14.363	70 359 331.927	59 729	T_l	−0.177
n50p20_0.9	85 418 931.120	88 234 498.766	93 043 256.980	2.042	6267	10.762	84 701 478.613	57 558	T_l	−0.839
n50p20_1	88 997 890.020	92 186 720.344	97 576 459.335	1.523	6147	12.223	87 741 146.608	58 065	T_l	−1.412
n60p5_0.6	218 175 626.464	258 125 560.499	316 647 915.027	8.784	40 849	118.767	219 293 459.438	39 230	T_l	0.512
n60p5_0.9	232 777 240.021	277 403 997.170	325 999 878.259	6.985	41 313	119.752	239 416 074.564	41 440	T_l	2.852
n60p5_1	243 530 155.182	284 627 723.265	340 237 023.255	7.142	42 875	103.83	235 827 088.777	39 723	T_l	−3.163
n60p10_0.6	111 552 173.839	135 640 524.677	156 153 682.114	8.393	27 453	74.717	118 449 727.989	38 829	T_l	6.183
n60p10_0.9	131 535 183.186	155 573 009.955	178 710 992.850	7.492	24 781	65.546	142 004 890.316	38 551	T_l	7.959
n60p10_1	138 075 933.821	166 199 725.250	188 279 038.320	7.516	21 298	58.496	150 272 070.555	38 792	T_l	8.832
n60p20_0.6	93 148 030.262	98 721 548.923	104 105 668.484	3.467	12 658	36.616	75 978 833.177	37 162	T_l	−18.432
n60p20_0.9	88 851 745.778	94 670 586.409	100 836 879.813	3.695	13 685	39.260	91 861 318.193	36 606	T_l	3.387
n60p20_1	93 148 030.262	98 721 548.923	104 105 668.484	3.467	12 658	36.616	95 896 536.510	38 441	T_l	2.950
n70p5_0.6	278 920 783.503	325 462 003.289	364 104 186.778	5.702	30 408	T_l	266 246 083.007	29 591	T_l	−4.544
n70p5_0.9	283 034 791.273	321 409 156.404	386 946 095.614	8.077	29 758	T_l	284 880 299.883	29 393	T_l	0.652
n70p5_1	273 888 410.984	325 256 438.211	388 261 183.934	7.470	29 234	T_l	273 234 070.228	30 151	T_l	−0.238

TABLE 6. continued.

Instance	*Q*-R*p*HLP						*A*-R*p*HLP			
	Best. Obj.	Obj. Val.	Worst. Obj	RSD (%)	NSols.	E.T (s)	Best. Obj.	NSols.	E.T (s)	D-HH S-HH GAP (%)
n70p10_0.6	147 995 519.875	196 405 706.755	230 882 243.670	12.887	28 566	T_l	146 816 665.360	29 003	T_l	−0.796
n70p10_0.9	157 276 215.723	208 255 351.587	250 391 870.069	16.022	28 570	T_l	152 420 742.955	28 202	T_l	−3.087
n70p10_1	165 472 326.626	214 335 121.031	252 828 375.404	14.922	28 284	T_l	156 580 041.463	29 489	T_l	−5.373
n70p20_0.6	78 638 647.164	84 280 943.991	86 865 342.476	2.282	14 321	61.790	79 293 425.895	28 723	T_l	0.832
n70p20_0.9	95 548 177.930	98 565 661.248	104 516 449.103	1.898	13 285	54.392	90 825 083.349	28 840	T_l	−4.943
n70p20_1	98 386 169.494	102 707 553.515	111 322 369.322	2.799	14 191	59.098	101 438 393.260	28 807	T_l	3.102
n80p5_0.6	304 139 283.029	331 657 804.247	378 679 065.464	5.516	20 306	T_l	284 098 390.774	20 748	T_l	−6.589
n80p5_0.9	288 693 885.330	325 826 362.841	396 738 762.084	7.026	21 778	T_l	314 791 010.943	21 289	T_l	9.039
n80p5_1	295 133 142.756	332 533 798.448	440 099 568.923	8.697	21 940	T_l	309 490 054.720	21 324	T_l	4.864
n80p10_0.6	151 880 711.675	172 863 606.135	214 290 172.717	7.377	20 292	T_l	159 791 183.320	20 424	T_l	5.208
n80p10_0.9	180 514 559.392	218 019 727.302	256 763 732.770	11.737	20 476	T_l	172 934 789.102	20 428	T_l	−4.198
n80p10_1	186 149 670.877	221 635 046.852	259 022 663.562	11.419	20 802	T_l	195 033 929.919	21 082	T_l	4.772
n80p20_0.6	83 243 698.738	90 626 666.751	97 211 936.303	4.755	19 273	T_l	85 251 676.676	19 896	T_l	2.412
n80p20_0.9	99 267 554.588	106 705 337.415	117 216 559.646	4.749	18 908	118.800	104 474 708.722	18 306	T_l	5.245
n80p20_1	101 425 785.813	112 028 525.638	122 923 162.325	3.746	19 289	118.706	106 691 490.502	18 277	T_l	5.191
n90p5_0.6	321 982 685.438	409 868 209.781	434 826 654.363	5.931	14 879	T_l	391 035 282.183	14 810	T_l	21.446
n90p5_0.9	359 544 944.625	423 430 862.458	445 669 034.677	4.494	15 106	T_l	402 812 167.418	14 932	T_l	12.033
n90p5_1	325 942 366.820	405 528 285.969	446 579 526.230	8.576	15 052	T_l	409 034 522.716	14 902	T_l	25.492
n90p10_0.6	203 320 551.364	233 592 093.502	279 524 032.625	7.591	15 178	T_l	213 355 522.040	13 631	T_l	4.935
n90p10_0.9	226 478 164.001	248 154 214.132	270 462 399.986	3.371	15 135	T_l	234 327 747.829	14 136	T_l	3.465
n90p10_1	218 553 405.514	245 763 640.548	263 350 539.420	5.161	15 312	T_l	227 529 520.631	13 543	T_l	4.107
n90p20_0.6	102 096 795.271	113 105 472.387	123 934 577.104	5.005	14 887	T_l	111 153 548.877	13 357	T_l	8.870
n90p20_0.9	123 250 455.106	132 280 389.036	148 355 525.116	4.484	15 682	T_l	124 069 295.838	13 028	T_l	0.664
n90p20_1	129 249 629.030	138 490 854.448	149 597 020.064	3.860	15 781	T_l	129 770 242.453	13 416	T_l	0.402
n100p5_0.6	350 750 185.785	419 351 639.324	461 279 433.829	6.813	11 108	T_l	413 744 534.857	10 615	T_l	17.959
n100p5_0.9	362 286 231.955	430 618 969.792	461 902 168.613	5.793	11 779	T_l	380 186 066.050	10 328	T_l	4.940
n100p5_1	347 769 592.209	420 779 656.452	461 847 202.147	7.362	12 132	T_l	392 244 521.813	10 325	T_l	12.788
n100p10_0.6	265 570 827.647	326 342 273.358	394 058 432.775	9.832	11 315	T_l	322 953 442.096	11 067	T_l	21.607
n100p10_0.9	303 917 689.784	352 244 171.665	415 946 074.765	9.326	11 405	T_l	333 524 774.811	11 123	T_l	9.741
n100p10_1	295 466 906.979	348 209 230.254	402 673 130.978	8.570	11 431	T_l	335 683 822.833	11 123	T_l	13.611
n100p20_0.6	126 367 344.630	160 834 003.206	192 509 037.339	11.683	11 155	T_l	149 724 213.860	10 306	T_l	18.483
n100p20_0.9	144 448 497.750	174 694 130.275	215 383 297.727	10.912	10 952	T_l	155 649 882.401	11 515	T_l	7.754
n100p20_1	153 457 447.022	179 623 658.904	207 111 424.314	8.565	11 221	T_l	168 806 900.499	10 958	T_l	10.002

best solution found by CPLEX. In Table 6, the "GAP" column is calculated based on the best-known solutions of the Q-learning hyper-heuristic and the A-learning hyper-heuristic (a negative GAP indicates that A-learning produced a better solution than Q-learning).

The time limit T_{limit} was set to 60 s (1 min) for small size instances and 120 s (2 min) for large size instances. For the Q-learning hyper-heuristic each instance has been executed 30 times.

From Table 5, it can be observed that both approaches often find the optimal solution when it exists, with a success rate of 78.787% for Q-learning and 69.69% for A-learning. The largest gap recorded among all instances for Q-learning and A-learning is 1.737% and 7.275%, respectively, for the same instance n14p6_1. On the other hand, the smallest gap obtained from Q-learning and A-learning is −20.682% and −13.369% (seen in n20p7_0.6). While both approaches yield good results, Q-learning remains the superior choice.

Figure 8a illustrates the computational time for the Q-learning approach. The time increases as the size of the instances grows. This confirms the observation made in Section 5.1 that there is no correlation between α and the computational execution time. On the other hand, for the same instance, the algorithm requires more time to converge when the number of hub nodes is small.

As observed in Figure 8b, the Q-learning algorithm encounters difficulties when the number of nodes reaches around 70, especially when the value of p is small. These difficulties are reflected in the inconsistent results obtained. For instance, in the case of n100p5, we obtained a lower objective value with $\alpha = 1$ compared to $\alpha = 0.6$. Similar challenges can be observed in the A-learning approach, as shown in Figure 9b. The algorithm struggles to find satisfactory results, particularly for instances with $p = 5$. It is worth noting that the A-learning approach is deterministic, meaning it follows the same execution path each time. Therefore, one might expect that obtaining a better solution is possible with a smaller value of α. However, in reality, this is not always the case, highlighting the impact of α on the search landscape.

The A-learning approach is characterized by its high computational time, as depicted in Figure 9a. The algorithm often reaches the time limit rather than converging, mainly due to the inclusion of the backtracking mechanism. The algorithm is designed to continuously search for better areas in the search space until either the time limit is reached or no significant improvements are found. This iterative process contributes to the increased time consumption of the A-learning approach.

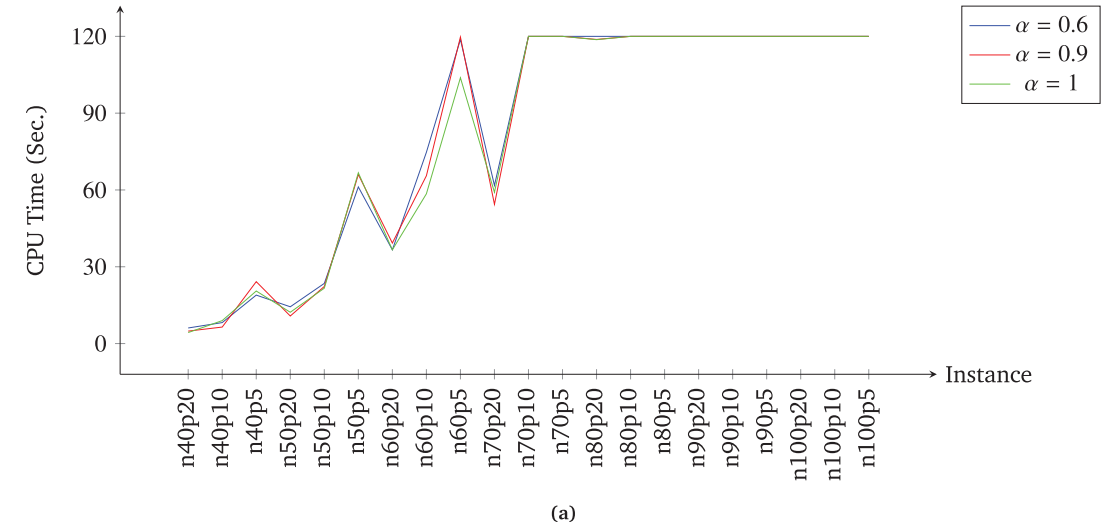
Table 6 provides a comprehensive overview of the two proposed hyper-heuristics for medium and large size instances, where finding the optimal solution becomes challenging.

In terms of the number of feasible solutions found (Fig. 10a), the A-learning procedure exhibits a higher count on medium size instances, which can be attributed to the early convergence of the Q-learning approach. However, for instances with the same execution time, both approaches yield a similar number of solutions, as they share the same main components (the low-level heuristics).

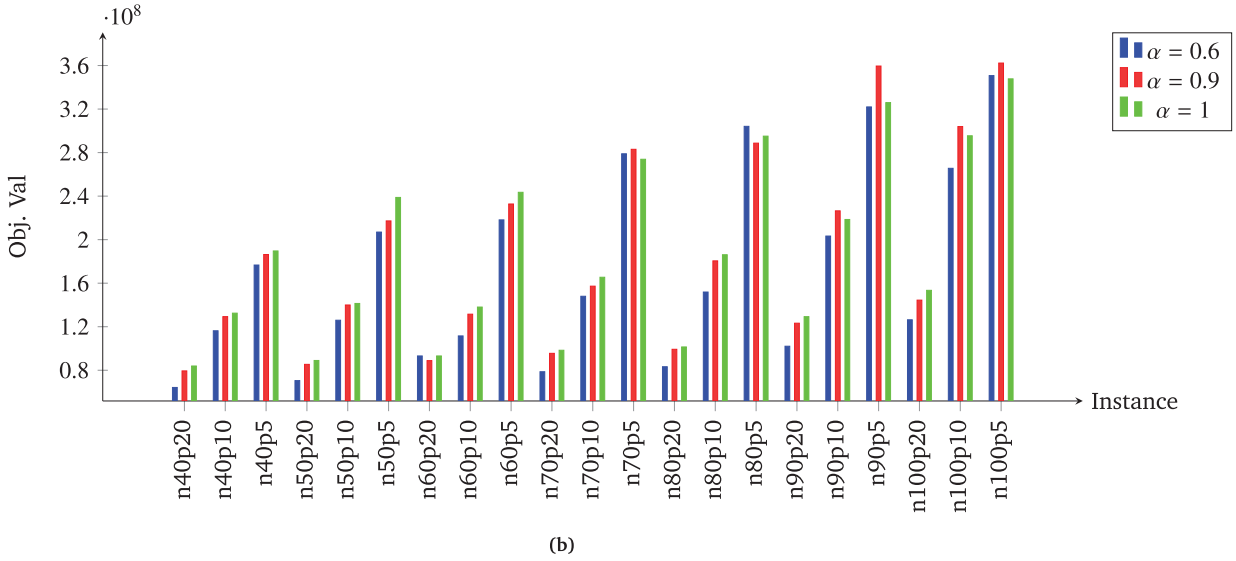
Regarding solution quality, although the A-learning approach produces good results, the Q-learning approach generally outperforms it (63.492% improvement, as shown in Tab. 6). Figure 10b demonstrates that for medium size instances, A-learning either provides a better solution or remains close to the best Q-learning solution found. However, as the instance size increases, A-learning struggles to keep up with the performance of the Q-learning approach.

Furthermore, Figure 10c compares the average solution values obtained by Q-learning after 30 executions with the best solutions found by A-learning. It is evident that A-learning consistently outperforms Q-learning on average (96.825% improvement, as indicated in Tab. 6).

Both algorithms demonstrate their effectiveness and are of great interest. The Q-learning approach proves to be more efficient, but stochastic in nature, while the A-learning approach is deterministic and stable, but more time-consuming.



(a)



(b)

FIGURE 8. Numerical results of the Q-learning approach. (a) Execution time. (b) Objective value.

6. Summary and conclusion

In this paper, we introduce the RpHLP, a new network architecture for hub location problems. This architecture incorporates a round-trip-based structure and introduces runaway nodes as an additional node class. The inclusion of runaway nodes adds redundancy to the network, offering alternative shipping routes with minimal setup costs. The primary objective of runaway nodes is to enhance connectivity, mitigate the consequences of hub edge failures or disruptions, and facilitate efficient exchanges within the network.

We have developed a MIP formulation for addressing the problem under investigation. However, due to the inherent complexity of the problem, the MIP formulation could only be solved by CPLEX for small-sized instances. To tackle medium and large-sized instances, we have proposed two hyper-heuristics as alternative approaches. A hyper-heuristic comprises a collection of low-level heuristics and a selection method, where the selection method determines which low-level heuristic to employ at each step in order to enhance the solution. In

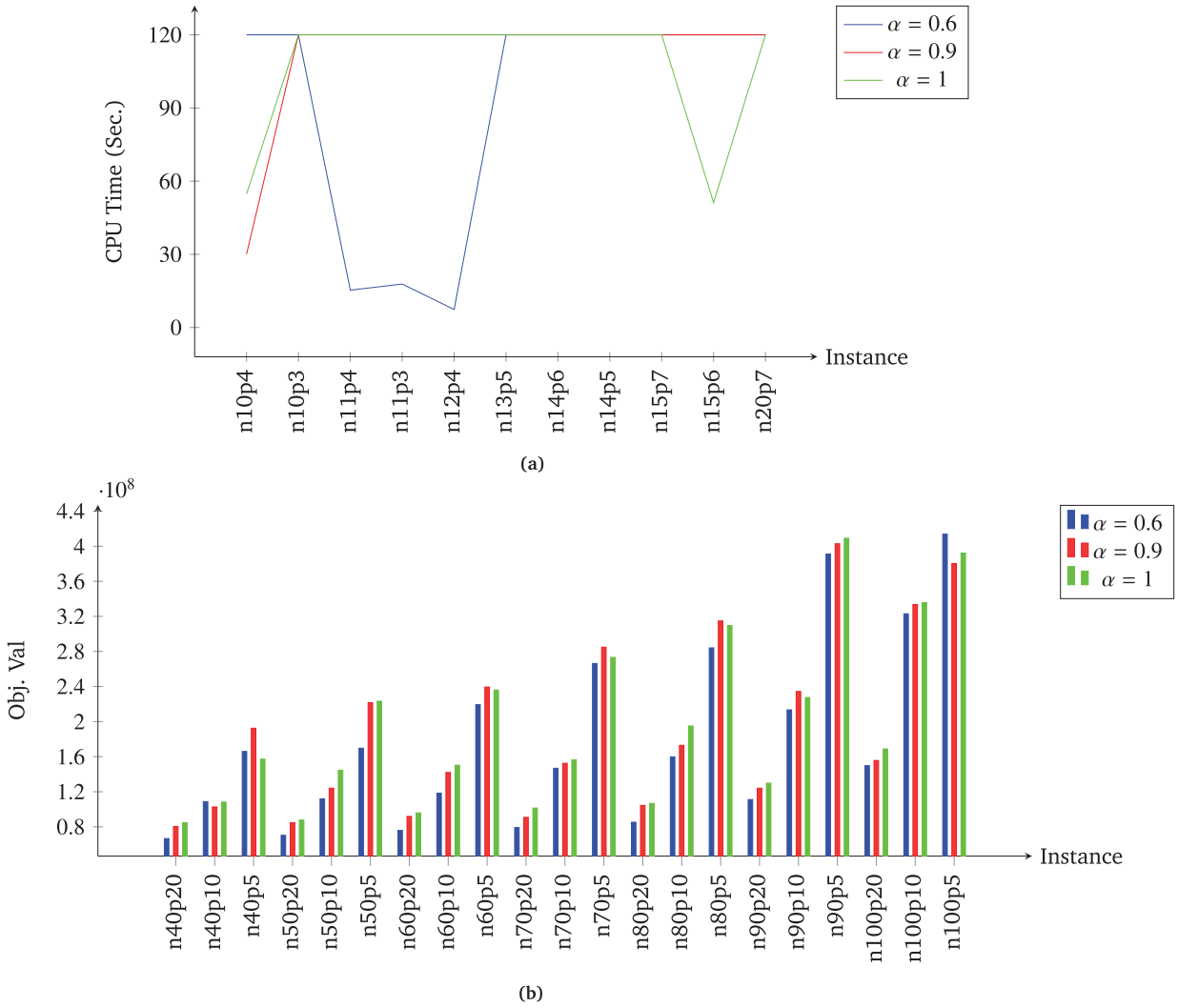


FIGURE 9. Numerical results of the A-learning approach. (a) Execution time. (b) Objective value.

our study, we have introduced two hyper-heuristics, both employing a reinforcement learning selection scheme. However, there are notable distinctions between the two. The first hyper-heuristic adopts a stochastic approach and is built upon Q-learning, while the second hyper-heuristic is deterministic and utilizes a novel selection method called A-learning.

The computational experiments conducted in our study have showcased the effectiveness of both approaches. Specifically, for small-sized instances, the hyper-heuristics have proven to be highly successful in locating the optimal solution. The Q-learning method has demonstrated its ability to yield excellent solutions within a reasonable timeframe. On the contrary, the A-learning approach is more time-consuming and requires additional time to generate solutions of good quality.

In general, both approaches are effective. A-learning, being deterministic, strikes a good balance between execution time and solution quality. On the other hand, the Q-learning approach is faster but may require multiple runs to achieve a high-quality solution.

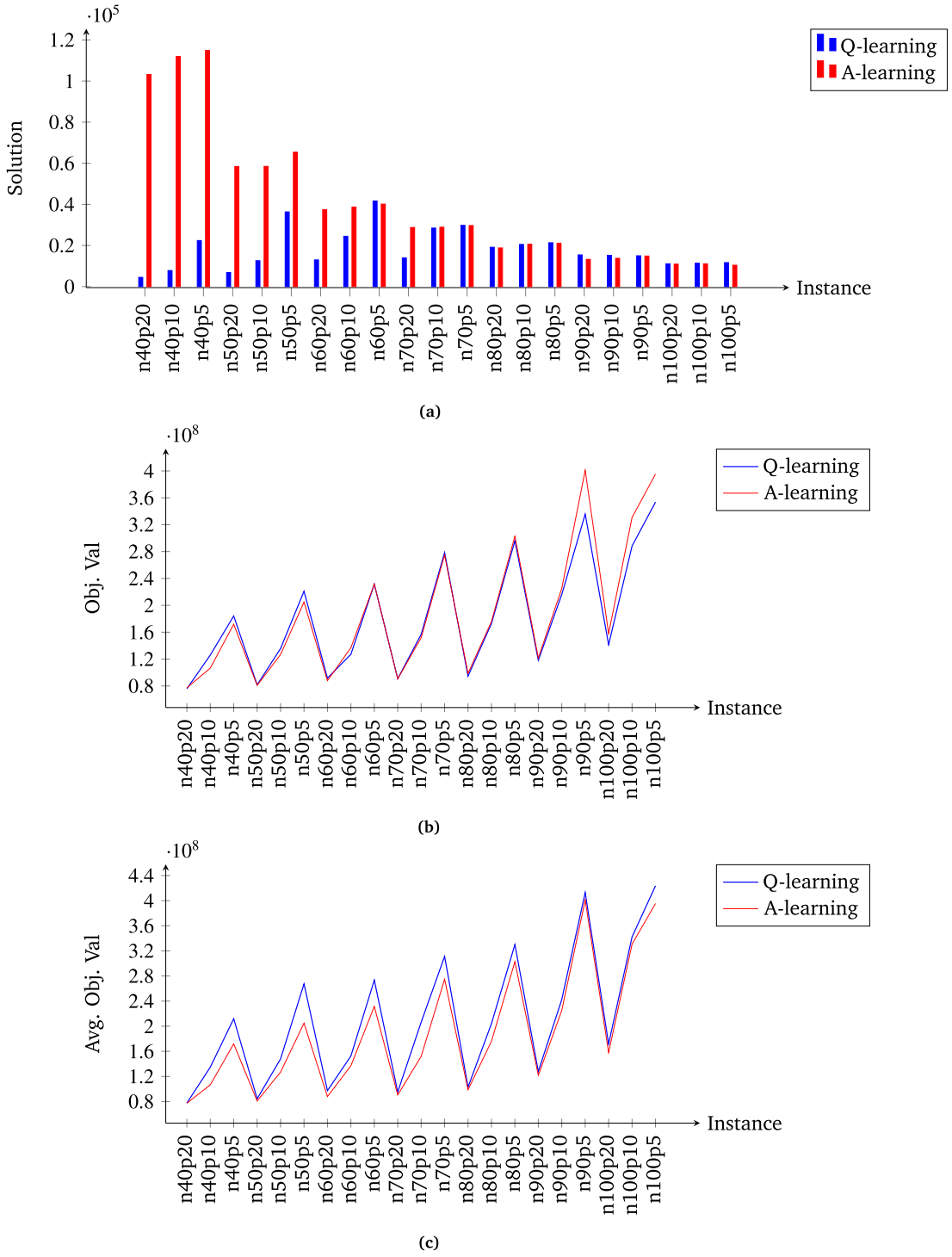


FIGURE 10. Numerical results comparison between Q-learning and A-learning. (a) Number of solutions found. (b) Best objective value comparison. (c) Average objective value comparison.

For future work, there are several avenues we plan to explore. Firstly, we aim to enhance the mathematical model by identifying additional classes of valid inequalities, which can help improve the quality of solutions obtained. This would involve investigating different structural properties and incorporating them into the formulation.

Additionally, we are interested in exploring the hybridization of meta-heuristic and exact methods. Combining the strengths of these two approaches can potentially yield high-quality solutions while maintaining computational efficiency. This hybrid approach could leverage the exploration capabilities of meta-heuristics and the precision of exact methods.

Furthermore, given the promising results of the hyper-heuristic approach, we intend to study and develop other learning mechanisms. By exploring more sophisticated selection approaches and expanding the low-level heuristic portfolio, we can further enhance the performance and adaptability of the hyper-heuristics.

Overall, our future work aims to improve the mathematical model, investigate hybridization strategies, and enhance the hyper-heuristic approach through advanced learning mechanisms and diverse low-level heuristic options.

Acknowledgements

The authors would like to thank the Directorate General of Scientific Research and Technological Development (DGRSDT), Ministry of Higher Education and Scientific Research for their support in this work. In addition, work of last author has been supported by the PGMO, the Gaspard Monge Program for Optimisation and Operational Research in the framework of BENMIP project.

References

[1] S.A. Alumur, H. Yaman and B.Y. Kara, Hierarchical multimodal hub location problem with time-definite deliveries. *Transp. Res. Part E: Logistics Transp. Rev.* **48** (2012) 1107–1120.

[2] N. Azizi, Managing facility disruption in hub-and-spoke networks: formulations and efficient solution methods. *Ann. Oper. Res.* **272** (2019) 159–185.

[3] O. Berman, Z. Drezner and G.O. Wesolowsky, The transfer point location problem. *Eur. J. Oper. Res.* **179** (2007) 978–989.

[4] E.K. Burke, M.R. Hyde, G. Kendall, G. Ochoa, E. Özcan and J.R. Woodward, A Classification of Hyper-Heuristic Approaches: Revisited. Springer International Publishing, Cham (2019) 453–477.

[5] J.F. Campbell, Location and allocation for distribution systems with transshipments and transportion economies of scale. *Ann. Oper. Res.* **40** (1992) 77–99.

[6] J.F. Campbell, Integer programming formulations of discrete hub location problems. *Eur. J. Oper. Res.* **72** (1994) 387–405.

[7] S.R. Cardoso, A.P. Barbosa-Póvoa, S. Relvas and A.Q. Novais, Resilience metrics in the assessment of complex supply-chains performance operating under demand uncertainty. *Omega* **56** (2015) 53–73.

[8] G. Carello, F.D. Croce, M. Ghirardi and R. Tadei, Solving the hub location problem in telecommunication network design: a local search approach. *Networks* **44** (2004) 94–105.

[9] P. Carroll, B. Fortz, M. Labbé and S. McGarraghy, Improved formulations for the ring spur assignment problem, in Network Optimization, edited by J. Pahl, T. Reiners and S. Voß. Springer Berlin Heidelberg, Berlin, Heidelberg (2011) 24–36.

[10] S. Çetiner, C. Sepil and H. Süral, Hubbing and routing in postal delivery systems. *Ann. Oper. Res.* **181** (2010) 109–124.

[11] S. Chaharsooghi, F. Momayezi and N. Ghaffarinasab, An adaptive large neighborhood search heuristic for solving the reliable multiple allocation hub location problem under hub disruptions. *Int. J. Ind. Eng. Comput.* **8** (2016) 191–202.

[12] I. Contreras, M. Tanash and N. Vidyarthi, Exact and heuristic approaches for the cycle hub location problem. *Ann. Oper. Res.* **258** (2017) 655–677.

[13] P.I. Cowling, G. Kendall and E. Soubeiga, A hyperheuristic approach to scheduling a sales summit, in Practice and Theory of Automated Timetabling III, PATAT '00. Springer (2001) 176–190.

[14] K. Danach, S. Gelareh and R.N. Monemi, The capacitated single-allocation p-hub location routing problem: a Lagrangian relaxation and a hyper-heuristic approach. *EURO J. Transp. Logistics* **8** (2019) 597–631.

[15] E.M. de Sá, I. Contreras and J.-F. Cordeau, Exact and heuristic algorithms for the design of hub networks with multiple lines. *Eur. J. Oper. Res.* **246** (2015) 186–198.

[16] E.M. de Sá, I. Contreras, J.-F. Cordeau, R. Saraiva de Camargo and G. de Miranda, The hub line location problem. *Transp. Sci.* **49** (2015) 500–518.

[17] J. Denzinger, M. Fuchs and M. Fuchs, High performance ATP systems by combining several AI methods, in Proceedings of the 15th International Joint Conference on Artifical Intelligence. Vol. 1 of *IJCAI'97*. Morgan Kaufmann Publishers Inc. (1997) 102–107.

[18] J. Ebery, M. Krishnamoorthy, A. Ernst and N. Boland, The capacitated multiple allocation hub location problem: formulations and algorithms. *Eur. J. Oper. Res.* **120** (2000) 614–631.

[19] A.T. Ernst and M. Krishnamoorthy, Efficient algorithms for the uncapacitated single allocation p-hub median problem. *Location Sci.* **4** (1996) 139–154.

[20] S. Gelareh and S. Nickel, Hub location problems in transportation networks. *Transp. Res. Part E: Logistics Transp. Rev.* **47** (2011) 1092–1111.

[21] S. Gelareh, N. Maculan, P. Mahey and R.N. Monemi, Hub-and-spoke network design and fleet deployment for string planning of liner shipping. *Appl. Math. Modell.* **37** (2013) 3307–3321.

[22] S. Gelareh, R.N. Monemic and F. Semet, Capacitated bounded cardinality hub routing problem: model and solution algorithm. Technical report. Preprint arXiv:1705.07985 (2017).

[23] O. Kemmar, K. Bouamrane and S. Gelareh, Hub location problem in round-trip service applications. *RAIRO-Oper. Res.* **55** (2021) S2831–S2858.

[24] H. Kim and M. O'Kelly, Reliable p-hub location problems in telecommunication networks. *Geog. Anal.* **41** (2009) 283–306.

[25] M.J. Kuby and R.G. Gray, The hub network design problem with stopovers and feeders: the case of federal express. *Transp. Res. Part A: Policy Pract.* **27** (1993) 1–12.

[26] M. Mohammadi, R. Tavakkoli-Moghaddam, A. Siadat and Y. Rahimi, A game-based meta-heuristic for a fuzzy bi-objective reliable hub location problem. *Eng. App. Artif. Intell.* **50** (2016) 1–19.

[27] R.N. Monemi and S. Gelareh, The ring spur assignment problem: new formulation, valid inequalities and a branch-and-cut approach. *Comput. Oper. Res.* **88** (2017) 91–102.

[28] R.N. Monemi, S. Gelareh, A. Nagih, N. Maculan and K. Danach, Multi-period hub location problem with serial demands: a case study of humanitarian aids distribution in Lebanon. *Transp. Res. Part E: Logistics Transp. Rev.* **149** (2021) 102201.

[29] M. O'Kelly, Hub facility location with fixed costs. *Papers Regional Sci.* **71** (1992) 293–306.

[30] I. Rodriguez-Martín, J.J. Salazar González and H. Yaman, A branch-and-cut algorithm for the hub location and routing problem. *Comput. Oper. Res.* **50** (2014) 161–174.

[31] I. Rodriguez-Martín, J.J. Salazar González and H. Yaman, The ring k-rings network design problem: model and branch-and-cut algorithm. *Networks* **68** (2016) 130–140.

[32] B. Rostami, N. Kämmerling, C. Buchheim and U. Clausen, Reliable single allocation hub location problem under hub breakdowns. *Computers & Operations Research* **96** (2018) 15–29.

[33] D. Skorin-Kapov, J. Skorin-Kapov and M. O'Kelly, Tight linear programming relaxations of uncapacitated p-hub median problems. *Eur. J. Oper. Res.* **94** (1996) 582–593.

[34] UNCTAD, Review of maritime transport, in United Nations Conference on Trade and Development, New York and Geneva (2018).

[35] M. Yahyaei, M. Bashiri and M. Randall, A model for a reliable single allocation hub network design under massive disruption. *Appl. Soft Comput.* **82** (2019) 105561.

[36] H. Yaman, B.Y. Kara and B.C. Tansel, The latest arrival hub location problem for cargo delivery systems with stopovers. *Transp. Res. Part B: Methodol.* **41** (2007) 906–919.

[37] K. Yang, Y. Liu and G. Yang, An improved hybrid particle swarm optimization algorithm for fuzzy p-hub center problem. *Comput. Ind. Eng.* **64** (2013) 133–142.

[38] M. Zhalechian, S.A. Torabi and M. Mohammadi, Hub-and-spoke network design under operational and disruption risks. *Transp. Res. Part E: Logistics Transp. Rev.* **109** (2018) 20–43.

[39] W. Zhong, Z. Juan, F. Zong and H. Su, Hierarchical hub location model and hybrid algorithm for integration of urban and rural public transport. *Int. J. Distrib. Sensor Netw.* **14** (2018) 155014771877326.