RAIRO-Oper. Res. 55 (2021) S2831–S2858
https://doi.org/10.1051/ro/2020125

RAIRO Operations Research
www.rairo-ro.org

HUB LOCATION PROBLEM IN ROUND-TRIP SERVICE APPLICATIONS

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

Abstract. In this paper, we introduce a new hub-and-spoke structure for service networks based on round-trips as practiced by some transport service providers. This problem is a variant of Uncapacitated Hub Location Problem wherein the spoke nodes allocated to a hub node form round-trips (cycles) starting from and ending to the hub node. This problem is motivated by two real-life practices in logistics wherein *runaway* nodes and *runaway connections* with their associated economies of scale were foreseen to increase redundancy in the network. We propose a mixed integer linear programming mathematical model with exponential number of constraints. In addition to the separation routines for separating from among exponential constraints, we propose a hyper-heuristic based on reinforcement learning and its comparable counterpart as a variable neighborhood search. Our extensive computational experiments confirm efficiency of the proposed approaches.

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

Received January 31, 2020. Accepted October 31, 2020.

1. Introduction

While hub-and-spoke operations have almost always been practiced in the modern transportation industry, in certain circumstances, these structures are the only possible way ahead. This work has been motivated by two consultancy projects that have been carried out over the last 5 years: a set of maritime projects and an aid distribution problem arising in the context of Syrian refugees in Lebanon.

We are dealing with structures depicted in Figure 1 (see [21]), which frequently appear in various and sometimes very different areas including supply chain logistics as well as telecommunications.

However, this structure has some weaknesses, as every potential failure or disruption can render an important part of the network unreachable. In the following, we point out two cases that we have encountered in our collaboration with the stakeholders from the industry.

Liner shipping industry. In liner shipping wherein vessels operate round-trips, this notion has received much more attention. About 80 per cent of the world trade by volume and more than 70 per cent of world trade in value is carried by sea, making liner shipping the basis of world trade [46]. Liner shipping operates on hub-and-spoke structures where major terminals are considered as global or regional hubs and the smaller

Keywords. Hub location problem, liner shipping, runaway node, branch-and-cut, hyper-heuristic, variable neighborhood search, reinforcement learning, k-means.

[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`, `omke1993@gmail.com`

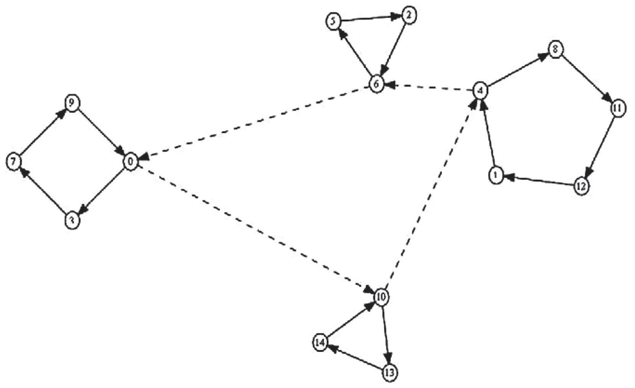
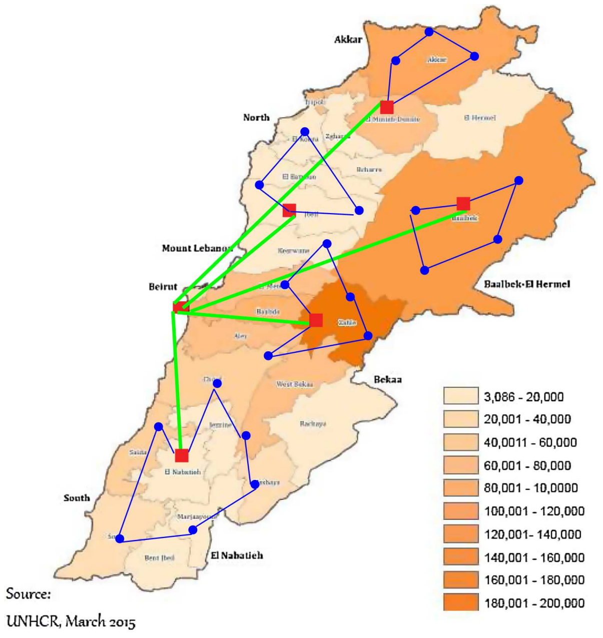


FIGURE 1. A typical hub-and-spoke based round-trip service network.

ports are the spoke/feeder ports. Almost a decade ago, when the extremely costly and spacious mega-vessels of 18 000 TEU (twenty-foot equivalent) such as Emma Maersk have emerged and were deployed, both ports and liner operators were really concerned. The string managers (those in charge of designing seasonal round-trip service network in shipping companies) were worried that such costly and huge vessels may have to be deployed under-utilized (at least during some periods) without being able to efficiently exploit their economies of scale. The port operators or major terminals were worried of being now exposed to a much higher risk should any possible disruptions occur. If such a disruption happens and delays go beyond the normal and expected turnaround time for the vessels (which would increase operation costs for liners), it would causes dissatisfaction for the liners and jeopardize the ports competitiveness in their business. Such large vessels could not call many ports due to the draught requirement and moreover, many European ports did not have the capacity (not even potential for physical expansion) to serve such huge vessels. The solution was mainly in the hand of network operators rather than hub operators, to introduce sufficient redundancy such that a reasonable part of the risks could be mitigated. This could be achieved by providing possibility of cross-rotation *escape* operation through some nominated ports. One way to achieve this was to make sure that the feeder round-trips do not rely on (not monopolized by) one and only one hub port and if the single hub fails (due to strikes, threats, etc.), the operation can still be re-directed through *runaway connections* at a minimal overhead operational cost.

Distribution of humanitarian aids. A very recent case we have recently worked on is related to distribution of humanitarian aids among the UNHCR-recognized displaced Syrians who fled out of the war situation and scattered all across Lebanon as one of the major host countries. A central depot in Beirut suburbs, supplies a set of hub locations identified among the demand nodes. 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 happens 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* and *runaway connection*.

We need to introduce some redundancy in the hub-and-spoke structure in Figures 1 and 3 in such a way that in an (un)likely event of disruption in the hub-level elements, the service provision remains possible. Such a redundancy can be achieved by providing alternative paths for spoke nodes on the cycle (feeder) of a given hub node. We achieve this by connecting cycles *via* a third type of nodes we refer to as *runaway* nodes. These nodes do not necessarily incorporate very sophisticated and expensive hub-type facilities, yet the infrastructure is upgraded to outperform spoke nodes and enable them to process larger volumes in the case of necessity.



(a)

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.*

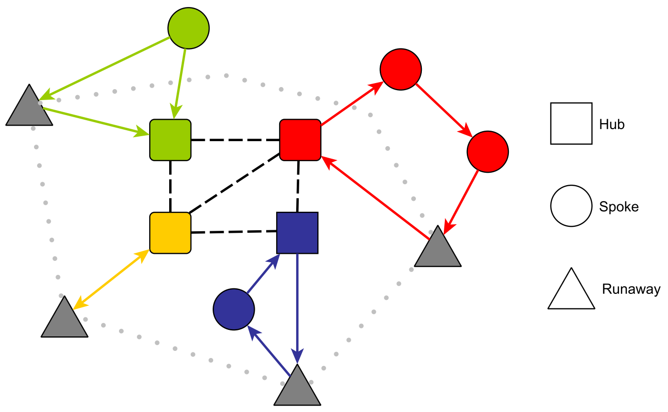


FIGURE 3. A typical structure of network in a runaway hub-and-spoke structure.

1.1. Literature review

This work can be considered as a variant of the uncapacitated single allocation hub location problem (USAHLP). In the following, we review some of the closely related contributions in the literature, with particular attributes: (1) single allocation scheme, (2) cyclic spoke-level sub-networks, and (3) solution approaches based on (meta-)heuristics and hyper-heuristics.

Danach *et al.* [16] proposed a Mixed Integer Programming (MIP) formulation, a Lagrangian relaxation and a hyper-heuristic for a hub location and routing problem. In this study, the single allocation scheme is considered where the capacity is defined on the volume of flow circulating on a spoke-level route that needs to respect the capacity of transporters available on it.

Azizi [2] introduced a MIP formulation and a Particle Swarm Optimization (PSO) algorithm for the Uncapacitated Single Allocation p-hub Location Problem under risk of hubs disruption. The author constructs networks in which every single demand point has a backup hub to be served from in case of disruption.

Zhong *et al.* [52] tackled the hierarchical hub location model and proposed a MIP formulation for the problem with hub capacity constraints as well as a hybrid meta-heuristic (genetic algorithm and tabu search).

A two-stage formulation for reliable Single Allocation Hub Location Problem was introduced in Rostami *et al.* [42] with a Benders decomposition approach to solve large scale instances. In this study, whenever a hub breaks down, its corresponding flow is rerouted *via* a single backup hub.

Monemi and Gelareh [33] proposed a 2-index integer programming (IP) formulation and a branch-and-cut algorithm with some classes of valid inequalities for the Ring Spur Assignment Problem introduced by Carroll *et al.* [9]. This problem arises in the design of next-generation telecommunications networks but shares some common features with this work.

A 2-index model and a branch-and-cut algorithm based on Benders decomposition are given in Gelareh *et al.* [22] for the Bounded Cardinality Capacitated Hub Routing Problem (BCCHRP) with route capacity constraints.

In Chaharsooghi *et al.* [11], the reliable uncapacitated multiple allocation hub location problem under hub disruptions is considered. The author proposed a two-stage stochastic model and an adaptive large neighborhood search meta-heuristic as a non-exact approach. In this problem, whenever a hub fails, the spokes allocated to that hub, are either reallocated to other hubs that are still working or a penalty is paid in the case they do not receive any service due to the high reallocation costs.

Contreras *et al.* [12] presented a MIP formulation and a branch-and-cut algorithm for the Cycle Hub Location Problem (CHLP). A greedy randomized adaptive search procedure (GRASP) is developed to obtain feasible solutions for large-scale instances of the CHLP.

Rodriguez-Martin *et al.* [41] proposed a branch-and-cut algorithm for the problem of designing a two level network where the upper level consists of a backbone ring network connecting the hub nodes, and the lower level is formed by access ring networks that connect the spoke nodes to the hub nodes. It is a purely location problem and does not incorporate any flow.

Mohammadi *et al.* [32] proposed a bi-objective mixed-integer non-linear programming and an evolutionary algorithm for the Single Allocation p-hub Center-Median Problem under data uncertainty, where the objective is to obtain a reliable network. Interested readers are also referred to the recent network reliability studies: Yahyaei et al. [48], Zhalechian *et al.* [51] and Cardoso *et al.* [7].

In Martins de Sá *et al.* [29] a Benders decomposition algorithm and several metaheuristics were proposed for the Hub Line Location Problem (HLLP) introduced in Martins de Sá *et al.* [30]. This problem is tailored for public transportation systems.

The hub location and routing problem is studied in Rodriguez-Martin *et al.* [40]. A MIP formulation and a branch-and-cut algorithm are proposed. In this work, the capacity constraint is defined in terms of number of spokes per cycle, and the number of hubs is assumed to be an exogenous information.

Gelareh *et al.* [21] proposed a hub-and-spoke structure with one central hub cycle and spoke-level (feeder) cycles attached to every hub node. They use Lagrangian decomposition approach equipped with a Lagrangian heuristic.

Yang *et al.* [50] proposed a hybrid particle swarm optimization (PSO) algorithm for the p-Hub Center Problem in fuzzy environments by combining PSO, genetic operators and a local search (LS).

Alumur *et al.* [1] proposed a MIP for a hierarchical multi-modal hub location problem, where two types of hub nodes and hub edges are considered (for ground and air transportation) and a time definite delivery service.

Gelareh and Nickel [20] proposed a MIP with a Benders decomposition method and a greedy heuristic to solve the Uncapacitated Multiple Allocation Hub Location Problem adapted for urban transport and liner shipping network design.

Çetiner *et al.* [10] proposed an iterative two stages heuristic, which firstly locates the hub nodes and then design routes using traveling salesman problem heuristics. The Turkish postal delivery system data is used as a case study.

A Reliable p-hub Location Problem was proposed in Kim and O'Kelly [26] for telecommunication networks, where two mathematical model formulations are given considering the single and multiple assignment schemes.

In Berman *et al.* [3], a model is proposed in which, given n nodes, the objective is to locate p helicopter pads and one facility (hospital) to minimize the total time (minisum) or the maximum time (minimax).

Yaman *et al.* [49] proposed a minimax mathematical model for ground-based cargo delivery system with stopovers for the latest arrival hub location problem.

A local search approach and different meta-heuristic algorithms were proposed in Carello *et al.* [8] for the capacitated single allocation hub location problem (CSAHLP) where the hubs are transit nodes and the spokes are access nodes.

A branch-and-bound procedure was proposed in Ebery *et al.* [18] for the capacitated multiple allocation hub location problem applied to postal networks together with an efficient heuristic algorithm using shortest paths to obtain the upper bound.

Campbell [6] presented the first linear programming formulations for Multiple/Single Allocation Uncapacitated/Capacitated Hub Location Problems. This work has introduced a fundamental set of hub location problems that have served as a building block for a lot of research works. Skorin-Kapov *et al.* [44] proposed an improvement to the best-known MIP formulation for the Single Allocation p-hub Median Problem (SApHMP).

Kuby and Gray [27] developed a MIP formulation to design the least cost single-hub air network assuming that the hub location is predetermined.

O'Kelly [35] introduced single allocation hub location problem (SAHLP) with fixed costs making the number of hubs an endogenous part of the problem and proposed a quadratic integer formulation. Campbell [5] introduced the first model for the multiple allocation problem.

For most part, the works tackling Hub Location Problems in the literature consider the standard hub and spoke network structures. In addition, most of the non-exact approaches proposed in the previous studies are generally heuristics and meta-heuristics, leaving great potential of the hyper-heuristic approaches almost untouched in this context.

This paper proposes an extension to the work in Danach *et al.* [16] and therefore shares some elements. However, our work is distinguished from Danach *et al.* [16] as follows: in Danach *et al.* [16], the hub-level network is a complete subgraph, while in here, it is an endogenous part of the problem and not necessarily a complete subgraph. In the aforementioned work, there are two sorts of nodes and connections (spoke and hub nodes), while in this study we define an additional type of node/connection and we use the conventional term from practice to refer to it as *runaway*. Runaway nodes and edges (the edges connecting runaway nodes) provide alternative paths and augment the redundancy to meet the origin-destination demands. Finally, in Danach *et al.* [16], every origin-destination demand passes through the hub-level network while this is not necessary the case in the current work.

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

1.2. Contribution and scope

The problem in this paper is motivated by a couple of real-life cases wherein the spoke-level network structure resembles the one depicted in Figures 2 and 3. In the real practice, we seek to upgrade such structures to introduce more redundancy through runaway nodes such as the one in Figure 3. The dashed lines (runaway connections) connecting runaway nodes (nodes that have attributes somewhere between being a sophisticated hub node and a simple spoke node) lying on the spoke-level round-trips introduce some level of redundancy and cope with potential failures (strikes, unforeseen threats, etc.) at the hub nodes or the connections to/from

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

Work	Alloc.	Num. hubs	Objective	Capacity	Solution method
Danach *et al.* [16]	SA	Exogenous	Time (transit + transshipment)	Yes	MIP + Lagrangian relaxation + Hyper-heuristic
Zhong *et al.* [52]	SA	Endogenous	Cost	Yes	MIP + Meta-heuristic
Contreras *et al.* [12]	SA	Exogenous	Cost	No	MIP + Branch-and-cut + Meta-heuristic
Rodriguez-Martin *et al.* [41]	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-Martin *et al.* [40]	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 + Meta-heuristic + Hyper-heuristic

the hub nodes. In contrast to a backup hub node and sophisticated hub edges (connecting such backup node to the hub-level network) which are expensive to setup and maintain, the runway nodes are some enhancements to the normal spoke nodes and are only connected to one or two peer runaway nodes using connections that are somewhere between the performance of a hub edge and a spoke edge, making it much cheaper to operate and maintain until the disruption is over. Moreover, when the hub-level network is targeted by a disruptive occurrence and becomes temporarily unavailable, this notion provides excess capacity somewhere outside the focus of disruption and in a decentralized fashion – *i.e.* at the spoke level network. We have carried out a thorough investigation of the matter in two cases from the real practice and a close collaboration with the relevant decision makers from the sectors. From the theoretical and modeling point of view, this model opens a new perspective in including redundancy in the hub-and-spoke structures and generalizes some of the previous work (including our work in [21]). We propose the first (exponential) mixed integer programming formulation for this problem, an efficient separation routine to separate from among the exponentially many constraints, and an efficient heuristic-selection hyper-heuristic delivering high quality solutions in a very reasonable time. Extensive computational experiments on randomly generated instances of various sizes, confirm computational efficiency of the proposed solution framework and the viability of the approach.

This paper is organized as follows: The problem is formally described in Section 2 and a mathematical model with an exponential number of constraints is proposed in Section 3. Section 4 provides a detailed description and elaborates on the components of the hyper-heuristic approach (initial solution, selection method, low-level heuristics and the main procedure), the metaheuristic and the separation routine. Section 5 reports and discusses the results of our computational experiments. In Section 6, we summarize, draw conclusions and provide suggestions for further research directions.

2. Problem statement

The p-Hub Location Problem with Runaway (pHLPwR) (Fig. 3) can be formally described as in the following: *Given a set of nodes V where $|V| = n$, a cost matrix C where c_{ij} is the cost per unit of flow on the edge i to j, a flow/demand matrix W where w_{ij} is the flow to be sent from i to j and the fixed costs of setting up* *hub nodes and runaway nodes. The fixed costs of setting up hub and runaway nodes, hub edges, spoke arcs and runaway connections are denoted by F_k, G_k, I_{kl}, S_{kl} and T_{kl}, respectively. The problem is to find p nodes called hub nodes and establish the hub-level network using hub edges, connecting hub nodes in such a way that the hub-level network remains connected with undirected connections (edges). The remaining $n - p$ spoke nodes will be allocated to different hub nodes from among the p hubs. The spoke level nodes form directed rotation services starting from the hub node, visiting all the spoke nodes allocated to it and returning to the hub node. There will be at least two spoke nodes allocated to every hub node unless practical real-life situations do not permit this, in which case the sole allocated spoke node is called an isolated spoke node and is spurred at the hub node. A runaway node is an upgraded spoke node chosen from among the spoke nodes allocated to a given hub in order to provide a second access to outside the rotation in addition to the one provided by the hub node. Every cycle will include one runaway node, which is connected to two other such nodes on different rotations. The runaway connections are undirected. A discount factor, α, represents the economies of scale at the hub-level and a λ counterpart ($\alpha \leq \lambda \leq 1$ as the runaway connections do not provide the full function of a hub edge and as a result, the factor of economies of scale are less advantageous for the runaways) applies to the runaway connections. The objective is to find the optimal structure minimizing the total costs associated to setting up facilities and transportation costs on such a structure.*

3. Mixed-integer linear programming formulations

The variables of model follow: $x_{ij} = 1$ $(\forall i \neq j)$, if node i is allocated to hub j, 0, otherwise; $h_i = 1$ if i is a hub, 0, otherwise; $y_{ij} = 1$ $(\forall i \neq j)$, if there exists a spoke arc (i, j), 0 otherwise; $b_{ij} = 1$ $(\forall i < j)$, if a hub edge $\{i, j\}$ is established between two hub nodes i and j, 0 otherwise; $z_{ij} = 1$ $(\forall i < j)$, if there exists a runaway connection $\{i, j\}$ between i and j, 0 otherwise; $g_i = 1$, if i is runaway node, 0 otherwise; w_{ijkl} represents the fraction of flow from i to j routed *via* hub edge (k, l); s_{ijkl} stands for the fraction of flow from i to j routed *via* spoke edge (k, l) and v_{ijkl}, represents the fraction of flow from i to j routed *via* runaway connection (k, l).

Additionally, for any subset $S \subset V$, $\delta^+(S) = \{a = (i, j) | i \in S, j \in V/S\}$, $\delta^-(S) = \{a = (j, i) | i \in S, j \in V/S\}$, $y(\delta(S)^+) = \sum_{i \in S, j \in V/S} y_{ij}$.

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

$$\begin{aligned}
\min \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{\substack{k=1 \\ k \neq j}}^{n} \sum_{\substack{l=1 \\ l \neq k \\ l \neq i}}^{n} & w_{ij} c_{kl} (s_{ijkl} + \lambda v_{ijkl} + \alpha w_{ijkl}) \\
& + \sum_{k=1}^{n} \sum_{\substack{l=1 \\ l > k}}^{n} I_{kl} b_{kl} + \sum_{k=1}^{n} \sum_{\substack{l=1 \\ l \neq k}}^{n} S_{kl} y_{kl} + \sum_{k=1}^{n} \sum_{\substack{l=1 \\ l > k}}^{n} T_{kl} z_{kl} \\
& + \sum_{k=1}^{n} F_k h_k + \sum_{k=1}^{n} G_k g_k \qquad (3.1)
\end{aligned}$$

$$\text{s.t.} \sum_{j=1}^{n} h_j = p, \qquad (3.2)$$

$$x_{ij} \leq h_j, \qquad \forall i, j \in V, j \neq i, \qquad (3.3)$$

$$\sum_{j=1, j \neq i}^{n} x_{ij} + h_i = 1, \qquad \forall i \in V, \qquad (3.4)$$

$$g_i + h_i \leq 1, \qquad \forall i \in V, \qquad (3.5)$$

$$\sum_{j > i}^{n} z_{ij} + \sum_{j < i}^{n} z_{ji} = 2 g_i, \qquad \forall i \in V, \qquad (3.6)$$

$$\sum_{i=1}^{n}\sum_{j>i}^{n} z_{ij} \leq p, \tag{3.7}$$

$$z_{ij} \leq g_j, \qquad \forall i \in V, j > i, \tag{3.8}$$

$$g_i + x_{ik} + g_j + x_{jk} \leq 3, \qquad \forall i,j,k \in V, k \neq j, k \neq i, j > i, \tag{3.9}$$

$$y_{ij} + x_{ik} \leq 1 + x_{jk}, \qquad \forall i,j,k \in V, k \neq j, k \neq i, j \neq i, \tag{3.10}$$

$$y_{ij} + h_i \leq 1 + x_{ji}, \qquad \forall i,j,k \in V, k \neq j, j \neq i, \tag{3.11}$$

$$y_{ij} + x_{ij} \leq 1 + h_j, \qquad \forall i,j,k \in V, k \neq i, j \neq i, \tag{3.12}$$

$$y_{ij} + y_{ji} + x_{ki} + x_{kj} \leq 3 - h_i - h_j, \qquad \forall i,j,k \in V, k \neq j, k \neq i, j > i, \tag{3.13}$$

$$\sum_{j=1, j\neq i}^{n} y_{ij} = 1, \qquad \forall i \in V, \tag{3.14}$$

$$\sum_{j=1, j\neq i}^{n} y_{ji} = 1, \qquad \forall i \in V, \tag{3.15}$$

$$b_{ij} \leq h_i, \qquad \forall i,j \in V, j > i, \tag{3.16}$$

$$b_{ij} \leq h_j, \qquad \forall i,j \in V, j > i, \tag{3.17}$$

$$\sum_{l=1, l\neq i}^{n} w_{ijil} + \sum_{l=1, l\neq i}^{n} v_{ijil} + \sum_{l=1, l\neq i}^{n} s_{ijil} = 1, \qquad \forall i,j \in V, j \neq i, \tag{3.18}$$

$$\sum_{l=1, l\neq j}^{n} w_{ijlj} + \sum_{l=1, l\neq j}^{n} v_{ijlj} + \sum_{l=1, l\neq j}^{n} s_{ijlj} = 1, \qquad \forall i,j \in V, j \neq i, \tag{3.19}$$

$$\sum_{\substack{l=1\\ l\neq i,\\ l\neq k}}^{n} w_{ijkl} + \sum_{\substack{l=1,\\ l\neq i,\\ l\neq k}}^{n} v_{ijkl} + \sum_{\substack{l=1,\\ l\neq i,\\ l\neq k}}^{n} s_{ijkl} = \sum_{\substack{l=1,\\ l\neq j,\\ l\neq k}}^{n} w_{ijlk} + \sum_{\substack{l=1,\\ l\neq j,\\ l\neq k}}^{n} v_{ijlk} + \sum_{\substack{l=1,\\ l\neq j,\\ l\neq k}}^{n} s_{ijlk}, \qquad \forall i,j,k \in V, k \neq i, k \neq j, j \neq i, \tag{3.20}$$

$$s_{ijkl} \leq y_{kl}, \qquad \forall i,j,k,l \in V, l \neq k, l \neq i, k \neq j, j \neq i, \tag{3.21}$$

$$v_{ijkl} + v_{ijlk} \leq z_{kl}, \qquad \forall i,j,k,l \in V, l > k, j \neq i, \tag{3.22}$$

$$w_{ijkl} + w_{ijlk} \leq b_{kl}, \qquad \forall i,j,k,l \in V, l > k, j \neq i, \tag{3.23}$$

$$y(\delta^{+}(S)) \geq \sum_{j \in V/S} x_{ij}, \qquad \forall i \in S \subset V, \tag{3.24}$$

$$y(\delta^{-}(S)) \geq \sum_{j \in V/S} x_{ij}, \qquad \forall i \in S \subset V, \tag{3.25}$$

$$h_i, g_i, x_{ij}, y_{ij}, z_{ik}, b_{ik} \in \{0,1\}, \qquad \forall i,j,k,l \in V, j \neq i, k > i, \tag{3.26}$$

$$w_{ijkl}, v_{ijkl}, s_{ijkl} \in (0,1), \qquad \forall i,j,k,l \in V, l \neq k, l \neq i, k \neq j, j \neq i. \tag{3.27}$$

The objective function minimizes the transportation costs and the setup costs for hubs, runaways, hub edges, runaway connections and spoke arcs. Constraints (3.2) ensure that the number of hubs is equal to p. Constraints (3.3) guarantee that a node i can be allocated to node j, only if node j is a hub. Constraints (3.4) ensure that a node i is either a hub node or is allocated to only one hub. Constraints (3.5) assure that a node i cannot be a hub and a runaway node at the same time. Constraints (3.6) ensure that a runaway node must be adjacent to

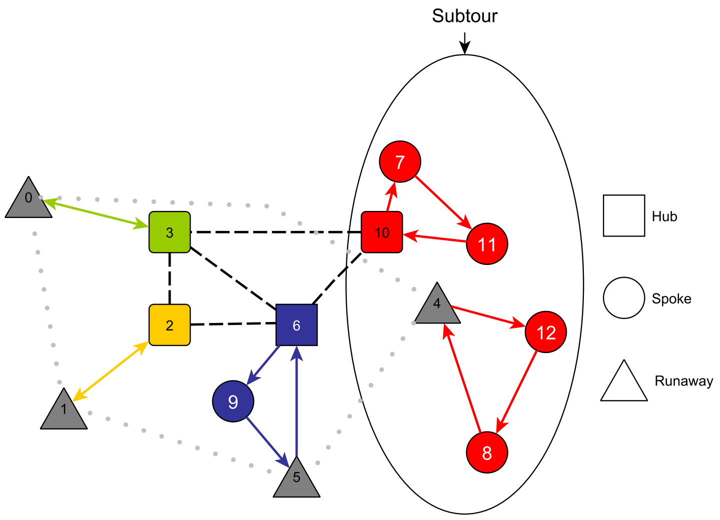


FIGURE 4. Subtour example ($n = 13, p = 4$).

two runaway connections. Constraints (3.7) assure that the number of runaway connections is at most equal to p. Constraints (3.8) ensure that if there is a runaway connection (i, j), then j is a runaway node. Constraints (3.9) ensure that in each cycle there is at most one runaway node. Constraints (3.10)–(3.12) guarantee that a spoke arc links two spokes of the same hub (cycle). Constraints (3.13) are aggregations of the two constraints $y_{ij} + y_{ji} \leq 1$ and $x_{ki} + x_{kj} \leq 2 - h_i - h_j$. The former makes sure that two spoke arcs in opposite directions do not exist and the later guarantees that one spoke node cannot be allocated to two hub nodes at the same time.

Constraints (3.14) ensure that a spoke i has exactly one outgoing spoke arc. Constraints (3.15) ensure that a spoke i has exactly one incoming spoke arc. Constraints (3.16) and (3.17) ensure that if the hub edge b_{ij} exists then the nodes i and j are hubs. Constraints (3.18) assure an origin-destination flow $i - j$ leaves i *via* a spoke, a hub or a runaway connection. Constraints (3.19) assure that an origin-destination flow $i - j$ arrives to its destination j *via* a spoke, a hub or a runaway connection. Constraints (3.20) assure that flow conservation holds at every intermediate node visited along a origin-destination path for flow $i - j$.

Constraints (3.21)–(3.23) guarantee that a flow between two nodes i and j will traverse the link $k - l$ (spoke arc, runaway connection or hub edge) if compatible link exists.

Constraints (3.24) and (3.25) ensure that $\forall i \in S \subset V$ if i is a spoke node allocated to a hub in V/S, there is at least one arc going out of (entering into) S. For any given subset $S \subset V$ and any $i \in S$, in the absence of any arc (i, j) with the tail in S and the head in the complementary set, any path from i to the hub serving i entirely lies within S. Thus, there exists no j outside S, which serves i for any choice of $j \in V/S$. On the contrary, if i is served by a hub in V/S there is a unique path cutting S (at least once) towards the hub serving i.

Constraints (3.24) and (3.25) ensure that situations similar to Figure 4 will not occur.

4. HYPER-HEURISTIC AND VNS APPROACHES FOR pHLPwR

The complexity of our problem and its computational intractability make it impractical to solve realistic size instances. We therefore propose a hyper-heuristic and a VNS approaches aiming to find high quality solutions for larger instances in a reasonable amount of time.

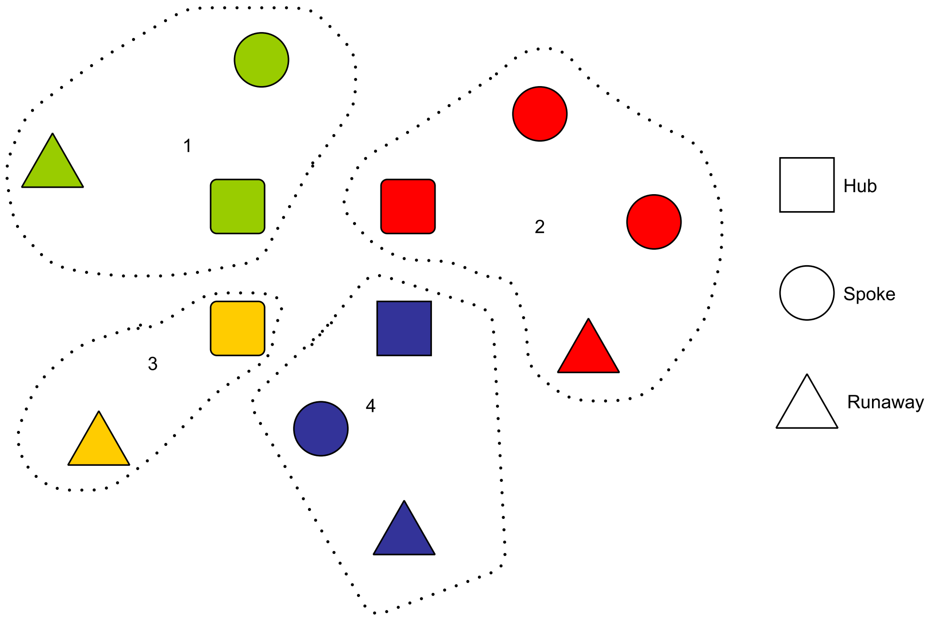


FIGURE 5. pHLPwR without the network structure ($p = 4$ clusters).

4.1. Hyper-heuristic approach

The term hyper-heuristic was first used in 1997 to describe a protocol that combines several artificial intelligence methods in the context of automated theorem proving [17]. It was then employed in combinatorial optimization [13] as *heuristics to choose heuristics*. In this viewpoint, a hyper-heuristic is a high-level approach that solves hard computational search problems, given a particular problem instance and a number of low-level heuristics. It selects and applies the most suitable low-level heuristic at every decisional milestone. The main ingredients of our hyper-heuristic are elaborated in the sequel.

4.1.1. Initial solution

An initial solution, or more importantly a good one, empirically speaking, contributes significantly in the success of many methods (even exact ones) and a hyper-heuristic is of no exception. In the case of this problem, we have carried out an extensive preliminary computational experiments that confirmed again the importance of a good quality initial solution.

A solution to this problem is characterized by (1) location of hub nodes, (2) runaway nodes location, (3) spokes allocation, (4) hub-level, runaway-level structures, and the spoke-level network (*i.e.* the cycles).

One can start with a clustering problem. Figure 5 shows that for $p = 4$, we obtain 4 clusters, and every cluster contains a single runaway (of course if it does not only serving a single isolated node, in which case there will be no runaway) node and one and only one hub node. One of the most popular unsupervised clustering methods is the well-known k-means method [28], due to its ease of implementation and efficiency. Interested readers are referred to some of the related recent contributions including Pérez-Ortega *et al.* [38], Huang *et al.* [24], Mourelo Ferrandez *et al.* [34], Todosijević *et al.* [45], Yahyaei *et al.* [47] and O'Kelly [36].

The main steps of our pHLPwR implemented k-means are described in the Algorithm 1 ($k = p$ the number of hubs). First, we initialize the *centroids* [23]. Then, we construct the clusters. Finally, we locate the hub – as the nearest node to the centroid – and a runaway node for every cluster – chosen to be the farthest node to the centroid. The reason is that one would expect this node to be sufficiently far from the surrounding area of the disrupted hub node.

Algorithm 1: pHLPwR k-means.

```
1 Select the $k$ farthest nodes as initial centroids;
2 Repeat
3 Assign every node to the closest centroid to form $k$ clusters;
4 For each cluster recompute centroids;
5 Until convergence;
6 For each cluster select the nearest node to the centroid as a hub;
7 For each cluster select the farthest node to the centroid as a runaway node;
```

Once Algorithm 1 terminates, we inter-connect all the hubs (complete graph) and construct the cycles for each cluster using a greedy constructive heuristic choosing the cheapest edge to add at every step. Our extensive preliminary computational experiments has revealed that this method is effective and provides quality initial solution for our hyper-heuristic.

4.1.2. Selection method

The selection procedure is the main component of a heuristic-selection hyper-heuristic. A good selection procedure leads to a better decisions when choosing the low-level heuristic to apply and therefore finding good solutions becomes more likely. The selection approach used in this study belongs to the class of reinforcement learning methods (see [4]). A reinforcement learning approach is based on a scoring system. In this work we use an on-line learning approach, each time a low-level heuristic is called, its weight value is updated dynamically; *rewarded*, if the low-level improves the solution or *penalized*, if it worsen the solution.

We define the weight variable as in the following [15]:

$$w_i = \left(\sum_{1}^{N_i} (f^{\text{in}} - f^{\text{out}}) \right) / N_i. \tag{4.1}$$

The weight (w_i) of a low-level heuristic i is equal to the negative sum (minimization case) of the difference in objective function values after applying the low-level heuristic i divided by the total number of times that it has been called during the search where f^{in} is the current objective value, f^{out} is the objective value after calling the low-level heuristic i and N_i is the number of times the low-level heuristic i was used. This weight gives an on-average effectiveness of a low-level heuristic. In other terms, a low-level heuristic with a higher value would probably return better solutions.

The low-level heuristics weights are initialized according to their performances on the initial solution. In fact, after creating the initial solution, we let every heuristic try to improve it until meeting a stopping criteria (the number of non-improving solutions). The equation (4.1) is used to calculate the initial weights.

It must be noted that we have also carried out extensive computational results to see if initializing with random weights would be any better as more exploration should be introduced. However, the conclusion out of numerous observations was that no clear pattern of impact on the performance of method – neither clearly improving nor really deteriorating – could be found. Therefore, we have decided to not include them in this study.

The selection procedure (Algorithm 2) chooses the low-level heuristic i with the highest weight and nc_i less or equal to the parameter R_l (R_l is the maximum authorized consecutive calls) and increments the number of consecutive calls of the low-level heuristic, i, *i.e.* nc_i (see line 11)). If $R_l = 1$, then the selection procedure will select a sequence of low-level heuristics from the best to the worst.

4.1.3. Low-level heuristics

In the following we define the neighborhood structures and the set of low-level heuristics used in our hyper-heuristic. We will also briefly explain how each of them performs:

Algorithm 2: Select-heuristic.

```
Input: A finite sets: $W = \{w_1, w_2, \dots, w_n\}$ of weights, $Nc = \{nc_1, nc_2, \dots, nc_n\}$ the number of consecutive calls of
       each low-level heuristic and $R_l$ the restriction on the number of consecutive calls.
Output: $j = \arg\max_{w_j}$
1  $max \leftarrow -\infty$
2  $j \leftarrow -1$
3  for $i \leftarrow 1$ to $n$ do
4      if $w_i > max$ and $nc_i \leq R_l$ then
5          if $j > -1$ then
6              $nc_j \leftarrow 0$
7          $max \leftarrow w_i$
8          $j \leftarrow i$
9      else
10         $nc_i \leftarrow 0$
11 $nc_j \leftarrow nc_j + 1$
12 return $j$
```

Network low-level heuristics. These heuristics tend to improve the solution without altering the membership of nodes to the clusters.
LINK-HUB: this low-level heuristic adds or deletes a hub edge in the hub-level network for each pair of hubs. While adding a hub edge between two hub nodes, if not already available, does not harm the feasibility, a deletion can cause infeasibility and disconnectedness in the hub-level network in which case the solution is discarded.
LINK-RUNAWAY: add and/or delete a runaway connection to get a new runaway-level network. The number of runaway connections incident to each runaway node must not exceed 2.
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) of the nodes in the cluster and complete the tour.

Distribution low-level heuristics. These local searches try to improve every cycle without changing the number of nodes allocated to it (swapping between nodes).
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 every spoke node, we swap this node with its corresponding runaway node such that the spoke becomes runaway and the runaway becomes a spoke.
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 find the best cluster composition (moving nodes among clusters).
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 finds for a given spoke node i allocated to a hub node k, the nearest spoke node j in the whole network (not only in the nearest cycle to k as seen in the precedent low-level heuristic). After finding the spoke node j with its corresponding hub node l:

– 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.

4.1.4. Hyper-heuristic procedure

While in meta-heuristics the search space is usually the space of solutions and the main goal is to find the best solution, in a hyper-heuristic, the search space is the space of low-level heuristics and therefore the objective is to find, at each step, the best low-level heuristic to apply in order to obtain the best feasible solution. Before explaining our algorithm process we give two definitions to ease understanding.

Move acceptance strategy. The solution obtained at the end of an invocation of a low-level heuristic can be accepted or rejected according to the move acceptance strategy being used. In the case of our algorithm, we use a random walk acceptance criteria, where a solution is accepted regardless of its quality. This strategy adds some noise to the search path, which avoids premature convergence to a local optimum.

Perturbation heuristic. The role of this heuristic is to prevent getting stuck in a local optima by giving distant solutions. In other terms, the objective is to push towards more exploration to search possibly unexplored areas in the search space rather than immediately improving the current solution.

– DESTROY : it partially destroys a solution. For each hub with more than n/p allocated spoke nodes, the cycle is disconnected by removing all the corresponding spoke edges.
– REALLOCATE : every disconnected spoke is reallocated to the nearest hub among the current hub nodes.
– CONSTRUCT : for each hub and its corresponding spoke nodes, spoke edges are added step by step using the nearest neighbor heuristic in order to reconstruct a new cycle.

In order to increase the likelihood of escaping from the local optima and finding a promising solution search space area, the perturbation heuristic accepts as input, the best-found solution rather than the latest visited node.

Our hyper-heuristic starts with the calculation of the initial weights according to the definition seen in Section 4.1.2. The low-level heuristic with the highest weight is designated. Subsequently, the best solution found so far and its objective value are recorded in x' and min, respectively. If x' is the best solution found during the process, it will be stored in x^* and its objective function value in *top*. At every call to a low-level heuristic i, its weight w_i is updated according to (4.1). The perturbation heuristic is called d times before reaching the stopping criteria and takes, x^*, as an entry. We set our termination criteria as the number of consecutive non-improving moves, N_{limit}, or the time limit, T_{limit}. A tabu list T_l is also added. The tabu list is a short-term memory that holds the recently visited solutions to prevent the same solutions to be reconsidered in the near future and a possible cycling in the same search space area. The tabu tenure that provides good results often grows with the size of the problem, that is why we have chosen n as the tabu tenure. In other terms, a solution remains tabu for n iterations then becomes non-tabu.

One can summarize the algorithm process in seven main steps (see Algorithm 3). (1) the low-level heuristics weights are initialized (line 2), (2) After each N_{limit}/d non-improving moves (line 7) the perturbation heuristic is called on the best-found solution (line 8), otherwise the low-level heuristic with the best performance so far, is applied on the current solution (line 10), (3) the neighborhood is explored for the best solution, which is not in the tabu list (line 16). The solution is then considered as current incumbent solution x', even if it is a

Table 2. The parameters used in Algorithm 3.

Parameter	Definition
d	The number of perturbation heuristic calls to escape from a local optimum
T_l	The tabu list
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

non-improving move (line 18), (4) the incumbent solution found x' is added to the tabu list T_l and if the size of T_l exceeds n, the oldest solution is removed in a first-in, first-out (FIFO) scheme (line 20). (5) the weight of the low-level heuristic is updated (line 21), (6) if the current solution improves over the best-found solution, we update the best-found solution, x^*, (7) steps 2–7 are iterated until the termination criteria is met (line 6).

The parameters of our algorithm are presented in Table 2.

4.2. Variable neighborhood search approach

Our hyper-heuristic is a rather comprehensive and complete framework and our goal is not to present a second metaheuristic approach as a state-of-the-art method for this problem. Our aim is to show how a combination of components of our hyper-heuristic would perform in the absence of hyper-heuristic settings. Therefore, a variable neighborhood search (VNS) [31] scheme perhaps makes this comparison more reasonable and can be referred to as a comparable counterpart to our hyper-heuristic.

Let $\mathcal{N} = \{N_1, ..., N_{k_{max}}\}$ be a set of neighborhood structures. The Algorithm 4 iterates over four main steps until the stopping criteria is met (line 6): (1) calls the *Shaking* heuristic with neighborhood structure N_i (line 8), (2) performs the *LocalSearch* procedure (line 9), (3) updates the current solution in x^* (line 12) if an improvement is observed (line 11), (4) examines the subsequent neighborhood (N_{i+1}) if no improvements reported (line 17).

The aim of the *Shaking* procedure is to change and explore other interesting search space areas in the neighborhood N_k and to avoid the local optimum. On the other hand, the role of the *LocalSearch* procedure is exploring and discovering the descent direction within neighborhood N_k to reach a local optimum. Figure 6 summarizes the VNS scheme.

For a fair comparison between the two algorithms, we have used the low-level heuristics of the hyper-heuristic to construct the *Shaking* neighborhoods and the *LocalSearch* procedure. We considered the perturbation operators as the *Shaking* neighborhoods, and the other low-level heuristics as the *LocalSearch* procedure (see Sect. 4.1.3). In doing so, every solution obtained with the hyper-heuristic can be obtained by the VNS algorithm as well. Our implemented version of VNS has the same stopping criteria as the hyper-heuristic (time limit T_{limit} or non-improving moves N_{limit}).

Interested readers on the VNS applications in HLPs are referred to the recent works in the literature Dai *et al.* [14], Serper and Alumur Alev [43], Rahmaniani *et al.* [39], Jarboui *et al.* [25] and Pérez *et al.* [37].

4.3. Cutting planes

As the proposed model is not a compact one, valid inequalities (3.24) and (3.25) need to be separated on the fly and upon violation.

Algorithm 3: Hyper-heuristic.

```
   Input: Data instance, $R_l$, $d$, $T_{\text{limit}}$ and $N_{\text{limit}}$
   Output: The best solution found
1  $x' \leftarrow Initial\_solution()$;
2  $Weight\_initialization(x')$;
3  $x^* \leftarrow x'$
4  $top \leftarrow Eval(x^*)$
5  $nbr\_iter \leftarrow 0$
6  while $nbr\_iter \leq N_{\text{limit}}$ and $time < T_{\text{limit}}$ do
7      if $((nbr\_iter\ mod(N_{\text{limit}}/d)) = 0)$ then
8          $sols \leftarrow Perturbation(x^*)$
9      else
10         $sols \leftarrow select\_heuristic(x')$
11     $x' \leftarrow sols(0)$ // the first solution of the array of solutions sols
12     $min \leftarrow Eval(x')$
13     $i \leftarrow 1$
14     while $i \leq nbr\_sols$ do
15         $cost \leftarrow Eval(sols(i))$
16         if $min > cost$ and $Not\_tabu(T_l, sols(i))$ then
17             $min \leftarrow cost$
18             $x' \leftarrow sols(i)$
19         $i++$
20     $Update\_tabu(T_l, x')$
21     $Update\_weights()$;
22     if $min < top$ then
23         $x^* \leftarrow x'$
24         $top \leftarrow min$
25         $nbr\_iter \leftarrow 0$
26
27     $nbr\_iter++$
28 return $x^*$
```

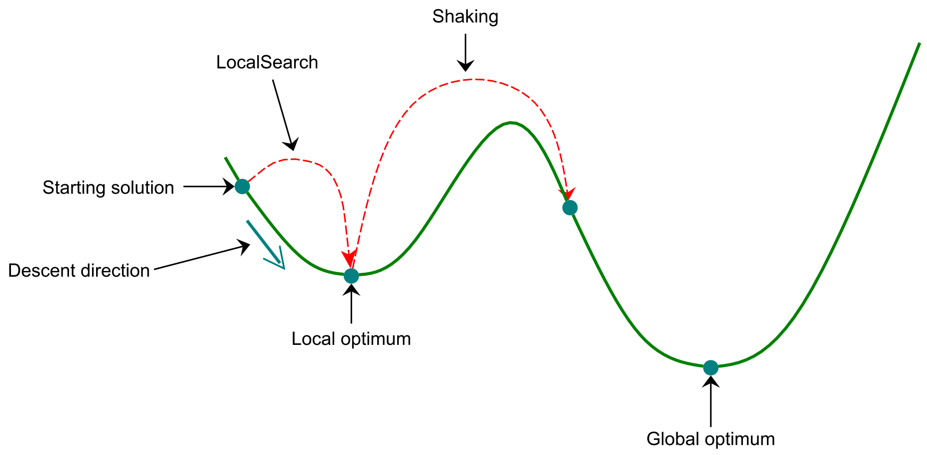


FIGURE 6. VNS basic scheme.

Algorithm 4: Implemented VNS.

```
Input: Data, a set of neighborhood structures $\mathcal{N} = \{N_1, N_2, \ldots, N_{kmax}\}$, $N_{\text{limit}}$ and $T_{\text{limit}}$
Output: The best solution found
1  $x' \leftarrow Initial\_solution()$;
2  $top \leftarrow Eval(x')$
3  $x^* \leftarrow x'$
4  $i \leftarrow 0$
5  $nbr\_iter \leftarrow 1$
6  while $nbr\_iter \leq N_{\text{limit}}$ and $time < T_{\text{limit}}$ do
7      $i \leftarrow i\ mod(i_{max})$ ($i_{max}$ is the number of neighborhood structures)
8      $x' \leftarrow Shaking(x', N_i)$
9      $x' \leftarrow LocalSearch(x')$
10     $cost \leftarrow Eval(x')$
11     if $cost < top$ then
12         $x^* \leftarrow x'$
13         $top \leftarrow cost$
14         $nbr\_ite \leftarrow 0$
15         $i - -$
16     $nbr\_ite \leftarrow nbr\_ite + 1$
17     $i + +$
18 return $x^*$
```

Separation of valid inequalities (3.24) **and** (3.25)**.** Let s be a dummy node, V be the set of nodes visited and $G'(V \cup \{s\}, A)$ a directed graph. For each i establish an arc from s to every j with capacity $\bar{x}_{ij}$, if $\bar{x}_{ij} > 0$. Then, add an arc (i, j) with capacity $\bar{y}_{ij}$ for $\bar{y}_{ij} > 0$. Now, if we push one unit of flow from s destined to a node i, then i must receive a volume of flow equivalent to $\sum_j \bar{x}_{ij}$. A set $S \subset V$ where $i \in S, s \notin S$ defines a cut of $\delta^+(S)(\delta^-(S))$ if the cut capacity is less than $\sum_j \bar{x}_{ij}$. S will deliver valid inequalities of (3.24) (3.25).

5. Computational results

Due to the confidentiality, we are only able to report our results on the instances generated based on the well-known 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: $F_k = \left(\sum_{i=1}^n \sum_{j=1, j\neq i}^n w_{ij} / \max_{i\neq k} d_{ik}\right) \times 10\text{e}^8$, 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: $G_k = \frac{1}{2}\left(\sum_{i=1}^n \sum_{j=1, j\neq i}^n w_{ij} / \max_{i\neq k} d_{ik}\right) \times 10\text{e}^8$. G_k is calculated in the same way as F_k but slightly cheaper than hub nodes.
– Hub edges: $I_{kl} = \left(\frac{w_{kl}/d_{kl}}{\max_{i,j\neq i} w_{ij}/d_{ij}}\right) \times 10\text{e}^7$. 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.
– runaway edge: $T_{kl} = \frac{1}{2}\left(\frac{w_{kl}/d_{kl}}{\max_{i,j\neq i} w_{ij}/d_{ij}}\right) \times 10\text{e}^7$. T_{kl} is calculated with respect to I_{kl}(a runaway edge is less expensive than a hub edge).
– Spoke edge: $S_{kl} = \frac{1}{4}\left(\frac{w_{kl}/d_{kl}}{\max_{i,j\neq i} w_{ij}/d_{ij}}\right) \times 10\text{e}^7$. S_{kl} is calculated with respect to I_{kl} (a spoke edge is less expensive than a hub and runaway edge).

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_zz* where *ww* is the number of nodes, *xx* is the number of hubs, *yy* is the hub-level discount factor α and *zz* is the runaway connection discount factor λ ($\alpha \leq \lambda \leq 1$). Three combinations have been taken into account, namely, $(\alpha, \lambda) \in \{(0.6, 0.8), (0.8.0.9), (0.9, 1)\}$ to avoid too many similarities and have a better approximation of real-life cases. $\lambda = 1$ does not distinguish between the spoke connection and a runaway connection.

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-HLPwR refers to the exact branch-and-cut method (solving with CPLEX and separating from among the exponential number of constraints), H-HLPwR stands for the hyper-heuristic method and VNS-HLPwR refers to the prescribed VNS method.

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 and the exponential constraints are separated *via* a `IloUserCutCallback` (see Sect. 4.3). The time limit was set to 86 400 s (*i.e.* 1 day) and the tolerance ϵ is equal to 1e-6. Constraints (3.24) and (3.25) were separated at every *integer* node (feasible solution).

The results are reported in Table 3 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 and "Ncuts" represents the number of separated sub-tour elimination cuts.

Generally speaking, the objective value increases with the size of the instance and for the same instance size, n, the smaller p construct an upper envelop on the objective function of instances with a higher p (see Fig. 7b).

Figure 7a shows the relation between α and λ with the computational difficulties of the problem. For the same instance size, the smaller α and λ are, the higher is the computational time needed to prove optimality. Indeed, the smallest execution times recorded are for $\alpha = 0.9$, $\lambda = 1$ (see Tab. 3) where runaway connections offers no advantage over the spoke edges. As we only had provably optimal solutions for up to size $n = 15$, Figure 7a depicts only up to this size.

Most of the instances with less than 20 nodes were solved to optimality in less than 24 h. However, for 15–25 nodes, CPLEX often finds a feasible solution but needs more time to prove optimality. CPLEX encounters serious difficulties when dealing with instances of more than 20 nodes – even delivering a feasible solution remains a challenge.

We observe that the number of cuts added correlates to the number of hub nodes p. The lower the p is, compared to the number of nodes n, the higher is the number of cuts to be separated. This sounds reasonable since the smaller p/n is, the more nodes will sit on every cycle and as such, we may need more sub-tours to be eliminated. For $n = 15$, $p = 7$, no cuts are added because every cycle has at most three nodes and as such, we may not obtain sub-tours.

It must be noted that the number of sub-tour elimination separated remains reasonable.

5.2. Hyper-heuristic vs VNS

The results of the hyper-heuristic and VNS approaches are depicted in Table 4 for small size instances and Table 5 for medium and large size instances. Again, "Obj. Val." is the objective value of the best solution found. "NSols" represent the number of distinct feasible solutions encountered along the search, "E.T." stands for the execution time, the column "GAP" in Table 4 reports the relative GAP between the best-known solution of the hyper-heuristic (respectively best-known solution of VNS) and the best solution found by CPLEX. In Table 5

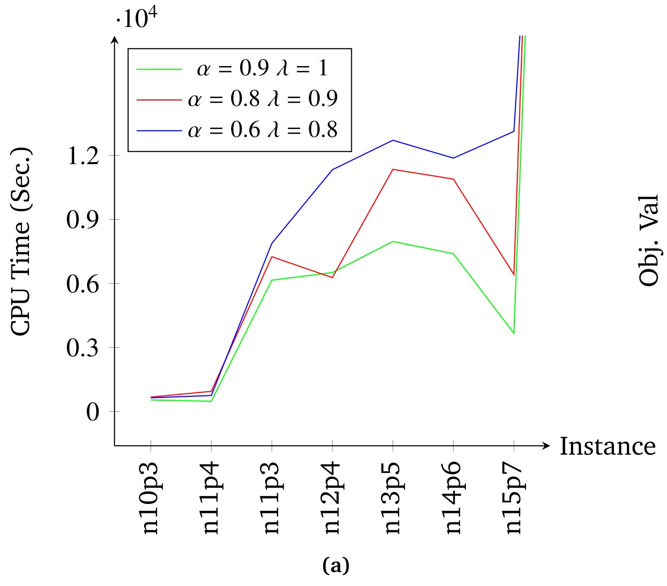
TABLE 3. Computational results of MIP-HLPwR.

Instance	Obj. Val.	NSols.	E.T. (s)	GAP (%)	Nnodes	Status	Best Obj.	NCuts
n10p3_0.6_0.8	77 593 660.780	10	659.868	0.000	3558	Optimal	77 593 660.780	66
n10p3_0.8_0.9	82 365 794.641	7	725.919	0.000	3826	Optimal	82 365 794.641	83
n10p3_0.9_1	85 258 929.966	14	557.875	0.000	3560	Optimal	85 258 929.966	87
n11p3_0.6_0.8	20 358 494.269	24	8293.729	0.000	16 369	Optimal	20 358 494.269	125
n11p3_0.8_0.9	21 375 615.575	11	6526.427	0.000	13 438	Optimal	21 375 615.575	98
n11p3_0.9_1	22 051 911.670	18	6646.141	0.000	15 101	Optimal	22 051 911.670	94
n11p4_0.6_0.8	17 597 082.029	10	772.829	0.000	2633	Optimal	17 597 082.029	34
n11p4_0.8_0.9	19 128 595.920	14	1049.542	0.000	3929	Optimal	19 128 595.920	29
n11p4_0.9_1	19 835 503.663	10	562.119	0.000	2391	Optimal	19 835 503.663	26
n12p4_0.6_0.8	20 492 068.237	14	10 546.571	0.000	18 470	Optimal	20 492 068.237	87
n12p4_0.8_0.9	21 780 501.640	10	6276.293	0.000	11 316	Optimal	21 780 501.640	55
n12p4_0.9_1	22 833 868.689	10	7252.798	0.000	13 666	Optimal	22 833 868.689	61
n13p5_0.6_0.8	22 572 799.697	15	13 185.484	0.000	16 054	Optimal	22 572 799.697	53
n13p5_0.8_0.9	24 523 121.813	12	10 762.899	0.000	14 012	Optimal	24 523 121.813	42
n13p5_0.9_1	25 878 687.968	8	9460.868	0.000	14 012	Optimal	25 878 687.968	64
n14p5_0.6_0.8	39 620 089.247	16	86 407.689	3.151	33 465	Feasible	38 371 289.728	79
n14p5_0.8_0.9	43 344 082.949	17	86 410.295	3.401	32 961	Feasible	41 869 544.721	108
n14p5_0.9_1	45 209 210.686	11	79 831.093	0.000	47 822	Optimal	45 209 210.686	76
n14p6_0.6_0.8	36 633 192.377	7	10 972.735	0.000	8570	Optimal	36 633 192.377	20
n14p6_0.8_0.9	40 507 279.264	11	10 087.959	0.000	8771	Optimal	40 507 279.264	24
n14p6_0.9_1	42 934 170.429	9	6991.121	0.000	7914	Optimal	42 934 170.429	24
n15p6_0.6_0.8	42 721 023.156	7	58 676.703	0.000	17 714	Optimal	42 721 023.156	47
n15p6_0.8_0.9	48 534 512.566	14	86 405.381	4.848	19 267	Feasible	46 181 503.431	59
n15p6_0.9_1	50 737 550.187	8	78 024.244	0.000	33 688	Optimal	50 737 550.187	44
n15p7_0.6_0.8	40 866 011.253	9	11 010.160	0.000	9771	Optimal	40 866 011.253	0
n15p7_0.8_0.9	45 711 471.247	8	7544.630	0.000	4825	Optimal	45 711 471.247	0
n15p7_0.9_1	48 506 492.760	10	4706.846	0.000	3577	Optimal	48 506 492.760	0
n20p6_0.6_0.8	84 052 376.665	2	86 408.736	40.104	377	Feasible	50 343 311.492	92
n20p6_0.8_0.9	91 036 839.818	6	86 410.608	32.523	318	Feasible	61 428 873.974	50
n20p6_0.9_1	94 125 937.923	6	86 411.965	26.786	642	Feasible	68 912 769.711	66
n25p7_0.6_0.8	–	0	86 400.794	∞	52	AbortTimeLim	47 856 545.739	0
n25p7_0.8_0.9	137 309 250.562	1	86 404.304	56.389	45	Feasible	59 881 449.529	14
n25p7_0.9_1	–	0	86 401.029	∞	57	AbortTimeLim	65 785 424.857	5
n30p8_0.6_0.8	–	0	86 402.557	∞	27	AbortTimeLim	47 775 456.176	0
n30p8_0.8_0.9	–	0	86 402.089	∞	23	AbortTimeLim	60 085 139.739	0
n30p8_0.9_1	–	0	86 402.542	∞	24	AbortTimeLim	66 130 965.810	0
n35p8_0.6_0.8	–	0	86 405.147	∞	8	AbortTimeLim	46 776 775.958	0
n35p8_0.8_0.9	–	0	86 417.112	∞	7	AbortTimeLim	58 880 661.267	0
n35p8_0.9_1	–	–	–	–	–	Failed	–	–
n40p8_0.6_0.8	–	–	–	–	–	Failed	–	–
n40p8_0.8_0.9	–	–	–	–	–	Failed	–	–
n40p8_0.9_1	–	–	–	–	–	Failed	–	–

the "GAP" column is calculated using the best-known solution of the hyper-heuristic and best-known solution of VNS (a negative GAP means that VNS gave a better solution than the hyper-heuristic).

The time limit T_{limit} was set to 600 s (10 min). For the hyper-heuristic, the parameter R_l is set to 3 (number of consecutive call limit) and d is set to 2 (*i.e.* the perturbation heuristic is called two times before reaching the stopping criterion).

The main difference between the two approaches is that in VNS, the change of neighborhood structure does not rely on any learning mechanism and each time a neighborhood is no longer capable of improving a solution, changing neighborhood i to the next neighborhood $i + 1$ is done without taking into account any information



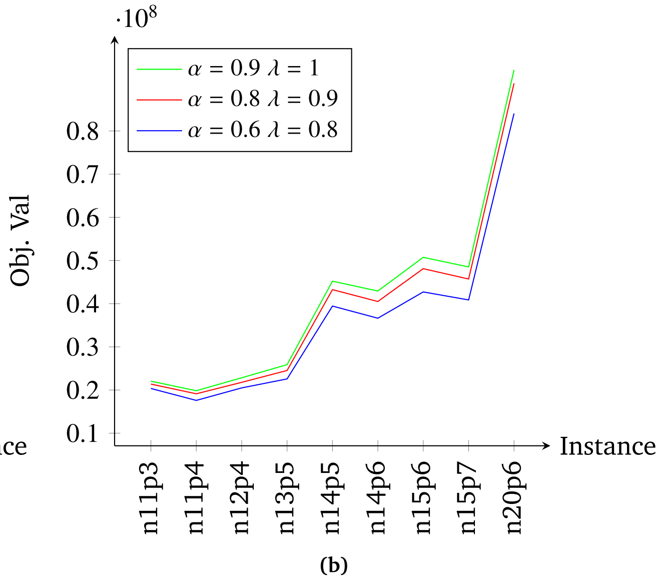


FIGURE 7. Numerical results of the MIP. (a) MIP time. (b) Objetive value comparison.

gathering or memory usage. On the other hand, the hyper-heuristic uses a learning mechanism to allow the overall process of algorithm to evolve with each iteration and to choose each time the best low-level heuristic to apply.

One can conclude from Table 4 that the solutions obtained by the hyper-heuristic are in general better than those of VNS, since VNS reaches the optimal solution only once (*i.e.* for n10p3_0.8_0.9) and the maximum gap recorded among all instances is equal to 5.877%, while the hyper-heuristic fails only once in finding the optimal solution (see n13p5_0.8_0.9) and there the gap is equal to 0.003%. The hyper-heuristic approach always finds a solutions better than the feasible solutions returned by CPLEX and in such a case the minimal gap recorded is equal to -41.261%.

The Table 6 elaborates on the solution structures obtained by the hyper-heuristic for medium and large size instances where "S.N." and "H.N." are the smallest and highest number of nodes per cycle, respectively. The ratio between the smallest and highest number of nodes and the average nodes per cycle (n/p) are given in "(S.N. $\times$ p)/n" and "(H.N. $\times$ p)/n", respectively (if (S.N. $\times$ p)/n = (H.N. $\times$ p)/n, then the number of nodes associated with each hub is similar). The column "Hub Nodes" depicts the hub nodes obtained for each instance. The hub level network structure given in column "H. Net" where "C" stands for a complete graph and "NC" for incomplete graph.

Generally speaking in our obtained solutions structures (Tab. 6), there is always a hub with a high number of spokes allocated to and a hub with a low number of spokes allocated to (no solution had the same number of spokes for each hub). We observed that the hub-level network is rarely a complete graph and that the hub nodes change with α and λ, which shows the impact of α and λ on the network structure.

Figure 8a shows that the computational time increases as instance size grows. On the other hand, the lower α and λ are, the less is the computational time needed for large size instances.

The objective value for each instance of the same size should decrease with the decreasing α and λ but we notice that for large instances this is not the case (see Fig. 8b). Knowing that the hyper-heuristic is deterministic and as such the algorithm follows the same path at each execution, one may conclude that if we find a value with a certain α and λ, we might find a better value with smaller α and λ but it is not really the case. As we

TABLE 4. Results of the hyper-heuristic and the VNS algorithms for small size instances.

Instance	H-HLPwR				VNS-HLPwR			
	Obj. Val.	NSols.	E.T. (s)	Hyper cplex GAP(%)	Obj. Val.	NSols.	E.T. (s)	VNS CPLEX GAP (%)
n10p3_0.6_0.8	77 593 660.780	195 474	25.483	0.000	78 951 975.770	228 597	8.596	1.750
n10p3_0.8_0.9	82 365 794.641	133 999	14.252	0.000	82 365 794.641	244 174	9.298	0.000
n10p3_0.9_1	85 258 929.966	138 029	20.347	0.000	85 496 741.889	242 701	9.220	0.278
n11p3_0.6_0.8	20 358 494.269	244 826	37.637	0.000	20 449 738.680	273 972	12.202	0.448
n11p3_0.8_0.9	21 375 615.575	285 467	43.101	0.000	21 605 841.250	283 294	12.507	1.077
n11p3_0.9_1	22 051 911.670	181 412	25.000	0.000	22 057 943.393	316 949	14.218	0.027
n11p4_0.6_0.8	17 597 082.029	139 594	20.934	0.000	17 607 279.345	291 906	12.814	0.057
n11p4_0.8_0.9	19 128 595.920	121 142	19.751	0.000	20 044 347.146	291 449	12.869	4.787
n11p4_0.9_1	19 835 503.663	168 888	24.797	0.000	20 133 884.670	309 126	13.679	1.504
n12p4_0.6_0.8	20 492 068.237	219 046	30.155	0.000	20 511 547.066	343 459	17.670	0.095
n12p4_0.8_0.9	21 780 501.640	254 423	36.231	0.000	21 875 191.009	349 674	18.175	0.434
n12p4_0.9_1	22 833 868.689	151 099	23.888	0.000	23 059 461.255	352 199	17.891	0.987
n13p5_0.6_0.8	22 572 799.697	321 806	103.566	0.000	22 573 390.712	426 627	25.768	0.002
n13p5_0.8_0.9	24 524 102.513	162 854	27.975	0.003	25 063 539.424	452 671	26.931	2.203
n13p5_0.9_1	25 878 687.968	264 994	44.231	0.000	26 122 787.882	413 235	24.779	0.943
n14p5_0.6_0.8	39 613 147.399	373 042	26.972	–0.017	40 151 760.535	252 445	17.786	1.341
n14p5_0.8_0.9	43 252 132.757	330 950	45.807	–0.212	45 891 712.452	241 370	17.015	5.877
n14p5_0.9_1	45 209 210.686	802 421	121.697	0.000	45 724 316.173	478 423	33.085	1.139
n14p6_0.6_0.8	36 633 192.842	153 804	28.315	0.000	36 924 259.216	347 141	24.675	0.794
n14p6_0.8_0.9	40 507 279.264	196 164	33.912	0.000	40 714 563.974	311 890	21.565	0.511
n14p6_0.9_1	42 934 170.429	155 273	25.361	0.000	43 386 641.062	322 208	22.180	1.053
n15p6_0.6_0.8	42 721 023.156	353 664	29.042	0.000	44 480 326.779	354 487	28.907	4.118
n15p6_0.8_0.9	48 129 799.470	239 356	26.547	–0.833	48 537 186.317	350 375	28.715	0.005
n15p6_0.9_1	50 737 550.187	221 696	25.756	0.000	50 826 302.243	335 377	27.480	0.174
n15p7_0.6_0.8	40 866 011.253	355 698	28.966	0.000	42 345 232.627	419 349	33.688	3.619
n15p7_0.8_0.9	45 711 471.247	103 437	17.140	0.000	47 162 507.343	417 507	34.280	3.174
n15p7_0.9_1	48 506 492.760	274 968	29.948	0.000	49 589 687.830	432 517	34.716	2.233
n20p6_0.6_0.8	73 507 081.315	259 064	41.472	–12.546	82 417 191.184	103 146	15.603	–1.945
n20p6_0.8_0.9	81 841 623.929	112 504	18.744	–10.100	83 359 922.804	124 391	19.141	–8.432
n20p6_0.9_1	85 614 920.755	304 312	47.851	–9.042	88 920 174.914	104 588	16.196	–5.530
n25p7_0.8_0.9	80 654058.676	133 604	38.692	–41.261	83 149 139.101	97 433	26.715	–39.443

see with the case $\alpha = 0.8$ and $\lambda = 0.9$, the algorithm has more difficulties to find a good solution and got stuck in a local optimum.

Figure 9a shows that the larger the instance size is, the more time consuming is the hyper-heuristic compared to VNS before it terminates, which is an indication that the hyper-heuristic does not get stuck easily in local optimum and so the mechanisms added to the hyper-heuristic to avoid the local optimum works quite well.

In terms of number of feasible solutions found, VNS is better than hyper-heuristic since in 66% of the cases it finds more solutions due to the VNS intensification heuristic, which goes deeper on every search space area (see Fig. 9b). However in terms of solution quality, which is an important aspect of the comparison, Figure 9c depicts that the hyper-heuristic finds better solutions than VNS and is more reliable and stable. In fact, the hyper-heuristic delivers good solutions on most of the instances while having found less solutions than VNS.

Table 5 gives an overview of the results of the hyper-heuristic and VNS for large solution space, where finding a promising area containing eventually the optimal solution becomes difficult. We observe that the highest gaps are recorded for large size instances with large solution space. This gaps are due to the fact that the hyper-heuristic chooses the promising area to explore based on knowledge obtained from the low-level heuristic quality. In other terms, the hyper-heuristic is smarter than VNS, which is based on no learning mechanism.

Table 5. Results of the hyper-heuristic and the VNS algorithms for medium and large size instances.

	H-HLPwR			VNS-HLPwR			
Instance	Obj. Val.	NSols.	E.T. (s)	Obj. Val.	NSols.	E.T. (s)	Hyper VNS GAP (%)
n40p4_0.6_0.8	148 138 786.090	49 401	49.286	149 117 126.105	67 801	68.441	0.660
n40p4_0.8_0.9	161 114 461.028	58 541	57.061	171 562 517.275	43 309	44.606	6.484
n40p4_0.9_1	161 968 384.722	69 170	68.254	167 445 833.970	75 164	74.924	3.381
n40p5_0.6_0.8	113 005 244.011	115 219	111.825	128 879 209.738	64 899	66.036	14.047
n40p5_0.8_0.9	131 221 540.285	121 454	115.098	139 019 009.543	73 150	73.674	5.942
n40p5_0.9_1	154 162 076.939	52 450	60.923	160 013 968.489	68 049	62.954	3.795
n40p6_0.6_0.8	103 984 445.129	55 506	54.058	106 714 188.198	62 957	63.155	2.625
n40p6_0.8_0.9	104 767 544.042	89 533	87.950	108 889 345.755	77 408	78.165	3.934
n40p6_0.9_1	11 4734 524.101	58 083	57.155	113 796 258.989	80 747	80.090	–0.817
n50p4_0.6_0.8	157 909 765.531	14 552	27.030	169 438 116.289	21 700	38.844	7.300
n50p4_0.8_0.9	159 532 981.095	13 617	25.428	171 028 682.989	19 643	34.615	7.205
n50p4_0.9_1	158 102 762.268	20 520	37.662	174 131 059.797	29 592	52.548	10.137
n50p5_0.6_0.8	134 342 531.901	22 852	42.470	137 296 304.548	26 579	48.386	2.198
n50p5_0.8_0.9	141 056 845.686	32 446	58.329	154 072 647.019	24 924	44.502	9.227
n50p5_0.9_1	143 075 031.912	90 191	156.127	143 168 823.947	106 488	186.436	0.065
n50p6_0.6_0.8	111 256 527.176	74 433	119.888	117 844 078.512	77 151	122.628	5.921
n50p6_0.8_0.9	121 089 305.695	68 729	110.447	130 724 405.366	36 941	64.917	7.957
n50p6_0.9_1	125 815 623.235	83 985	136.744	128 589 484.436	45 055	81.393	2.204
n60p4_0.6_0.8	169 002 097.262	42 159	108.108	173 947 891.806	35 096	98.594	2.926
n60p4_0.8_0.9	175 188 580.124	24 738	65.251	187 725 676.808	56 915	158.611	7.156
n60p4_0.9_1	178 425 530.893	43 249	112.677	187 685 325.418	35 681	98.582	5.189
n60p5_0.6_0.8	154 967 528.427	23 333	73.805	160 569 265.881	55 401	160.104	3.614
n60p5_0.8_0.9	162 381 946.539	73 275	210.158	157 765 154.746	51 213	151.268	–2.843
n60p5_0.9_1	166 500 564.721	43 549	131.203	167 795 253.727	39 734	116.069	0.777
n60p6_0.6_0.8	122 545 418.790	77 638	226.746	145 232 971.584	88 203	223.523	18.513
n60p6_0.8_0.9	146 113 509.668	41 377	128.693	142 914 535.634	46 197	136.241	–2.189
n60p6_0.9_1	147 703 870.562	29 562	92.513	161 694 330.738	41 292	119.069	9.471
n70p4_0.6_0.8	192 428 066.145	28 486	138.032	179 045 038.186	40 338	179.023	–6.954
n70p4_0.8_0.9	206 603 867.306	34 696	172.072	228 474 586.016	36 444	160.256	10.585
n70p4_0.9_1	224 829 414.796	28 259	130.518	274 812 800.508	25 042	108.964	22.231
n70p5_0.6_0.8	149 207 903.263	112 073	438.122	161 857 851.826	151 284	578.071	8.478
n70p5_0.8_0.9	161 817 036.066	79 393	315.656	183 353 913.235	41 039	186.520	13.309
n70p5_0.9_1	187 921 152.051	100 723	414.303	193 965 901.238	69 184	261.476	3.216
n70p6_0.6_0.8	174 772 974.482	35 851	163.751	158 446 540.981	49 174	202.502	–9.341
n70p6_0.8_0.9	171 566 552.198	39 451	197.199	188 350 564.183	46 674	214.103	9.782
n70p6_0.9_1	158 806 582.863	46 254	221.485	167 322 855.988	73 401	278.733	5.362
n80p4_0.6_0.8	202 846 715.417	41 299	283.303	191 264 990.591	59 527	372.048	–5.709
n80p4_0.8_0.9	218 864 328.032	41 299	292.682	240 718 670.313	55 967	354.896	9.985
n80p4_0.9_1	220 342 768.129	71 331	475.963	265 826 532.426	46 223	278.956	20.642
n80p5_0.6_0.8	188 885 325.674	45 660	314.367	170 681 155.371	46 211	285.240	–9.637
n80p5_0.8_0.9	180 485 726.339	65 814	440.213	199 346 133.417	41 350	275.257	10.449
n80p5_0.9_1	170 580 133.611	64 659	429.543	180 860 955.637	98 915	541.556	6.026
n80p6_0.6_0.8	183 101 479.379	26 980	170.270	187 349 115.698	58 557	350.209	2.319

Table 5. continued.

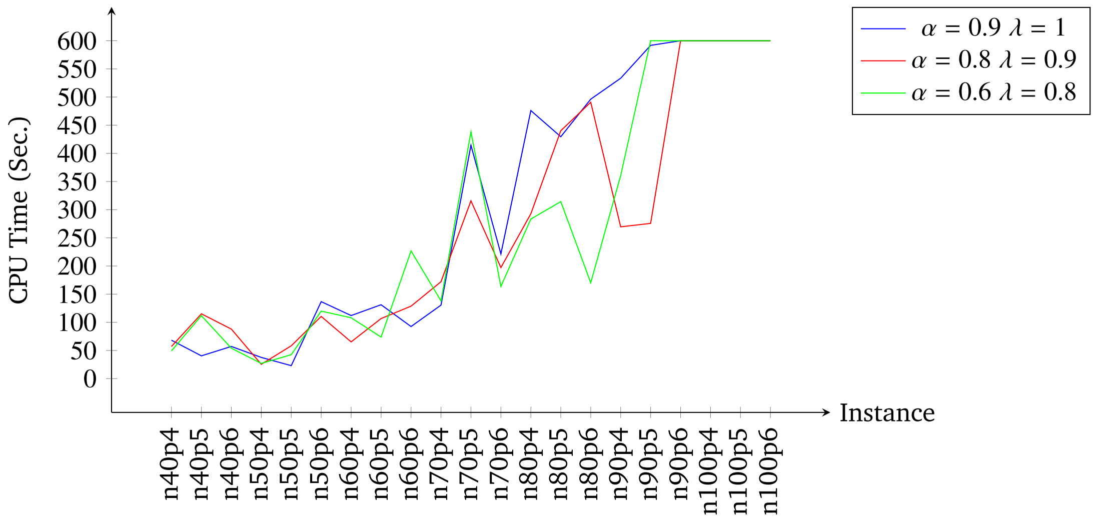
Instance	H-HLPwR			VNS-HLPwR			Hyper VNS GAP (%)
	Obj. Val.	NSols.	E.T. (s)	Obj. Val.	NSols.	E.T. (s)	
n80p6_0.8_0.9	183 912 599.384	75 699	490.697	177 529 525.433	67 853	432.838	–3.470
n80p6_0.9_1	156 996 820.272	75 653	496.245	181 233 695.779	70 842	437.536	15.437
n90p4_0.6_0.8	202 791 891.331	41 883	360.565	217 394 590.144	40 908	331.785	7.200
n90p4_0.8_0.9	224 482 869.606	51 825	T_{limit}	223 106 286.458	57 486	473.120	–0.613
n90p4_0.9_1	229 610 546.982	55 191	533.389	226 589 361.793	51 524	422.694	–1.315
n90p5_0.6_0.8	166 226 023.566	64 500	T_{limit}	207 363 553.578	79 885	T_{limit}	24.747
n90p5_0.8_0.9	184 094 110.772	28 865	275.663	185 995 172.129	73 340	T_{limit}	1.032
n90p5_0.9_1	167 416 557.335	66 038	591.735	207 381 153.700	54 811	454.516	23.871
n90p6_0.6_0.8	165 366 131.295	64 900	T_{limit}	193 071 250.058	78 046	T_{limit}	16.753
n90p6_0.8_0.9	200 491 533.112	64 651	T_{limit}	194 814 834.654	63 863	526.826	–2.831
n90p6_0.9_1	195 582 760.605	66 573	T_{limit}	182 757 300.005	72 799	T_{limit}	–6.557
n100p4_0.6_0.8	214 772 996.844	52 494	T_{limit}	212 171 192.914	53 987	575.176	–1.211
n100p4_0.8_0.9	267 448 010.993	59 359	T_{limit}	256 225 646.356	58 265	T_{limit}	–4.196
n100p4_0.9_1	219 585 842.209	48 640	T_{limit}	255 975 181.789	55 463	588.572	16.571
n100p5_0.6_0.8	182 589 034.595	51 399	T_{limit}	207 742 381.588	46 475	518.437	13.775
n100p5_0.8_0.9	207 744 504.144	54 316	T_{limit}	220 157 176.003	56 684	599.679	5.974
n100p5_0.9_1	215 090 122.749	63 343	T_{limit}	216 977 957.144	52 448	545.058	0.877
n100p6_0.6_0.8	167 022 456.992	52 533	T_{limit}	181 540 345.551	53 437	T_{limit}	8.692
n100p6_0.8_0.9	182 849 599.132	46 595	T_{limit}	185 083 903.866	55 913	T_{limit}	1.221
n100p6_0.9_1	168 247 877.285	51 697	T_{limit}	194 676 676.274	55 197	T_{limit}	15.708

Table 6. Hyper-heuristic structures information for medium and large size instances.

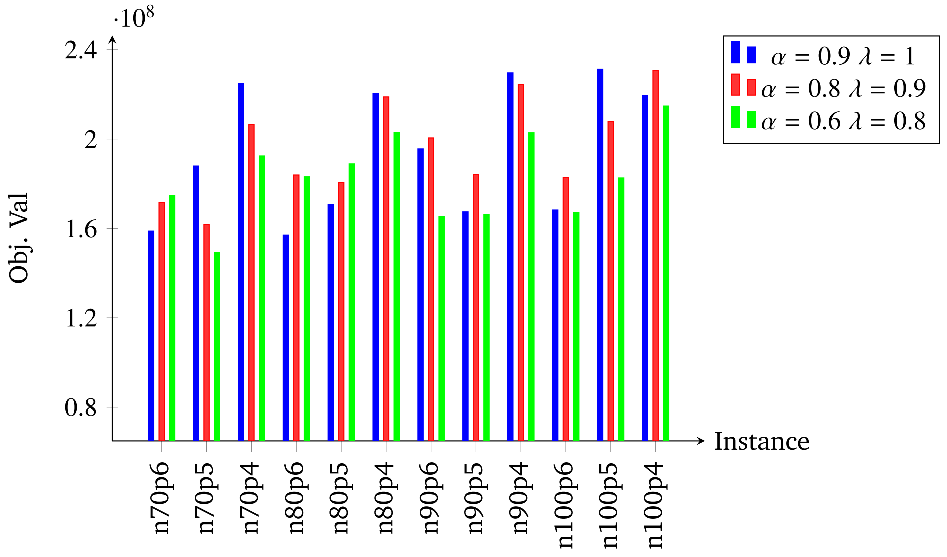
Instance	S.N.	(S.N. $\times$ p)/n	H.N.	(H.N. $\times$ p)/n	Hub Nodes	H. Net.
n40p4_0.6_0.8	6	0.60	11	1.10	9;17;2;25	C
n40p4_0.8_0.9	3	0.30	13	1.30	18;3;11;25	NC
n40p4_0.9_1	3	0.30	17	1.70	7;27;26;25	NC
n40p5_0.6_0.8	6	0.75	10	1.25	4;19;30;28;25	NC
n40p5_0.8_0.9	5	0.62	13	1.62	1;26;17;27;34	NC
n40p5_0.9_1	4	0.50	20	2.50	1;8;17;6;9	NC
n40p6_0.6_0.8	4	0.60	10	1.50	4;19;30;36;11;2	NC
n40p6_0.8_0.9	6	0.90	8	1.20	18;12;36;26;11;19	NC
n40p6_0.9_1	5	0.75	10	1.50	10;28;35;17;11;19	NC
n50p4_0.6_0.8	7	0.56	20	1.60	6;12;13;35	NC
n50p4_0.8_0.9	8	0.64	18	1.44	23;15;13;22	NC
n50p4_0.9_1	8	0.64	16	1.28	14;12;31;22	C
n50p5_0.6_0.8	6	0.60	17	1.70	5;15;13;31;6	NC
n50p5_0.8_0.9	2	0.20	16	1.60	5;23;13;12;6	NC
n50p5_0.9_1	2	0.20	20	2.00	23;31;13;12;42	NC
n50p6_0.6_0.8	3	0.36	14	1.68	32;34;39;33;15;7	NC
n50p6_0.8_0.9	4	0.48	10	1.20	12;45;6;37;36;13	NC
n50p6_0.9_1	5	0.60	13	1.56	3;32;23;31;6;22	NC
n60p4_0.6_0.8	2	0.13	24	1.60	26;42;47;41	NC
n60p4_0.8_0.9	7	0.46	29	1.93	26;42;44;8	NC

Table 6. continued.

Instance	S.N.	(S.N. $\times p)/n$	H.N.	(H.N. $\times p)/n$	Hub Nodes	H. Net.
n60p4_0.9_1	8	0.53	22	1.46	14;40;1;27	NC
n60p5_0.6_0.8	7	0.58	22	1.83	13;41;7;15;24	NC
n60p5_0.8_0.9	3	0.25	27	2.25	13;1;0;14;2	NC
n60p5_0.9_1	5	0.41	24	2.00	12;1;26;3;40	NC
n60p6_0.6_0.8	8	0.80	16	1.60	29;54;6;1;41;18	NC
n60p6_0.8_0.9	2	0.20	18	1.80	14;0;6;1;13;15	NC
n60p6_0.9_1	2	0.20	17	1.70	2;26;14;1;18;15	NC
n70p4_0.6_0.8	5	0.28	30	1.71	38;48;46;21	NC
n70p4_0.8_0.9	8	0.45	35	2.00	55;48;3;31	NC
n70p4_0.9_1	9	0.51	38	2.17	10;48;53;34	C
n70p5_0.6_0.8	9	0.64	25	1.78	8;48;3;21;46	NC
n70p5_0.8_0.9	6	0.42	26	1.85	8;48;18;4;63	NC
n70p5_0.9_1	6	0.42	26	1.85	6;5;20;45;46	NC
n70p6_0.6_0.8	6	0.51	21	1.80	38;48;18;3;46;34	NC
n70p6_0.8_0.9	5	0.42	20	1.71	10;44;2;43;31;19	NC
n70p6_0.9_1	3	0.25	25	2.14	21;15;2;3;6;14	NC
n80p4_0.6_0.8	2	0.10	37	1.85	54;78;23;22	NC
n80p4_0.8_0.9	6	0.30	39	1.95	54;53;8;21	C
n80p4_0.9_1	8	0.40	39	1.95	54;71;8;7	C
n80p5_0.6_0.8	2	0.12	31	1.93	54;3;6;21;71	NC
n80p5_0.8_0.9	13	0.81	20	1.25	49;53;36;35;37	NC
n80p5_0.9_1	13	0.81	22	1.37	54;57;52;33;20	NC
n80p6_0.6_0.8	4	0.30	31	2.32	49;6;63;1;4;33	NC
n80p6_0.8_0.9	8	0.60	22	1.65	49;11;35;2;22;18	NC
n80p6_0.9_1	7	0.52	24	1.80	56;54;35;2;22;52	NC
n90p4_0.6_0.8	15	0.66	37	1.64	23;10;55;25	NC
n90p4_0.8_0.9	11	0.48	40	1.77	4;10;18;19	NC
n90p4_0.9_1	11	0.48	32	1.42	50;10;55;24	NC
n90p5_0.6_0.8	13	0.72	28	1.55	37;21;7;8;13	NC
n90p5_0.8_0.9	12	0.66	24	1.33	55;2;8;25;31	NC
n90p5_0.9_1	11	0.61	28	1.55	55;42;8;39;30	NC
n90p6_0.6_0.8	6	0.40	30	2.00	55;2;18;61;26;17	NC
n90p6_0.8_0.9	3	0.20	33	2.20	55;10;16;23;44;5	NC
n90p6_0.9_1	5	0.33	33	2.20	55;10;8;23;38;4	NC
n100p4_0.6_0.8	11	0.44	41	1.64	73;31;6;69	NC
n100p4_0.8_0.9	6	0.24	54	2.16	66;31;32;69	NC
n100p4_0.9_1	15	0.6	41	1.64	66;3;29;69	NC
n100p5_0.6_0.8	7	0.35	36	1.80	69;27;77;50;9	NC
n100p5_0.8_0.9	6	0.30	38	1.90	69;27;77;50;7	NC
n100p5_0.9_1	6	0.30	37	1.85	69;31;13;47;9	NC
n100p6_0.6_0.8	4	0.24	29	1.74	6;48;9;47;44;7	NC
n100p6_0.8_0.9	4	0.24	33	1.98	10;48;12;46;44;7	NC
n100p6_0.9_1	4	0.24	26	1.56	6;21;9;47;27;68	NC



(a)



(b)

FIGURE 8. Numerical results of the H-HLPwR. (a) Execution time. (b) Objective value.

Our hyper-heuristic algorithm has demonstrated its effectiveness both in solution quality and computational time even for larger size instances when compared to comparable methods.

6. SUMMARY AND CONCLUSION

In this study, we have introduced a new variant of the hub location problem that we referred to as p-Hub Location Problem with Runaway. The runaway nodes introduced in this work are of special interest in service networks based on round-trips where a set of nodes being served by one single hub node whose failure can disrupt

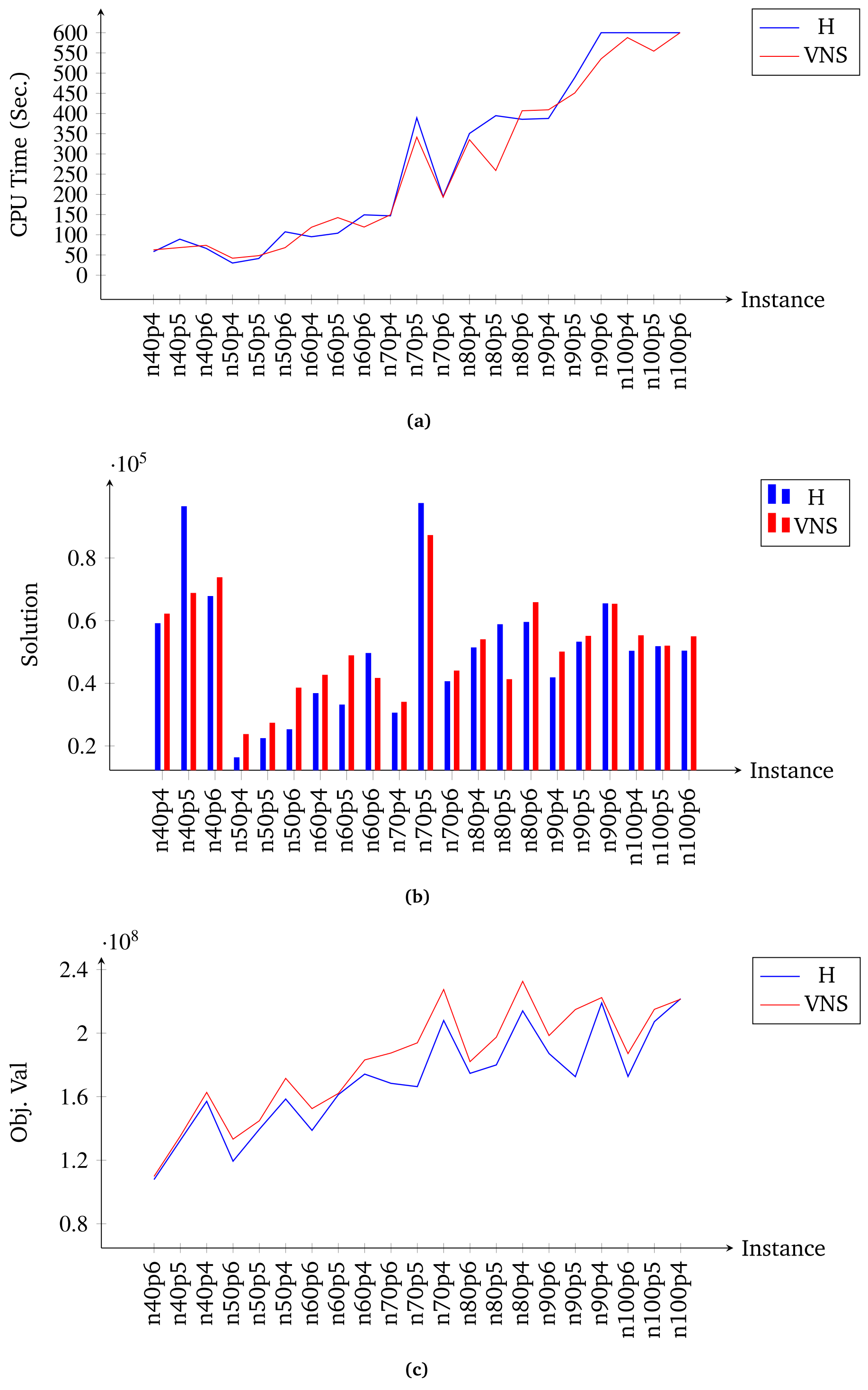


FIGURE 9. Numerical results comparison between the hyper-heuristic and VNS. (a) Time comparison. (b) Number of solutions found. (c) Objective value comparison.

an important part of the supply network. Based on what is being practiced in real life, we have proposed a mathematical model of this problem.

For the proposed mathematical model with exponential number of constraints, we provided separation routines to separate from such constraints in a branch-and-cut manner in CPLEX. Yet, due to the complexity of the problem, CPLEX can only tackle small size instances. We therefore proposed a hyper-heuristic approach for our problem. This approach consists of a set of low-level heuristics and a selection heuristic. The objective of the selection heuristic is to choose the best low-level heuristic to apply at different phases in order to find the best solutions. Our selection heuristic is guided by a reinforcement learning method based on a scoring system. We also improved the efficiency of our hyper-heuristic by adding a powerful perturbation heuristic and a tabu list. In order to be as close as possible to a fair comparison of the hyper-heuristic results, we proposed a VNS meta-heuristic implemented using the same hyper-heuristic low-level heuristics and components.

Our computational experiments confirm that whenever the optimal solution is known, the hyper-heuristic is almost always able to find it in much less computational time. On the other hand, the VNS approach, which inherits the same local search methods, seldom finds an optimal solution. The two approaches find feasible solutions in a reasonable amount of time but the absolute superiority remains with hyper-heuristic both for quality and computational time.

Further work directions include polyhedral analysis, identifying some tightening valid inequalities to improve the polyhedral description and primal decompositions. Furthermore, as the hyper-heuristic shown to be promising, more enhanced learning mechanisms, designing a richer portfolio of low-level heuristics and more sophisticated selection methods deserve more attention.

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. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan and R. Qu, Hyper-heuristics: a survey of the state of the art. *J. Oper. Res. Soc.* **64** (2013) 1695–1724.

[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. INOC 2011, edited by J. Pahl, T. Reiners and S. Voß. Vol. 6701 of *Lecture Notes in Computer Science*. Springer, 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] W. Dai, J. Zhang, X. Sun and S. Wandelt, Hubbi: iterative network design for incomplete hub location problems. *Comput. Oper. Res.* **104** (2019) 394–414.

[15] K. Danach, *Hyperheuristics in Logistics.* Ph.D. thesis, Ecole Centrale de Lille (2016).

[16] K. Danach, S. Gelareh and R. Neamatian 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.

[17] J. Denzinger and M. Fuchs, High performance ATP systems by combining several AI methods. In: Vol. 1 of *IJCAI'97. Proceedings of the 15th International Joint Conference on Artificial Intelligence*. 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. Model.* **37** (2013) 3307–3321.

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

[23] Z. He, Farthest-point heuristic based initialization methods for k-modes clustering. *CoRR*, abs/cs/0610043 (2006).

[24] D. Huang, Z. Liu, X. Fu and P. Blythe, Multimodal transit network design in a hub-and-spoke network framework. *Transp. A: Transp. Sci.* **14** (2018) 706–35.

[25] B. Jarboui, H. Derbel, S. Hanafi and N. Mladenovic, Variable neighborhood search for location routing. *Comput. Oper. Res.* **40** (2013) 47–57.

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

[27] 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 Practice* **27** (1993) 1–12.

[28] J.B. MacQueen, Some methods for classification and analysis of multivariate observations. In: Vol. 1 of *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press (1967) 281–297.

[29] E. Martins 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.

[30] E. Martins 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.

[31] N. Mladenović and P. Hansen, Variable neighborhood search. *Comput. Oper. Res.* **24** (1997) 1097–1100.

[32] 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. Intel.* **50** (2016) 1–19.

[33] 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.

[34] S. Mourelo Ferrandez, T. Harbison, T. Weber, R. Sturges and R. Rich, Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. *J. Ind. Eng. Manage.* **9** (2016) 374.

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

[36] M. O'Kelly, A clustering approach to the planar hub location problem. *Ann. Oper. Res.* **40** (1993) 339–353.

[37] M.P. Pérez, F.A. Rodíguez and J.M. Moreno-Vega, A hybrid VNS-path relinking for the p-hub median problem. *IMA J. Manage. Math.* **18** (2007) 157–171.

[38] J. Pérez-Ortega, N.A.-O. Nelva, A. Vega-Villalobos, R. Pazos-Rangel, C. Zavala-Diaz and A. Martinez-Rebollar, The K-means algorithm evolution, edited by K. Sud, P. Erdogmus and S. Kadry. In: *Introduction to Data Science and Machine Learning*. IntechOpen, Rijeka (2020).

[39] R. Rahmaniani, G. Rahmaniani and A. Jabbarzadeh, Variable neighborhood search based evolutionary algorithm and several approximations for balanced location-allocation design problem. *Int. J. Adv. Manuf. Technol.* **72** (2014) 145–159.

[40] I. Rodriguez-Martin, 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.

[41] I. Rodriguez-Martin, 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.

[42] B. Rostami, N. Kämmerling, C. Buchheim and U. Clausen, Reliable single allocation hub location problem under hub breakdowns. *Comput. Oper. Res.* **96** (2018) 15–29.

[43] E. Serper and S. Alumur Alev, The design of capacitated intermodal hub networks with different vehicle types. *Transp. Res. Part B: Methodol.* **86** (2016) 51–65.

[44] 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.

[45] R. Todosijević, D. Urosevic, N. Mladenovic and S. Hanafi, A general variable neighborhood search for solving the uncapacitated r-allocation p-hub median problem. *Optim. Lett.* **11**(2017) 1109-1121.

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

[47] M.Yahyaei, M. Bashiri and Y. Garmeyi, Multicriteria logistic hub location by network segmentation under criteria weights uncertainty. *Int. J. Eng. Trans. B: App.* **27** (2014) 1205–1214.

[48] 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.

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

[50] 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.

[51] 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.

[52] 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. Distr. Sensor Netw.* **14** (2018).