Hub Network Design with Flexible Routes

Luiza Bernardes Real[a], Ivan Contreras[b], Jean-François Cordeau[c], Ricardo Saraiva de Camargo[d], Gilberto de Miranda[e]

[a] *Instituto Federal de Minas Gerais, Brazil*
[b] *Concordia University and Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada H3G 1M8*
[c] *HEC Montréal and Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada H3T 2A7*
[d] *Departamento de Engenharia de Produção, Universidade Federal de Minas Gerais, Brazil*
[e] *Departamento de Matemática Aplicada, Universidade Federal do Espírito Santo, Brazil*

Abstract

This paper introduces a hub network design problem with flexible routes. The routes are flexible in the sense that each route may contain a mix of hub and non-hub nodes. We assume that commodity transfers can only be performed at hubs and that transportation costs are flow-dependent, which implies that instead of imposing fixed discount factors to represent economies of scale, these economies of scale stem from the transport technology chosen to operate the routes. We propose a mixed integer programming formulation and two metaheuristics based on the adaptive large neighborhood search paradigm to solve the problem. We report the results of computational experiments to assess the performance of the formulation and algorithms on a set of benchmark instances.

1. Introduction

Hubs are special facilities that act as transshipment or flow consolidation points in many-to-many transport applications. *Hub location problems* (HLPs) are a class of strategic logistics planning problems in which hub facilities must be located while demand nodes must be allocated to hubs in order to route the traffic between multiple origin-destination pairs. In hub-and-spoke systems, instead of directly connecting each origin-destination pair, flows from the same origin but with different destinations are consolidated at the hubs with other flows that have different origins but the same destination. Consolidating demands at hub terminals allows transportation carriers to use larger vehicles to exploit economies of scale and achieve lower transportation costs. Given their wide presence

Email addresses: luiza.real@ifmg.edu.br (Luiza Bernardes Real), icontrer@encs.concordia.ca (Ivan Contreras), jean-francois.cordeau@hec.ca (Jean-François Cordeau), rcamargo@dep.ufmg.br (Ricardo Saraiva de Camargo), miranda@ufes.br (Gilberto de Miranda)

in logistics systems, HLPs have been the focus of many researchers. Readers may refer to Alumur and Kara (2008), Campbell and O'Kelly (2012), Farahani et al. (2013) and Contreras and O'Kelly (2019) for recent surveys on hub location.

Classic HLPs rely on three main assumptions. First, demand flows have to be routed through one or at most two hubs, implying that a non-hub node is directly connected to at least one hub facility. Second, the hub network is considered to be fully interconnected. Third, a constant discount factor representing economies of scale is applied to the unit transportation cost of inter-hub connections. In several applications, these assumptions are reasonable and provide a good approximation of reality; in others, they may lead to suboptimal solutions.

Considering a fixed discount factor to represent economies of scale on inter-hub links may be more suitable for applications in which links between hubs are associated with faster transportation modes. Nonetheless, it may be an oversimplification for applications in which economies of scale are a consequence of the bundling of flows on hub arcs. This simplification may lead to solutions that grant a discount factor to inter-hub arcs only, even though these connections may carry considerably less flow than non-hub links. Consequently, in many cases, this may lead to miscalculations of the total network cost as well as to erroneous decisions of hub locations and non-hub allocations.

Several modeling approaches have been introduced to overcome this simplification. For instance, O'Kelly and Bryan (1998) and de Camargo et al. (2009) approach this issue by proposing formulations in which the economies of scale are modeled as flow dependent functions on the inter-hub arcs in a fully interconnected hub network. Nonetheless, they neither consider the selection of vehicles nor how they operate. Kimms (2006) argue that economies of scale stem from the transportation technology selected to be used in the system. He proposes three multiple allocation p-hub median problems with direct service and with fixed and variable costs. The goal is to determine the optimal number of vehicles to be used on each arc of a fully interconnected hub network. Considering flow dependent transportation costs based on modular arc costs, Tanash et al. (2017) study a modular hub location problem, which bears similarity to the work of Kimms (2006). The total transportation cost is calculated not in terms of the per unit flow cost, but with respect to the number of vehicles used on each arc. Although these models design the hub-and-spoke network and assign resources to arcs aiming to minimize the total cost of transporting flows, the routing of the resources is not taken into account.

The need to jointly consider location and routing decisions arises naturally in applications that have nodes with insufficient demand to justify direct connections with the hubs

such as in many-to-many flow transportation systems. In these cases, besides the location of hubs and routing of flows through the network, local tours that serve and connect the non-hub nodes to the installed hubs need to be generated so that flows from many origins to many destinations can be transported at a minimal cost. This gives rise to the so-called *many-to-many hub location-routing problem* (MMHLRP) introduced by Nagy and Salhi (1998). The authors study a MMHLRP that imposes capacity and maximum distance constraints to the local tours of a homogeneous fleet, and fixed costs to establishing hubs.

Wasner and Zäpfel (2004) consider a MMHLRP arising in a postal service application in which non-hub nodes are directly linked by local tours while all inter-hub flows are transferred through a central hub. Another variant of a MMHLRP is presented in Çetiner et al. (2010) in which an iterative two-stage solution procedure is described. In the first stage, hub location and non-hub node (multiple) assignment decisions are fixed whereas in the second stage a traveling salesman problem is solved for each installed hub. Camargo et al. (2013) and Rodríguez-Martín et al. (2014) study single allocation variants of the problem with bounded tour length to ensure service quality and present exact solution algorithms for solving them. Kartal et al. (2017) propose three different mixed integer programs (MIP) and two heuristic approaches to solve a single allocation p-hub median location and routing problem with simultaneous pickup and delivery. Karimi (2018) study a version of a single allocation hub location routing problem that considers hub and vehicle capacities. Kartal et al. (2019) present a p-hub center and routing network design problem which locates p-hubs, allocates non-hub nodes to the installed hubs, and generates vehicle routes for each installed hub in a way that the maximum travel time between all origin-destination pairs is minimized.

Most of the work on MMHLRPs assume that routing decisions happen only at the access level, and that the hub network is fully interconnected with one vehicle serving each pair of hubs. Nonetheless these assumptions may not lead to the most cost efficient topologies, since they may result in unnecessary routes. To prevent that, Lopes et al. (2016) investigate a MMHLRP with an incomplete hub network. Nodes are partitioned into tours with exactly one hub each, while creating an extra tour interconnecting all hubs. Hubs are restricted to serve a predetermined number of non-hub nodes, whereas the many-to-many flow decisions are implicitly made. Neither flow capacity on the tours or hubs nor maximal length on the routes are imposed.

In this paper we introduce a new general class of MMHLRPs denoted as *hub network design problems with flexible routes* (HNDPs). Besides considering hub location decisions and flow dependent transportation costs, HNDPs determine itineraries for the selected

vehicles. One of the major advantages of HNDPs over classical models is that no particular topological structure, such as cycles (Contreras et al., 2017), stars (Labbé and Yaman, 2008), trees (Martins de Sá et al., 2013), lines (Martins de Sá et al., 2015), incomplete (de Camargo et al., 2017) or fully interconnected hub networks (O'Kelly, 1986; Skorin-Kapov et al., 1996; Hamacher et al., 2004) is assumed a priori. The network topology is endogenously determined, being induced by the involved costs. This ensures that vehicles will be used more efficiently and economically, and that economies of scale will be obtained according to the transport technology chosen to operate the routes and their actual utilization.

HNDPs consist of opening a set of hub nodes and covering a set of nodes with a set of heterogeneous vehicle routes. Vehicles are assumed to start their route on any node of the graph, serve some nodes, and then return to the same initial node. Routes are not forced to visit hub nodes. Load capacity constraints and a maximum time for completing the routes must be respected. There are installation costs for establishing a hub and using a vehicle, while fixed operational costs are incurred every time a vehicle moves through an arc. We assume that all demand flows have to be routed with the selected vehicle routes. Instead of applying the traditional fixed discount factor to inter-hub links, it is assumed that economies of scale are a direct consequence of the vehicles assigned to the routes and not of the arc type being used. No restriction of any kind is imposed on the path of a demand flow. That is, demand flow can be routed from its origin to its destination by using a single vehicle route or by using a set of vehicle routes. However, flow transfers between vehicles are allowed only at hub nodes. Here, a hub works exclusively as a transshipment facility to transfer flows from one vehicle to another. There is no limit on the number of intermediate nodes (hub and non-hub) a path of a demand flow can have.

Figure 1 illustrates a solution of a HNDP with four hub nodes, 13 non-hub nodes and seven vehicle routes. Note that there are routes that start and end at non-hub nodes, and that, even though there are multiple routes passing through the same non-hub node, there is no flow exchange on non-hub nodes. It is also worth observing that demand flows follow the direction of the routes, i.e. an origin-destination demand (i, j) may have a longer path than its counterpart demand (j, i).

HNDPs greatly differ from previously studied MMHLRPs. They assume a given exogenous network topology and a homogeneous fleet with routes that start and end at the hubs. Normally vehicles can only pass a limited number of times through each demand node, while the path of a demand flow is restricted by the prescribed network topology. Most important of all, MMHRLPs completely disregard that the economies of scale derive

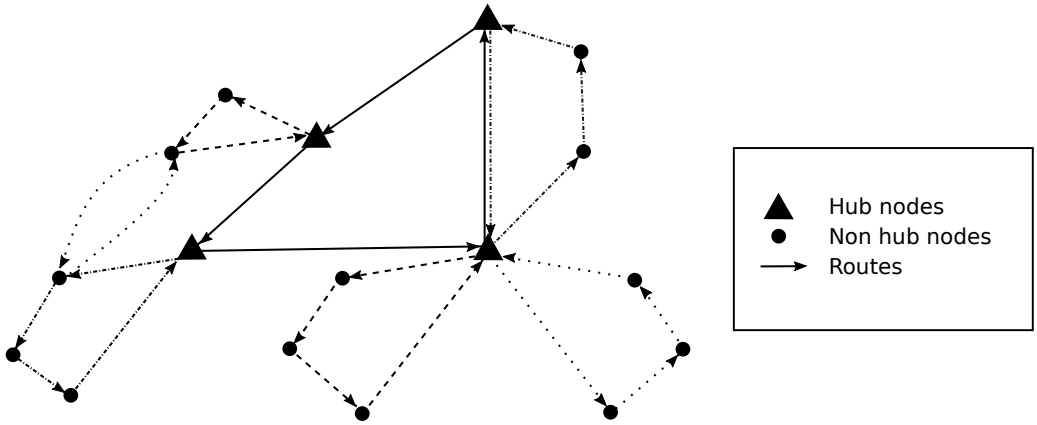

Figure 1: Example of a hub network with flexible routes

from the technology chosen to operate the routes and its utilization. To better highlight the similarities and dissimilarities between the HNDP and other MMHLPRP, Table 1 summarizes their main characteristics.

Table 1: Related literature to the HNDP

Research	Components						
	Vehicle Capacity	Route Length	Hub Network	Fleet Type	Allocation	Vehicle Cost	Hub Cost
Nagy and Salhi (1998)	✓	✓	FI	HO	SA		✓
Wasner and Zäpfel (2004)	✓	✓	FI	HO	MA		
Çetiner et al. (2010)		✓	FI	HO	MA		
Camargo et al. (2013)		✓	FI	HO	SA	✓	✓
Rodríguez-Martín et al. (2014)		✓	FI	HO	SA		
Lopes et al. (2016)		✓	PI	HO	SA		
Kartal et al. (2017)			FI	HO	SA		
Karimi (2018)	✓	✓	FI	HO	SA	✓	✓
Kartal et al. (2019)			FI	HO	SA		
Current research	✓	✓	PI	HE	MA	✓	✓

FI - Full Interconnected; PI - Partial Interconnected; SA - Single Allocation; MA - Multiple Allocation.
HO - Homogeneous fleet; HE - Heterogeneous fleet.

To the best of our knowledge, no work in the literature considers simultaneously hub location and vehicle routing decisions with capacity constraints, without assuming a constant discount factor to represent economies of scale, and without imposing any restriction on the network topology and on the paths used to route demand flows. Our goal is to propose a more general framework capable of designing hub networks that employ several vehicle routes to transport the demands via a heterogeneous fleet. These assumptions, together with the flexibility given to the routes, make the HNDP much more

difficult to formulate and solve when compared to previously studied hub network design problems involving routing decisions.

The HNDPs have numerous applications in ground, air and maritime freight transportation networks. An example of a concrete application arises in liner shipping in which companies usually operate a set of cyclic routes to provide shipping services. In this case, hub facilities correspond to ports where transshipment of cargo can occur and non-hub nodes can be seen as ports where transshipment is not allowed. Demand flows represent requests from customers for shipment of containers. Vessel routes may or may not contain hubs on them and there is no restriction in the path chosen to serve the requests. The goal of liner shipping companies is to design vessel routes at a minimal cost to satisfy the demand. Although shipping service routes cannot be reshuffled overnight, several circumstances, e.g. changes in demand, increases in fuel price and modifications in ship capacity, drive a liner shipping company to adjust its service routes and ship deployment on a small scale from time to time (see, for instance, Gelareh and Pisinger, 2011; Reinhardt and Pisinger, 2012; Song and Dong, 2013).

The main contribution of this paper is threefold. First, we introduce a new class of HLPs that encompass both strategic and tactical planning decisions. The strategic decisions consist of locating hub facilities and defining how many vehicles of each type should be used. Given that transforming a facility into a hub and owning a vehicle involve a significant capital investment, the decisions taken at this level are very important. The tactical decisions assign vehicles to arcs so that vehicle routes can define paths to route all demand flows. While these decisions can be taken separately, once strategic decisions are made, they may negatively impact the decisions of the tactical level, resulting in suboptimal solutions. Second, we propose a MIP formulation to solve the problem. Third, because the proposed MIP is too large to be solved in a timely manner by general purpose solvers, we also develop two metaheuristics based on an adaptive large neighborhood search (ALNS). To evaluate the efficiency and limitations of our algorithms, extensive computational experiments are performed on instances with up to 50 nodes.

The remainder of this document is organized as follows. Section 2 provides a formal definition of the problem and presents the mathematical formulation. Section 3 describes in detail the two ALNS metaheuristics. Computational results are reported in Section 4, followed by conclusions in the last section.

2. Problem definition and formulation

Let $G = (N^O, A^O)$ be a directed graph, with the node set $N^O = \{0, .., n\}$ and the arc set $A^O = \{(i, j) : i, j \in N^O; i \neq j\}$. Node 0 is a fictitious depot, where vehicles must start and end their routes. The sets $N = \{1, ..., n\}$ and $A = \{(i, j) : i, j \in N; i \neq j\}$ are the node and arc sets, respectively, without considering the depot. We define $H \subseteq N$ as the hub candidate set and h_k as the fixed cost incurred for installing a hub in node $k \in H$. There is a demand flow $w_{ij} \geq 0$ that needs to be routed from $i \in N$ to $j \in N$. We assume that w_{ij} can be split, i.e., can be routed using more than one path. We also consider that more than one vehicle can be used to serve a demand in a specific path. However, demand flows can change vehicles only at hub nodes. For example, consider the path $1-2-3-4$ to serve demand w_{14}. If vehicle 1 operates arc $(1, 2)$ and vehicle 2 operates arcs $(2, 3)$ and $(3, 4)$, a hub must be installed at node 2.

We consider an heterogeneous fleet of vehicles in which each vehicle type has a different capacity, speed, fuel consumption, and other specific characteristics. We denote by $\mathcal{R}$ the set of all vehicle types and use the index r to represent a particular vehicle type. For each $r \in \mathcal{R}$, τ^r denotes the capacity of a vehicle of type r and for each arc $(k, m) \in A$, t^r_{km} denotes the time it takes a vehicle of type r to traverse arc (k, m). The fixed cost related to the acquisition of a vehicle type $r \in \mathcal{R}$ is denoted by a^r. We associate the fixed cost c^r_{km} with vehicle type $r \in \mathcal{R}$ traversing arc $(k, m) \in A$. Finally, let T be an upper bound on the length of the routes.

Let $\rho(r)$ be a function that returns the number of available vehicles of type r, and R be the total number of available vehicle types. Let $P = \{1, ..., \rho(1), \rho(1) + 1, ..., \rho(1) + \rho(2), ..., \rho(1) + ... + \rho(R)\}$ be the set of vehicles, and index p be used to represent a specific vehicle. Let $\phi(p)$ be a function that returns the type of vehicle p. For example, if we have three types of vehicles ($R = 3$), having three vehicles for type 1 ($\rho(1) = 3$), four vehicles for type 2 ($\rho(2) = 4$), and two vehicles for type 3 ($\rho(3) = 2$), then set $P = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. For this case, vehicles $1, 2, 3$ are of type 1 ($\phi(1) = \phi(2) = \phi(3) = 1$), vehicles $4, 5, 6, 7$ are of type 2 ($\phi(4) = \phi(5) = \phi(6) = \phi(7) = 2$), and vehicles 8 and 9 are of type 3 ($\phi(8) = \phi(9) = 3$).

We now present an MIP formulation for the HNDP. Our formulation combines the decisions of two problems: 1) variables $\mathbf{y}$ and $\mathbf{f}$ related to the hub location problem, and 2) variables $\mathbf{q}$, $\mathbf{x}$ and $\mathbf{z}$ associated to the vehicle routing problem. Let $y_k \in \{0, 1\}$ be equal to one if and only if a hub is located at node $k \in H$. Let $f^p_{ijkm} \geq 0$ denote the percentage of demand w_{ij} passing through arc $(k, m) \in A$ in vehicle $p \in P$. Further, let $q^p \in \{0, 1\}$ be equal to one if and only if vehicle $p \in P$ is utilized. Also, let $x^p_{km} \in \{0, 1\}$ be equal to

one if and only if vehicle $p \in P$ passes through arc $(k,m) \in A^0$, and $z_k^p \in \{0,1\}$ be equal to one if and only if vehicle $p \in P$ passes through node $k \in N$. Using these variables, the HNDP can be formulated as follows:

$$\min \sum_{k \in H} h_k y_k + \sum_{p \in P} a^{\phi(p)} q^p + \sum_{p \in P} \sum_{(k,m) \in A} c_{km}^{\phi(p)} x_{km}^p \tag{1}$$

$$\text{s.t.:} \sum_{p \in P} \sum_{(i,m) \in A} f_{ijim}^p = 1 \qquad \forall (i,j) \in W \tag{2}$$

$$\sum_{p \in P} \sum_{(k,m) \in A} f_{ijkm}^p = \sum_{p \in P} \sum_{(m,k) \in A} f_{ijmk}^p \qquad \forall (i,j) \in W, k \in N : k \neq i, k \neq j \tag{3}$$

$$\sum_{p \in P} \sum_{(k,j) \in A} f_{ijkj}^p = 1 \qquad \forall (i,j) \in W \tag{4}$$

$$f_{ijkm}^p \leq x_{km}^p \qquad \forall (i,j) \in W, (k,m) \in A, p \in P : k \neq j, m \neq i \tag{5}$$

$$\sum_{\substack{(m,k) \in A \\ m \neq j}} f_{ijmk}^p - \sum_{\substack{(k,m) \in A \\ m \neq i}} f_{ijkm}^p \leq y_k \qquad \forall (i,j) \in W, k \in N, p \in P : k \neq i, k \neq j \tag{6}$$

$$\sum_{(k,m) \in A^0} x_{km}^p = \sum_{(m,k) \in A^0} x_{mk}^p \qquad \forall k \in N, p \in P \tag{7}$$

$$\sum_{(0,m) \in A^O} x_{0m}^p = q^p \qquad \forall p \in P \tag{8}$$

$$\sum_{(k,0) \in A^O} x_{k0}^p = q^p \qquad \forall p \in P \tag{9}$$

$$\sum_{(k,m) \in A} t_{km}^{\phi(p)} x_{km}^p \leq q^p T \qquad \forall p \in P \tag{10}$$

$$\sum_{(k,m) \in A} x_{km}^p = z_k^p \qquad \forall k \in N, p \in P \tag{11}$$

$$\sum_{(k,m) \in A} x_{km}^p = z_m^p \qquad \forall m \in N, p \in P \tag{12}$$

$$\sum_{u \in S} \sum_{v \in N^O \setminus S} x_{uv}^p + \sum_{u \in S} \sum_{v \in N^O \setminus S} x_{vu}^p \geq z_k^p \qquad \forall k \in S, S \subset N, |S| \geq 2, p \in P \tag{13}$$

$$x_{km}^p \leq q^p \qquad \forall p \in P, (k,m) \in A \tag{14}$$

$$\sum_{(i,j) \in W} w_{ij} f_{ijkm}^p \leq \tau^{\phi(p)} x_{km}^p \qquad \forall (k,m) \in A, p \in P \tag{15}$$

$$f_{ijkm}^p \geq 0 \qquad \forall (i,j) \in W, (k,m) \in A, p \in P : k \neq j, m \neq i \tag{16}$$

$$y_k \in \{0,1\} \qquad \forall k \in H \tag{17}$$

$$x_{km}^p \in \{0,1\} \qquad \forall (k,m) \in A^0, p \in P \tag{18}$$

$$q^p \in \{0,1\} \qquad \forall p \in P \tag{19}$$

$$z_k^p \in \{0,1\} \qquad \forall k \in N, p \in P. \tag{20}$$

The objective function (1) minimizes the total cost of designing the network and routing the vehicles. The first term of the objective function denotes the cost of installing hubs. The second term represents the cost of acquiring the vehicles, while the third term calculates the vehicle routing cost of traversing the arcs. Constraints (2)-(4) are flow balancing equations. Constraints (5) ensure that if any flow passes through an arc in a vehicle, then, this vehicle passes through such arc. Constraints (6) guarantee that if a demand flow changes vehicles in a node, then, such node is a hub. Constraints (7) balance the vehicle routes, i.e., if a vehicle arrives at a node, it must leave that node. Constraints (8) and (9) guarantee that the acquired vehicles start and end their routes at the fictitious depot. Constraints (14) ensure that a vehicle passes through an arc only if the vehicle is acquired. Constraints (15) are the capacity constraints, i.e. the total flow that passes through an arc in a vehicle must respect the capacity of that arc. Constraints (10) impose a travel time limit on each vehicle route. Constraints (11) and (12) activate variables z, i.e. if a vehicle traverses an arc, the nodes of that arc are in the vehicle route. Constraints (13) are the well-known sub-tour elimination constraints (SECs) which ensure that for each route sub-cycles will not be formed. Constraints (16)-(20) are the standard non-negativity and integrality conditions.

Let $\rho^s(r)$ and $\rho^e(r)$ be the first and the last indices of vehicle of type r, respectively. In order to reduce the symmetry of our formulation (1)-(20), we add the following constraints:

$$\sum_{(k,m)\in A} t_{km}^r x_{km}^p \leq \sum_{(k,m)\in A} t_{km}^r x_{km}^{p-1} \qquad \forall r \in \mathcal{R}, p \in P : p > \rho^s(r), p \leq \rho^e(r) \tag{21}$$

$$q^p \leq q^{p-1} \qquad \forall r \in \mathcal{R}, p \in P : p > \rho^s(r), p \leq \rho^e(r). \tag{22}$$

Constraints (21) ensure that longer routes will have vehicles indexed with smaller values for each type of vehicle. Constraints (22) guarantee that vehicles with a smaller index will be activated before vehicles with higher indices for each type of vehicle.

3. Solution algorithms

We next present two metaheuristics based on an ALNS framework to find feasible solutions for the HNDP in reasonable computation times. The ALNS framework was originally devised by Ropke and Pisinger (2006) for the pickup and delivery problem with

time windows. It can be seen as an extension of the large neighborhood search framework of Shaw (1998) but with an adaptive layer. Our first algorithm, denoted as TDALNS, follows a top down approach, in which the hub configuration is defined first followed by the construction of the routes. The second algorithm, denoted as BUALNS, follows a bottom up approach, in which the routes are built before the hub configuration is established.

Both algorithms are iterative procedures in which at every iteration, two main phases are performed. The first phase selects a *destroy* operator to remove n^d nodes from the routes, while the second phase chooses a *repair* operator to reconstruct the solution. Operators are selected according to an adaptive probabilistic mechanism derived from the score of each operator. Operators that have successfully found new improving solutions have a higher score and, therefore, a higher probability to be chosen again.

A general overview of TDALNS and BUALNS heuristics is given in Algorithms 1 and 2, respectively. An initial feasible solution s is first constructed. A new solution s^{new} is then obtained by sequentially applying destroy and repair operators on the current solution s. In the TDALNS, whenever ω^{ih} iterations have been performed, a hub configuration operator is chosen to modify the current network before applying the destroy and the repair operators. In the BUALNS, the hub configuration is the last decision to be fixed, which is done at the same time the feasibility of the solution is checked. If the new solution is disconnected, a connection procedure is called to change the network.

In the TDALNS, we verify if the network is connected with respect to the hub configuration. In our case, the network is sometimes strongly connected, but there may be paths in which the transshipment occurs at non-hub nodes, thus violating our assumption that flow transfers between vehicles can only occur at hub nodes. Recall that a flow can pass from arc (k, m) to arc (m, l) only if the same vehicle passes through both arcs or if node m is a hub. Therefore, if the current set of open hubs does not guarantee the feasibility of the network, the solution is rejected. Otherwise, we verify if there is enough capacity in the selected vehicle routes to serve all demand pairs. For the BUALNS, once the network is connected we call a procedure that defines the hub configuration and analyzes the capacity feasibility at the same time. In both algorithms, if there is not enough allocated capacity on the vehicle route to serve all demand pairs, the solution is dynamically penalized by taking into account its capacity violation. The algorithms terminate when a limit on the maximum number of iterations is reached.

The algorithm always accepts a better solution, whereas a worse solution can be accepted depending on a simulating annealing criterion. Infeasible solutions can also be accepted. We note that there is no guarantee that the objective value of an infeasible

solution will always be better than the objective value of an optimal solution. For this reason, at each iteration, we keep the best solution in general (s^*) and the best feasible solution (s^{f*}).

After destroying and repairing a solution, the resulting network may be infeasible for three different reasons. First, the network can be disconnected. In that case, a procedure is called to make it connected. Second, the network can be connected but to serve all the demand pairs some non-hub nodes may be used as transshipment points, which violates our assumption that flows can be transferred from one vehicle to another only at hub nodes. This situation may arise only in the TDALNS (and when it does, the solution is discarded). In the BUALSN, the hub configuration is made in a way that the feasibility from this point of view is always guaranteed. Third, the allocated capacity may not be enough to serve all the demand pairs. When this happens, the solution is penalized and it can be accepted or not according to the simulated annealing criterion. The next subsections describe in detail all the procedures used in Algorithms 1 and 2.

3.1. Initial solution

To construct an initial feasible solution, we divide the nodes into clusters by solving a p-median problem. In this problem, central nodes correspond to facilities and non-central nodes correspond to non-facility points. The objective is to minimize the sum of the distances between the chosen central nodes and the nodes allocated to each of them. This problem is formulated as an MIP and solved with CPLEX. Each cluster has then its central node with its respective assigned non-central nodes. For each cluster, we find a Hamiltonian cycle of minimal cost. To construct Hamiltonian cycles, we used the Concorde solver (Applegate et al., 2006). We also create a simple route for every pair of clusters that connects their central nodes. If the duration of any tour is greater than the maximum duration allowed, the tour is split into feasible tours. Even though we assign the largest vehicle type to perform all the tours, there is no guarantee that the network will have enough capacity to serve all the demand pairs. To define the hub configuration and to verify if any route needs to be duplicated, we solve an auxiliar mixed integer problem that we denote as the *routing flow problem* (RFP).

The RFP determines which hubs should be opened and whether the initial solution is feasible or if it would be necessary to add capacity to the network by duplicating some vehicle routes. Let $\mathcal{P}$ be the set of all routes in a solution. We redefine variables f considering only the arcs that exist in the solution network. In other words, arc $(k, m) \in A^{\mathcal{P}}$ is considered if there is any $p \in \mathcal{P}$ with $x^p_{km} = 1$. Let q^p be a decision variable

Algorithm 1 Basic steps of TDALNS

$s \leftarrow$ InititalSolution
$s^* \leftarrow s$
$s^{f*} \leftarrow s$
InitializeScores(π^D, π^R, π^H)
$ih \leftarrow 0, is \leftarrow 0$
repeat
  if $ih = \omega^{ih}$ **then**
    $O^h \leftarrow ChooseHubConfigurationOperator(H, \pi)$
    $s^{new} \leftarrow HubConfigurationOperators(O^h, s)$
    $ih \leftarrow 0$
  else
    $ih \leftarrow ih + 1$
  end if
  $O^d \leftarrow ChooseDestroyOperator(D, \pi)$
  $O^r \leftarrow ChooseRepairOperator(R, \pi)$
  $s^{new} \leftarrow DestroyAndRepairOperators(O^d, O^r, s^{new})$
  if $s^{new}$is disconnected **then**
    $s^{new} \leftarrow ConnectNetwork(s^{new})$
  end if
  if $s^{new}$is connected from a hub configuration point of view **then**
    $(s^{new}, feasibleSolution, hubFeasibility) \leftarrow AnalyzeCapacityFeasibility(s^{new})$
    if $feasibleSolution = false$ **then**
      if $hubFeasibility = true$ **then**
        Penalize($OF(s^{new})$)
      end if
    end if
    if $feasibleSolution = true$ or $hubFeasibility = true$ **then**
      if s^{new}satisfies the accepetance criterion **then**
        $s \leftarrow s^{new}$
      end if
      if $OF(s^{new}) < OF(s^*)$ **then**
        $s^* \leftarrow s^{new}$
      end if
      if $feasibleSolution = true$ **then**
        if $OF(s^{new}) < OF(s^{f*})$ **then**
          $s^{f*} \leftarrow s^{new}$
        end if
      end if
    end if
  end if
  if $is = \omega^{is}$ **then**
    UpdateScores(π^D, π^{R,π^H})
    $is \leftarrow 0$
  else
    $is \leftarrow is + 1$
  end if
until {the stopping condition is met}

Algorithm 2 Basic steps of BUALNS

$s \leftarrow$ InititalSolution
$s^* \leftarrow s$
$s^{f*} \leftarrow s$
InitializeScores(π^D, π^R)
$is \leftarrow 0$
repeat
  $O^d \leftarrow ChooseDestroyOperator(D, \pi)$
  $O^r \leftarrow ChooseRepairOperator(R, \pi)$
  $s^{new} \leftarrow DestroyAndRepairOperators(O^d, O^r, s^{new})$
  if $s^{new}$is disconnected **then**
    $s^{new} \leftarrow ConnectNetwork(s^{new})$
  end if
  $(s^{new}, feasibleSolution) \leftarrow AnalyzeCapacityFeasibilityAndDefineHubConfiguration(s^{new})$
  if $s^{new}$is connected from a hub configuration point of view **then**
    $(s^{new}, feasibleSolution, hubFeasibility) \leftarrow AnalyzeCapacityFeasibility(s^{new})$
    if $feasibleSolution = false$ **then**
      if $hubFeasibility = true$ **then**
        Penalize($OF(s^{new})$)
      end if
    end if
    if $feasibleSolution = true$ or $hubFeasibility = true$ **then**
      if s^{new}satisfies the accepetance criterion **then**
        $s \leftarrow s^{new}$
      end if
      if $OF(s^{new}) < OF(s^*)$ **then**
        $s^* \leftarrow s^{new}$
      end if
      if $feasibleSolution = true$ **then**
        if $OF(s^{new}) < OF(s^{f*})$ **then**
          $s^{f*} \leftarrow s^{new}$
        end if
      end if
    end if
  end if
  if $is = \omega^{is}$ **then**
    UpdateScores(π^D, π^R)
    $is \leftarrow 0$
  else
    $is \leftarrow is + 1$
  end if
until {the stopping condition is met}

to determine the number of times route p will be used, and let $\mathcal{C}^p$ be a parameter that represents the total cost of route p. Let $c_{km}^{\phi(p)}$ be the estimated cost of transporting a unit flow through the arc (k,m) of vehicle p, i.e. $c_{km}^{\phi(p)} = \mathcal{C}^p/(d_{km} \times \tau^{\phi(p)})$. Using these variables, we formulate the RFP as the following MIP:

$$\min \sum_{k\in N} h_k y_k + \sum_{p\in P} \mathcal{C}^p q^p + \sum_{p\in P} \sum_{(i,j)\in W} \sum_{(k,m)\in A^p} c_{km}^{\phi(p)} f_{ijkm}^r \tag{23}$$

$$\text{s.t.:} \sum_{p\in\mathcal{P}} \sum_{(i,m)\in A^p} f_{ijim}^p = 1 \qquad \forall (i,j) \in W \tag{24}$$

$$\sum_{p\in\mathcal{P}} \sum_{(k,m)\in A^p} f_{ijkm}^p = \sum_{p\in\mathcal{P}} \sum_{(m,k)\in A^p} f_{ijmk}^p \quad \forall (i,j) \in W, k \in N : k \neq i, k \neq j \tag{25}$$

$$\sum_{p\in\mathcal{P}} \sum_{(k,j)\in A^p} f_{ijkj}^p = 1 \qquad \forall (i,j) \in W \tag{26}$$

$$\sum_{\substack{(u,k)\in A^p \\ u\neq j}} f_{ijuk}^p - \sum_{\substack{(k,v)\in A^p \\ v\neq i}} f_{ijkv}^p \leq y_k \qquad \forall (i,j) \in W, k \in N, p \in \mathcal{P} : k \neq i, k \neq j \tag{27}$$

$$\sum_{(i,j)\in W} w_{ij} f_{ijkm}^p \leq \tau^{\phi(p)} q^p \qquad \forall (k,m) \in A^p, p \in \mathcal{P} \tag{28}$$

$$f_{ijkm}^p \geq 0 \qquad \forall (i,j) \in W, (k,m) \in A^p, p \in \mathcal{P} : k \neq j, m \neq i \tag{29}$$

$$q^p \geq 0 \qquad \forall p \in \mathcal{P} \tag{30}$$

$$y_k \in \{0,1\} \qquad \forall k \in N. \tag{31}$$

The objective function minimizes the cost of opening hubs, activating routes and routing demand flows. Constraints (24)-(28) are equivalent to constraints (2)-(4), (6), and (15), respectively. Constraints (29)-(31) show the variable domains. After solving the RFP, if a route is used more than once, i.e., $q^p > 1$, we create a copy of this route and add it to the set $\mathcal{P}$. If no flow passes through a route, then this route is removed from the set $\mathcal{P}$.

3.2. Destroy operators

The destroy phase consists of either removing n^d nodes from their respective routes or deleting r^d routes from the current solution. Each operator uses a different metric to choose the nodes/routes to be destroyed. The nodes/routes are always ordered from the best to the worst according to the operator metric. To introduce some randomization, instead of picking the best element (the one in the first position), we randomly choose a number $\sigma \in [0,1)$, and then select the closest position to the σ^p value. The parameter p controls the desired level of diversification. With $p = 1$, the metric is ignored and

the choice is completely random. With $p = \infty$, the best element is always chosen. As proposed by Shaw (1997), we use $p = 4$. The six destroy operators are:

- **Choose more expensive routes** (D^1): For each route r, we calculate an average cost per unit and per distance: $\mathcal{C}^r/(d^r \times \tau^r)$, in which $\mathcal{C}^r, d^r$ and τ^r are the total cost, total distance and capacity of the route r, respectively.

- **Choose shortest routes** (D^2): The routes are ordered in a non-decreasing order with respect to their length.

- **Randomly choose nodes** (D^3): There is no metric in this operator. A node is randomly selected and removed from all the routes it belongs to.

- **Unit cost greedy savings node selection** (D^4): For each candidate node i, we define $\beta(i, r)$ as the difference between the total cost of route r with and without node i. The nodes are always removed from all the routes they belong to. For this operator, the metric of each node is calculated as $\sum_{r \in P: i \in r} \beta(i, r)$, where the nodes are sorted in non-increasing order.

- **Shaw removal distance** (D^5): This operator and the following one are based on the Shaw removal heuristic used by Shaw (1997) and Ropke and Pisinger (2006). They remove nodes that are similar in some aspect. A route is first randomly chosen and then, a node k of such route is randomly chosen to be removed from this and other routes it may belong to. Then, the nodes are sorted, in relation to node k, from the closest to the furthest one. To introduce some randomization, distances are perturbed by a factor within the range $[0.80, 1.20]$.

- **Shaw removal demand** (D^6): A route is first randomly chosen and then, a node k of this route is randomly chosen to be removed from this and other routes it may belong to. Then, for each node $i \neq k$, we calculate the total demand between such nodes as $w_{ik} + w_{ki}$. The nodes with the largest values have a larger probability to be removed. To introduce some randomization, demands are perturbed by a factor within the range $[0.80, 1.20]$.

3.3. Repair operators

The repair operators insert the removed nodes into existing routes or generate new routes when the insertion is not possible. The operators are also selected according to the values of σ^p and are inspired by the basic greedy heuristic proposed by Shaw (1997).

- **Greedy distance route insertion (R^1)**: We define Δd_{ir} as the change in the distance of route r incurred by inserting node i into a position that increases the distance the least. If it is not possible to insert node i into route r, we set $\Delta d_{ir} = \infty$. We randomly select one node i between the nodes that have been removed in the previous phase. For this node i, the routes are ordered from the lowest to the highest Δd_{ir} values. The route that is at the position closest to σ^p is chosen to be the new route of i. Once the route is chosen, node i is inserted in the position that increases the duration of the route the least. During the insertion of a node, we recalculate Δd_{ir} for all nodes that have not been reinserted yet.

- **Greedy demand insertion (R^2)**: For each node removed i and for each existing route r, we calculate the change in the distance incurred by inserting i into position p of route r, Δd_{irp}. For each node i and route r, we calculate the demand between i and all the nodes belonging to r, D_{ir}. We randomly select node i among the nodes that have been removed in the previous phase. Then, for this node i, the route-position pairs (r, p) are sorted in non-increasing order of D_{ir} and non-decreasing order of Δd_{irp}. The idea is to try to insert node i into a route that contains nodes having the largest demand flow to and from node i. The route-position (r, p) that is the closest to the σ^p position value is chosen. During the insertion of a node, we recalculate Δd_{irp} and D_{ir} for all nodes that have not been reinserted yet.

- **Greedy node selection (R^3)**: We define Δd_{ir} as the change in the distance of route r incurred by inserting node i in a position that increases the distance the least. If it is not possible to insert i into route r, we set $\Delta d_{ir} = \infty$. We define c_i as the "cost" of inserting node i at its overall best position, such as $c_i = \min_{r \in R}\{\Delta d_{ir}\}$. Finally, we sort the nodes that need to be reinserted from the smallest to the highest value of c_i. We first choose a node to be reinserted according to the σ^p value, and insert it at its minimum cost position. During the insertion of a node, we recalculate c_i for all nodes that have not been reinserted yet.

After executing the repair operator, we verify if the capacity incident to any node is less then the incoming/outgoing flow of that node. If so, we try to insert the node in a route according to repair operator R^1. If it is not possible, we store such node in a list L. After verifying the capacity incident to each node, if there is at least one node in L, we generate a Hamiltonian cycle of minimal cost containing the nodes in L, splitting it if necessary. Every time we have just one node in L, we choose another node k to form a

route between them. This node is randomly chosen between the three nodes from and to which node i has the greatest demand.

3.4. Verifying the connectivity of the solution

There is no guarantee that the solution network obtained after applying the destroy and repair operators is connected. If this network is disconnected, we try to make it connected. We first identify the set V of partitions of the solution. While it is possible, we try to take a node from a partition s and put it into a route of another partition s^s, such that $s \neq s^s$. If this movement is feasible, partitions s and s^s correspond now to just one partition, allowing us to eliminate partition s^s from set V. If we cannot connect two partitions by putting a node from one partition into a route of another partition, we choose the closest two nodes between the two partitions and we construct a new route connecting these two nodes.

3.5. Hub configuration and capacity-feasibility check

The destroy and repair operators as well as the procedure applied to verify if the solution is connected are the same for both TDALNS and BUALNS. However, the way we define the hub configuration of a solution and the mechanism to check whether a solution is feasible with respect to the vehicle capacities differ from one version to the other. We next describe how each of the algorithms perform these two steps.

3.5.1. TDALNS Algorithm

In the TDALNS algorithm, the initial hub configuration is obtained by solving a p-median problem. Whenever ω^{ih} iterations have been performed, we modify the hub configuration before applying the destroy operator. We start by closing the hubs incident to exactly one route in the current solution. A hub configuration operator is next chosen to add one more hub to the current solution. We consider three hub configuration operators. Operators H^1 and H^3 choose the new hub according to the σ^p value.

- **Choose node with the highest incoming-outgoing flow** (H^1): Choose among the nodes with the highest incoming-outgoing flow.
- **Choose node randomly** (H^2): Randomly choose a node to become a hub.
- **Choose node according to geographic position** (H^3): Choose among the nodes that are closest to the central point of the graph.

We verify if the network is connected with respect to the hub configuration. If the hub configuration is not feasible, the solution is discarded. Otherwise, we check if the current network has enough capacity to serve all the demand flows. To do so, we solve a linear problem that we denote as the *capacity verification problem* (CVP). Let $\mathcal{P}^{ij}$ be the set containing all feasible paths to serve demand pair $(i,j) \in W$ over the solution network defined by the arcs in the active vehicle routes, i.e., $A(s) = \{(k,m) \in A : \bar{x}^p_{km} = 1\}$, where $x^p_{km} = 1$ indicates that vehicle p passes through arc (k,m) on the current solution s. Let parameter $b^l_{kmp} \in \{0,1\}$ be equal to one if path l contains vehicle $p \in P$ passing through arc $(k,m) \in A$. We also define variables $e^p_{km} \geq 0$ to represent the overflow on arc $(k,m) \in A$ for vehicle $p \in P$, and $f^{ij}_l \geq 0$ to be the percentage of demand w_{ij} that goes via path $l \in \mathcal{P}^{ij}$. The CVP can then be written as:

$$\min \sum_{p \in P} \sum_{\substack{(k,m) \in A: \\ \bar{x}^p_{km}=1}} e^p_{km} \tag{32}$$

$$\text{s.t.:} \sum_{l \in \mathcal{P}^{ij}} f^{ij}_l = 1 \qquad \forall (i,j) \in W \tag{33}$$

$$\sum_{(i,j) \in W} \sum_{\substack{l \in \mathcal{P}^{ij}: \\ b^l_{kmp}=1}} w_{ij} f^{ij}_l - e^p_{km} \leq \tau^{\phi(p)} \bar{x}^p_{km} \qquad \forall p \in P, (k,m) \in A : \bar{x}^p_{km} = 1 \tag{34}$$

$$f^{ij}_l \geq 0 \qquad \forall (i,j) \in W, l \in \mathcal{P}^{ij} \tag{35}$$

$$e^p_{km} \geq 0 \qquad \forall (k,m) \in A, p \in P : \bar{x}^p_{km} = 1, \tag{36}$$

where $\bar{\mathbf{x}}$ represents the current solution that forms the connected network. The objective function minimizes the total overflow of the network (32). Constraints (33) guarantee that all the demand pairs will be served by a path, while constraints (34) calculate the overflow on each arc.

The current solution can either have a feasible network or an infeasible one due to either a lack of capacity or an insufficient number of opened hubs. To solve the CVP, instead of enumerating all paths, we generate them on the fly. First, an auxiliary graph $\tilde{G} = (N, \tilde{A}(s))$ is created with the active arcs and vehicles of the current solution s. For each pair $(i,j) \in W$, we initialize $\mathcal{P}^{ij}$ with its shortest path in the current network. Then, the CVP is solved.

If the CVP objective function is equal to zero, then the procedure terminates. If transshipment occurs only at hub nodes, solution s is feasible. Otherwise, paths already added to the $\mathcal{P}^{ij}$ set are not enough to serve all the demand pairs. However, as set $\mathcal{P}$ does

not necessarily contain all possible paths to serve the demand, we cannot yet assure that the solution is infeasible. Thus, we verify if there is any new path to be considered when solving the CVP in order to prevent overflows from occurring. Let W' be the set of demand pairs $(i, j) \in W$, that need to pass through any arc with overflow to be served. So, after constructing set W' and updating the auxiliary graph $\tilde{G}$, forbidding arcs with overflow, we look for new paths. On one hand, if we cannot find any new path for demand pairs with overflowing arcs, the solution is considered infeasible and penalized according to the overflown quantity found. On the other hand, if the value of the objective function (32) is zero and if transshipment occurs only at hub nodes, the solution is feasible. Otherwise, if the value of the objective function (32) is zero while any transshipment occurs at non-hub nodes, then the hub configuration is considered infeasible and the solution is discarded.

Procedure FindPath is used to identify the shortest path from node i to node j in the current auxiliary graph $\tilde{G}$. We use Dijkstra's algorithm, in which the cost of the arcs corresponds to their length. If in the path, it is necessary to change vehicles in a node m, and if node m is not an active hub, we add to the shortest path cost the cost of activating node m as a hub.

Our approach shares similarities with column generation algorithms because we create paths on demand. The CVP corresponds to the Master Problem. The pricing subproblem is equivalent to identifying new paths for pairs $(i, j) \in W$. The CVP is solved heuristically because instead of using the dual variables to price new columns, we simply forbid arcs with overflow. We solve the CVP until the value of the objective function is zero or until it is not possible to add more paths to the problem according to our criteria. Preliminary computational experiments showed this was the most efficient way to check feasibility.

3.5.2. BUALNS Algorithm

In the BUALNS algorithm, after ensuring a connected network, we first verify if the current solution has enough arc capacity and if it is the case, the hub network is defined. The procedure to check arc capacity feasibility consists of solving the CVP, as described before for the TDALNS algorithm. The nodes in which transshipment occurs are stored in set H. If the solution is found to be infeasible, it is penalized according to the overflown quantity found and can be accepted or not according to a simulated annealing criterion. Otherwise, we need to define a feasible hub network. To define the cheapest

hub configuration, we solve the Hub Activation Problem (HAP) stated as follows:

$$\min \sum_{k \in H} h_k y_k \tag{37}$$

$$\sum_{l \in \bar{\mathcal{P}}^{ij}} f_l^{ij} = 1 \qquad \forall (i,j) \in W \tag{38}$$

$$\sum_{(i,j) \in W} \sum_{\substack{l \in \bar{\mathcal{P}}^{ij} \\ b_{kmp}^l}} w_{ij} f_l^{ij} \leq \tau^{\phi(p)} \bar{x}_{km}^p \qquad \forall p \in P, (k,m) \in A \tag{39}$$

$$y_k \geq f_l^{ij} \qquad \forall k \in H, (i,j) \in W, l \in \bar{\mathcal{P}}_k^{ij} \tag{40}$$

$$f_l^{ij} \geq 0 \qquad \forall (i,j) \in W, l \in \bar{\mathcal{P}^{ij}} \tag{41}$$

$$y_k \in \{0,1\} \qquad \forall k \in H. \tag{42}$$

Let $\bar{\mathcal{P}}^{ij}$ be the set of paths for demand pair $(i,j) \in W$ that were added in the CVP phase, and $\bar{\mathcal{P}}_k^{ij}$ be the set of paths for demand pair $(i,j) \in W$ containing node k, such as node k needs to become a hub so that the path becomes feasible. In the HAP, the objective function minimizes the cost of activating hubs. Constraints (38) guarantee that all demand pairs are served, while constraints (39) make sure that the allocated capacity is respected. Constraints (40) ensure that flow can pass through a path that needs a node to become a hub just if this node is activated as a hub. Constraints (41) and (42) show the variable domains. It is important to note that if $|H| = 1$, we do not have to solve the HAP, because in that case there is no decision to be made, we just activate the only hub that is in set H.

3.6. Selection of operators

Let π_i be the weight associated with operator i. The probability of choosing operator i is given by $\pi_i / \sum_j \pi_j$. The update of the weight of each operator is done according to the performance of each operator in a time interval (segment) by means of scores. The score of each operator is initially set to 10. At each iteration, each score can be increased by a factor as follows: (a) θ^1 if the operator results in a new best solution, (b) θ^2 if the operator results in a new solution worst than the minimum, but better than the current one, and (c) θ^3 if the operator results in a solution that is worse than the current one, but satisfies the acceptance criterion. At the end of each segment we calculate new weights for each operator, as follows:

$$\pi_i = (1-\gamma)\pi_i + \gamma \frac{\eta_i}{\zeta_i},$$

where η_i is the score of operator i obtained during the last segment, ζ_i is the number of times operator i was used in the last segment, and γ is the reaction factor that controls how quickly the weight adjustment algorithm reacts to the last segment performance.

3.7. Penalizing infeasible solutions

When we detect that the new solution does not contain enough capacity to serve all demand pairs, we add to the objective function the quantity $\alpha_{km} \sum_{p \in P} \sum_{(k,m) \in A} e^p_{km}$ which carries the penalization factor α_{km}. This factor is initially set to $10 \sum_{r \in \mathcal{R}} \max_{(k,m) \in A} c^r_{km}$. At every 25 iterations, we change this penalization factor. If the best solution s^* is feasible, the penalization factor is decreased to $\alpha_{km} = 0.90\alpha_{km}$. Otherwise, it is increased to $\alpha_{km} = 1.10\alpha_{km}$.

3.8. Acceptance criterion

To accept a new solution, we adopt a simulated annealing criterion. An improving solution is always accepted whereas a worse solution is accepted with probability $e\hat{}(\phi(s') - \phi(s))/t)$, where $\phi(s)$ returns the objective function value of solution s. The temperature t is calculated by the formula $t_i = ct_{i-1}$ at iteration i. We set the initial temperature as a percentage of the objective function of the initial solution, such as $t_0 = \phi(s^0)c^0$. We determine c according to Ribeiro et al. (2014). We suppose a final temperature as a percentage of the objective function of the initial solution, $t_f = \phi(s^0)c^f$. Let Ω be the maximum number of iterations performed, then c is calculated as $c = \sqrt[\Omega]{\frac{t_f}{t_0}}$.

4. Computational experiments

We next present the results of the computational experiments performed to assess the behavior of our exact and heuristic algorithms. In all experiments, we set a maximum time limit of 24 hours of CPU time (86,400 seconds). All experiments were performed on an Intel(R) Xeon(R) CPU E5-2687W v3 @ 3.10GHz computer with 750 GB of memory running on a Linux environment. The formulations were coded in C++ using the Concert Technology of CPLEX 12.9 to solve them. Up to four threads were allowed to run during the executions. After some preliminary tests, to improve performance, we set the following variable priority order for branching: y, q, z and x. We chose the barrier optimizer to solve the nodes of the branch-and-bound tree.

The separation and addition of SECs were implemented via lazy cut callbacks and user cut callbacks. To detect the violated SECs for a fractional or integer solution, for each vehicle $p \in P$, such that $q^p > 0$, we solved a series of minimum s-t cut problems

in a support graph $G' = (N^{0'}, A^{0'})$ in which $N^{0'} = \{i \in N^0 | z_i^p > 0\}$ and $A^{0'} = \{(i,j) \in A^0 | x_{ij}^p > 0\}$. We set the fictitious depot as the source node s and the nodes $i \in N^{0'}$ as the sink node t. A violated SEC is identified every time the value of the resulting minimum cut is less than 2 units. We used the Concorde callable library by **?** to solve the associated mincut problems.

The metaheuristics were also coded in C++. We use the irace package (López-Ibánez et al., 2011), an automatic configuration tool, to fine tune the values of the parameters used in both algorithms. We used the default package setting using a maximum number of 58320 experiments during the tuning. We decided to run the irace Package on 16 different instances. For the TDALNS algorithm, we change the hub configuration every 50 iterations, i.e. $\omega^{ih} = 50$. The number of nodes n^d and routes r^d to be destroyed are picked within the range $[1, \lambda \times n]$ and $[1, \lambda \times R]$, respectively, where the parameter R represents the total number of routes in the current solution. The parameter λ starts with value 0.1 and is increased by 0.1 every Ψ iterations. For the TDALNS and BUALNS algorithms, we set Ψ to 50 and 5, respectively. When λ reaches the maximum value Υ, we set it to 0.1 again. For the TDALNS and BUALNS algorithms, we use $\Upsilon = 0.6$ and $\Upsilon = 0.7$, respectively. The weights of the operators are reset every 50 iterations to $\omega^{is} = 50$. For the TDALNS algorithm, we use $(\theta^1, \theta^2, \theta^3) = (0, 10, 5)$, while, for the BUALNS variant, we use $(\theta^1, \theta^2, \theta^3) = (10, 5, 0)$. The reaction factor γ is set to 0.1. For the acceptance criterion, we consider $c^0 = 1$ and $c^f = 0.01$. The heuristics were applied 10 times to each instance with each execution taking 25,000 iterations.

4.1. Test instances

We used two sets of instances for our experiments. The first data set was randomly generated, whereas the second one was extracted from the Australian Post (AP) standard data set (Ernst and Krishnamoorthy, 1996). Both data sets assume $T = 24$ and two types of vehicles which were set to have the same speed of 1 unit per hour.

For the first data set, the nodes were randomly scattered around clusters. To guarantee the feasibility of the instance, we consider that the central point of each cluster is at most 12 units away from each other central point. We considered a 16×16 square region with up to four possible clusters. The x and y coordinates of the central points of clusters 1, 2, 3 and 4 are $(4, 4)$, $(12, 12)$, $(12, 4)$ and $(4, 12)$, respectively.

We generated a set of instances by varying the number of clusters, and the total number of nodes in the network. For instance, if clusters 2 and 3 were selected, and an instance with 10 nodes is being created, then the other 8 nodes would be randomly

generated. To scatter the non-central cluster nodes, we first randomly assign them to a central node, and then we randomly generate their x and y coordinates such that they will lay within three units of distance to a central node of the cluster. The demands between the central points of the clusters were randomly chosen as $\lceil U(0.10, 0.30) \times 750 \rfloor$, while between the other nodes they were randomly selected as $\lceil U(0.01, 0.10) \times 100 \rfloor$, where the operator $\lceil \cdot \rfloor$ returns the nearest integer.

We assumed that all nodes are hub candidates, while the fixed costs to install a hub in a node were randomly chosen within the range $[20000, 30000]$. In our experiments, we selected problems with $|N| = \{6, 7, 8, 9, 10, 20, 30, 40, 50\}$ nodes. These random instances are referred to as NR, where N represents the number of nodes considered in the instance. For example, instance 10R contains 10 nodes. The vehicle costs were set to $b^I = 3000, b^{II} = 4500$, $a^I = 20000$, $a^{II} = 35000$ for each instance. For each problem size, we generated 4 instances for the HNDP varying the value of τ^I to assess the effect of different economies of scale. Table 2 shows the capacity values considered. The potential economies of scale obtained when using a larger vehicle when it is fully utilized is calculated as $((a^{II} + b^{II})/\tau^{II})/((a^I + b^I)/\tau^I)$.

Table 2: Capacity of the vehicles and potential economies of scale for the random instances

Instances	economies of scale	τ^I	τ^{II}
6R-10R	0.21	90	750
	0.34	150	
	0.50	220	
	0.80	350	
20R	0.21	250	2040
	0.34	400	
	0.50	590	
	0.80	1070	
30R	0.21	290	2360
	0.34	470	
	0.50	690	
	0.80	1100	
40R	0.21	370	3000
	0.34	600	
	0.50	880	
	0.80	1400	
50R	0.21	510	4160
	0.34	830	
	0.50	1210	
	0.80	1940	

The AP data set consists of demand flows and Euclidean distances between 200 districts of Australia. We selected problems with $|N| = \{10, 15, 20, 25, 30, 35, 40, 45, 50\}$ nodes, but disregarding flows w_{ii} for each $i \in N$, i.e. we set $w_{ii} = 0$. We changed the distances in a way that from each node it is possible to go to at least half of the total nodes of the graph within 12 hours. The vehicle costs were set to $b^I = 6000, b^{II} = 9000$, $a^I = 20000$, $a^{II} = 30000$ and $\tau^{II} = 3000$ for each instance. For each problem size, we generated four instances for the HNDP by varying the value of τ^I to assess the effect of different economies of scale. Table 3 shows the capacity values considered for the small vehicle and the potential economies of scale obtained.

Table 3: Capacity of the vehicles and potential economies of scale for the AP instances ($\tau^{II} = 3,000$)

Economies of scale	τ^I
0.20	400
0.30	600
0.50	1,000
0.80	1,400

4.2. Solving small instances

We first focus on analyzing the performance of the proposed MIP formulation and algorithms to solve small instances. Table 4 reports the results obtained for instances 6R, 7R, 8R, 9R, 10R, and 10AP considering economies of scale of 0.21/0.20, 0.34/0.30, 0.50 and 0.80. We consider four vehicles of type I and two vehicles of type II. Column Dev(%) presents the deviation between the optimal solution and the solutions found by the MIP formulation, TDALNS and BUALNS. In particular, columns under the heading AVG(BKS) report the deviation between the optimal solution and the average(best) solution found by TDALNS and BUALNS. Column TRN (s) presents the time spent to solve the linear programming relaxation of the root node of the branch-and-bound tree. Column LP(%) reports the deviation between the optimal solution and the linear programming relaxation value.

As can be seen in Table 4, the devised metaheuristics take much less time than the exact algorithm to solve the instances, as expected. The MIPFV version requires 804 and 434 times more computational effort on average than the TDALNS and the BUALNS versions, respectively. For instance 9R-0.80, the computational effort needed to solve just the linear relaxation of the formulation is about eight and four times larger than the time spent by the TDALNS and BUALNS algorithms, respectively. The MIPFV formulation was not able to reach optimality for two instances within the given time limit, 10R-0.50,

Table 4: Results for small instances: 6R, 7R, 8R, 9R, 10R and 10AP.

Economies of scale	Instances	Time(s)			Dev(%)					MIPFV	
		MIP	TDALNS	BUALNS	MIP	TDALNS		BUALNS		TRN (s)	LP (%)
						BKS	AVG	BKS	AVG		
0.21/0.20	6R	34	14	27	0.00	0.00	0.00	0.00	0.00	0	19.84%
	7R	71	15	25	0.00	0.00	0.00	0.00	0.00	1	20.36%
	8R	86	17	31	0.00	0.00	0.00	0.00	0.00	2	18.73%
	9R	264	22	33	0.00	0.00	0.00	0.00	0.00	4	16.90%
	10R	17454	26	55	0.00	0.00	0.00	0.00	0.00	11	24.07%
	10AP	84177	28	50	0.00	4.77	5.72	4.74	4.75	16	29.12%
Average		17014	20	37	0.00	0.80	0.95	0.79	0.79	6	21.50%
0.34/0.30	6R	76	13	26	0.00	0.00	0.00	0.00	0.00	0	32.91%
	7R	982	14	28	0.00	0.00	0.00	0.00	0.00	1	33.07%
	8R	317	17	32	0.00	0.00	0.00	0.00	0.00	2	18.73%
	9R	1691	22	36	0.00	0.00	0.00	0.00	0.00	6	17.20%
	10R	27077	26	57	0.00	0.00	0.00	0.00	0.00	11	23.88%
	10AP	30561	28	54	0.00	0.00	0.00	0.00	0.00	19	18.58%
Average		10117	20	39	0.00	0.00	0.00	0.00	0.00	7	24.06%
0.5	6R	43	17	30	0.00	0.00	4.25	0.00	1.58	0	36.63%
	7R	204	22	29	0.00	1.38	1.56	1.38	1.38	1	36.03%
	8R	2394	24	37	0.00	0.00	0.00	0.00	0.00	2	30.78%
	9R	32403	24	36	0.00	0.00	0.00	0.00	0.00	5	32.96%
	10R	86400	26	57	5.20	8.67	9.33	0.00	8.45	13	29.84%
	10AP	86400	26	60	16.03	0.83	0.83	0.83	0.83	20	33.59%
Average		34641	23	42	3.54	1.81	2.66	0.37	2.04	7	33.31%
0.8	6R	20	17	30	0.00	0.00	0.00	0.00	0.00	0	39.27%
	7R	17	22	30	0.00	0.00	0.00	0.00	0.00	1	23.93%
	8R	86	25	37	0.00	0.00	0.00	0.00	0.00	3	20.93%
	9R	431	23	44	0.00	0.00	0.00	0.00	0.00	182	20.22%
	10R	10068	26	54	0.00	0.00	1.23	0.00	0.58	12	29.94%
	10AP	37758	27	67	0.00	0.00	0.00	0.00	0.00	19	33.35%
Average		8063	23	44	0.00	0.00	0.21	0.00	0.10	36	27.94%

and 10AP-0.50, due to the complexity of the problem. However, for instance 10R-0.50, the BUALNS was able to find the optimal solution within less than one minute. The TDALNS and BUALNS failed to find the optimal solution in four (10AP-0.20, 7R-0.50, 10R-0.50, and 10AP-0.50) and three (10AP-0.20, 7R-0.50, and 10AP-0.50) instances, respectively, of the 24 proposed test problems.

Table 5 shows the results obtained for instances 20R, 15AP and 20AP. We considered five vehicles for each type I and II. We took into account two additional different performance measures: Dev_BKS is the percentage deviation between the best solution found by the version and the best known solution; while GAP_CPLEX is the optimality gap found by CPLEX within 24 hours of CPU time.

Table 5: Results for medium instances: 20R, 15AP and 20AP

Economies of scale	Instances	Time(s)			Desv_BKS(%)			MIPFV	
		MIPFV	TDALNS	BUALNS	MIPFV	TDALNS	BUALNS	TRN (s)	GAP_CPLEX(%)
0.21/0.20	20R	86400	143	329	0.00	0.00	0.69	52199	32.74%
	15AP	86400	62	156	0.00	0.00	0.00	2627	29.05%
	20AP	86400	124	314	0.00	0.26	0.00	54102	26.01%
0.34/0.30	20R	86400	141	332	0.00	2.08	0.00	86388	100.00%
	15AP	86400	63	159	0.73	0.73	2.17	2038	27.76%
	20AP	86400	122	325	0.00	0.40	0.00	57579	30.01%
0.5	20R	86400	140	337	0.00	0.00	1.84	86385	100.00%
	15AP	86400	69	176	3.50	5.78	3.50	1626	33.03%
	20AP	86400	149	391	0.00	0.00	0.00	38073	31.35%
0.8	20R	86400	127	353	0.00	0.00	3.22	83842	27.21%
	15AP	86400	61	181	0.00	0.00	0.00	1116	30.64%
	20AP	86400	121	378	0.00	4.44	0.00	29823	26.64%
Average		86400	110	286	0.35	1.14	0.95	41317	41.20%

When the instance size is greater than 15 nodes, it becomes much more difficult to solve by CPLEX regardless of the formulation being used. Before calling the CPLEX solver, the best solution found by the devised metaheuristics is supplied as an initial solution to it. The fact that the metaheuristcs performed better than MIPFV stands out once again. While the metaheuristics found good solutions within 400 seconds, the MIPFV took 41316 seconds on average just to attain the linear programming relaxation of the formulation for these instances. For some instances (20R-0.30 and 20R-0.50), the MIPFV was not able to find any lower bound within one day of computing time. These results highlight how difficult our problem is.

4.3. Solving large instances

To better assess the devised metaheuristics, we also tested them on larger instances. Since the complexity of the problem does not allow the formulation to neither prove optimality nor improve the best solution found by the proposed metaheuristics for instances with 15 nodes, we chose to solve larger instances with more than 25 nodes using only the metaheuristics. We considered five vehicles for each type I and II. Table 6 reports the average computational time in seconds, the deviation between the best solution found by the version and the best known solution (Dev_BKS), and the deviation between the average objective function value found and the best known solution (Dev_avg).

According to Table 6, the TDALNS version gets worse solutions than BUALNS though taking less time. Table 5 shows that BUALNS version provides slightly better solutions on average. For instances with 15 and 20 nodes, the BUALNS version found the best known solution seven times, whereas the TDALNS variant got the best known solution

Table 6: Results for large instances: 30R, 40R, 50R, 25AP, 30AP, 35AP, 40AP, 45AP and 50AP.

Economies of scale	Instances	Time(s)		Dev_BKS(%)		Dev_Avg(%)	
		TDALNS	BUALNS	TDALNS	BUALNS	TDALNS	BUALNS
0.20	30R	465	1246	0.00	0.41	4.25	1.14
	40R	1257	3084	3.50	0.00	10.14	4.40
	50R	2886	7958	4.95	0.00	10.78	5.30
	25AP	217	598	2.62	0.00	7.04	2.35
	30AP	383	1130	2.08	0.00	5.12	2.24
	35AP	646	1918	0.00	0.59	3.70	4.04
	40AP	1074	3326	8.07	0.00	13.24	7.50
	45AP	1607	5384	0.00	7.20	10.78	10.31
	50AP	2517	8595	4.48	0.00	9.67	5.75
Average		1228	3693	2.86	0.91	8.30	4.78
0.30	30R	464	1256	0.65	0.00	3.35	1.30
	40R	1267	3454	6.69	0.00	12.27	5.70
	50R	2826	8153	4.92	0.00	8.46	6.38
	25AP	221	623	0.00	1.42	8.15	7.68
	30AP	387	1140	5.26	0.00	8.85	4.64
	35AP	643	2014	0.00	1.88	3.87	5.12
	40AP	1077	3488	0.00	7.21	14.97	14.15
	45AP	1621	5668	4.05	0.00	8.04	6.20
	50AP	2581	9021	0.00	3.01	10.81	8.52
Average		1232	3869	2.40	1.50	8.75	6.63
0.50	30R	443	1242	0.69	0.00	4.29	2.55
	40R	1187	3293	0.00	1.98	5.83	6.67
	50R	2818	8460	0.39	0.00	5.61	6.91
	25AP	217	756	0.00	2.54	8.45	8.48
	30AP	389	1208	1.12	0.00	9.23	6.70
	35AP	640	2134	5.85	0.00	12.02	6.17
	40AP	1164	3748	10.63	0.00	13.82	6.48
	45AP	1621	6121	0.00	1.81	8.29	4.81
	50AP	2530	9752	0.50	0.00	8.49	6.22
Average		1223	4079	2.13	0.70	8.45	6.11
0.80	30R	429	1242	2.30	0.00	4.27	5.73
	40R	1214	3561	0.00	6.96	6.81	12.15
	50R	2887	9124	0.00	8.61	6.87	14.34
	25AP	218	746	12.79	0.00	18.15	5.70
	30AP	379	1430	2.46	0.00	11.94	4.82
	35AP	629	2513	10.55	0.00	15.36	6.20
	40AP	1042	4600	0.00	2.46	10.01	8.81
	45AP	1597	7269	0.98	0.00	12.84	6.23
	50AP	2483	11621	7.15	0.00	19.21	8.39
Average		971	3317	4.01	2.57	10.48	8.25

six times. For the large instances, although the TDALNS is in average three times faster than the BUALNS version, the BUALNS is in average two times better to find the best known solutions, and 1.4 times better to find averages values. To obtain these values we used the ratio of the summed averages.

Despite taking more computing time, the BUALNS seems to present a better performance. The larger computational effort is due to the step in which the CPLEX solver is called in each iteration to find the cheapest hub configuration. In the TDALNS the hub configuration is randomly defined. In order to confirm this behavior, we performed a statistical test. Generally, there are two types of statistical tests, namely parametric and non-parametric. Three conditions must be met to use a parametric test, including normality, independence, and heteroscedasticity. A Kolmogorov-Smirnov test was carried out to check the first condition (i.e. normality). The results reported in Table 7 show that the normality condition is not fulfilled in our case (i.e., the p-value is less than .05).

For this reason, a non-parametric test, the Wilcoxon test, was used to compare the devised algorithms in terms of best known solutions found. To calculate the Wilcoxon signed-rank test, the Wilcoxon function of the scipy Python package was used. First, we used the default alternative hypothesis to test if the algorithms were statistically different and later, we use the alternative hypothesis to verify if the BUALNS was better than the TDALNS version. The p-values of the Wilcoxon tests are reported in Table 8. The results show that there is a significant difference between the algorithms, and that the BUALNS is indeed better than TDALNS to find BKSs.

Table 7: Results of the normality analysis.

Algorithm	Kolmogorov-Smirnov test	
	Statistic	P-value
TDALNS	0.445	0.000
BUALNS	0.235	0.001

Table 8: Results of the Wilcoxon tests.

Pairwise comparison	Alternative	Statistic	P-value
TDALNS-BUALNS	default	233.0	0.0
TDALNS-BUALNS	greater	1198.0	0.0

To draw further insights from the algorithms, their performance to find a target solution is studied. The worst average objective function value found for two selected instances (25AP-030, 25AP-080) is given as the target solution. Each method is run for 50 executions, using different seeds each time, until the target solution is found. The computational time required to find it is recorded. The results are then displayed using time-to-target (TTT) plots introduced by Aiex et al. (2002, 2007). For a given instance, a TTT plot displays on the ordinate axis the probability of the algorithm to obtain a solution as good as a given target value within a running time in seconds, shown on

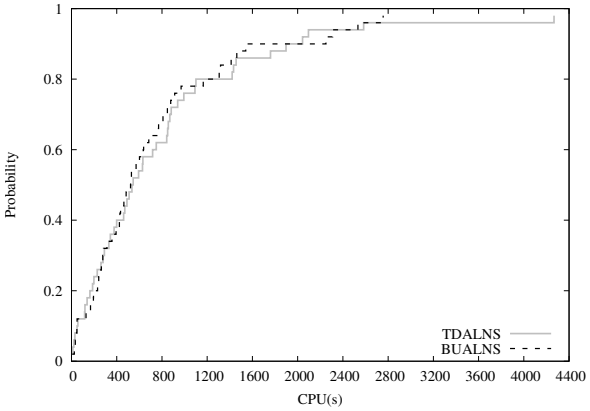

Figure 2: TTT plot for 25AP-030

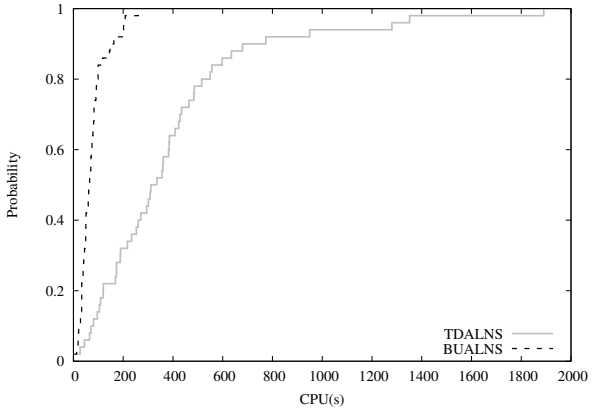

Figure 3: TTT plot for 25AP-080

the abscissa axis. Figures 2 and 3 show the attained TTT plots for the two instances. For instance 25AP-030, it can be seen that both versions have very similar behavior. The BUALNS finds the target solution 100% of the time within 2800 seconds, while the TDALNS has around 2% of probability to spend more seconds than that. On the other hand, for instance 25AP-080, the BUALNS has a much higher probability to find better solutions than the TDALNS in a shorter time. It took 10% of the time to assert the target solution.

Finally, the benchmark performance profiles of the algorithms were examined. Performance profiles, shown in Figure 4, are graphic tools which aid to evaluate and compare the performance of a set of algorithms on solving a problem set. They show the plot of a performance function over a chosen performance measure. Further details of performance profiles can be found in Dolan and Moré (2002). Here, the number of times the best known solutio (BKS) is found by an algorithm is used as a performance measure. Hence the performance function $\rho(\tau)$ represents the proportion of instances solved by an algorithm, with performance within a factor τ of the best behavior obtained, considering all the analyzed algorithms. The TDALNS finds the BKS for only 40% of the instances whereas the BUALNS finds the BKS for 80% of the instances.

After these analyses we can point out that, although the hub configuration is a more strategic decision than defining the vehicle routes, and that strategic decisions are usually taken before tactical decisions, the results here indicate that locating the hubs after knowing the vehicle routes is a better practice.

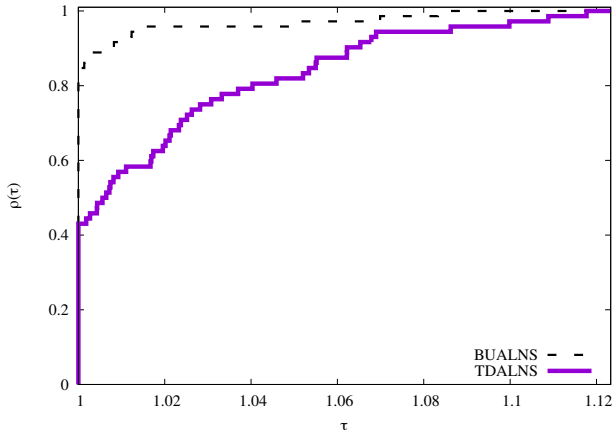

Figure 4: Performance profile of the devised algorithms.

5. Conclusion

In this paper, we have studied a flexible hub network design problem. The flexibility stems from the fact that no fixed discount factors representing economies of scale are imposed. Here, economies of scale directly derive from the transport technology chosen to operate the routes which ensures that vehicles can be used more efficiently. Moreover, no particular topological structure is assumed, which means that: *(i)* vehicle routes do not necessarily have hubs on them; *(ii)* there is no limitation on the number of vehicles that passes through each demand node, and; *(iii)* the demand can be served directly or passing through as many non-hub or hub nodes as it is cost convenient. These aspects make the problem more complex to formulate and solve. The HNDP is specifically suitable for liner shipping network design, in which a hub network design with flexible periodic routes must satisfy demands for shipments of containers at minimal cost. A mathematical formulation and two metaheuristics based on the adaptive large neighborhood search heuristic were proposed. On the one hand, the MIPFV takes considerable computational time just to solve the linear programming relaxation of the formulation for small instances, highlighting how difficult the problem is. On the other hand, the devised metaheuristics find good solutions within a reasonable time. The computational results showed that, for the considered instances, the bottom up approach found better solutions than the top down approach.

Acknowledgments

The authors are grateful to agencies CAPES, CNPq, and FAPES for financially supporting this research.

References

Aiex, R. M., Resende, M. G., and Ribeiro, C. C. (2002). Probability distribution of solution time in grasp: An experimental investigation. *Journal of Heuristics*, 8(3):343–373.

Aiex, R. M., Resende, M. G., and Ribeiro, C. C. (2007). Ttt plots: a perl program to create time-to-target plots. *Optimization Letters*, 1(4):355–366.

Alumur, S. and Kara, B. Y. (2008). Network hub location problems: The state of the art. *European Journal of Operational Research*, 190(1):1 – 21.

Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. (2006). *The traveling salesman problem: a computational study*. Princeton university press.

Camargo, R. S. d., Miranda, G. d., and Løkketangen, A. (2013). A new formulation and an exact approach for the many-to-many hub location-routing problem. *Applied Mathematical Modelling*, 37(12-13):7465–7480.

Campbell, J. F. and O'Kelly, M. E. (2012). Twenty-five years of hub location research. *Transportation Science*, 46(2):153–169.

Çetiner, S., Sepil, C., and Süral, H. (2010). Hubbing and routing in postal delivery systems. *Annals of Operations research*, 181(1):109–124.

Contreras, I. and O'Kelly, M. E. (2019). Hub location problems. In Laporte, G., Nickel, S., and Saldanha da Gama, F., editors, *Location Science*, pages 311–344. Springer International Publishing, Cham.

Contreras, I., Tanash, M., and Vidyarthi, N. (2017). Exact and heuristic approaches for the cycle hub location problem. *Annals of Operations Research*, 258(2):655–677.

de Camargo, R. S., de Miranda, G., and Luna, H. P. L. (2009). Benders decomposition for hub location problems with economies of scale. *Transportation Science*, 43(1):86–97.

de Camargo, R. S., de Miranda Jr, G., O'Kelly, M. E., and Campbell, J. F. (2017). Formulations and decomposition methods for the incomplete hub location network design problem with and without hop-constraints. *Applied Mathematical Modelling*, 51:274–301.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213.

Ernst, A. T. and Krishnamoorthy, M. (1996). Efficient algorithms for the uncapacitated single allocation p-hub median problem. *Location science*, 4(3):139–154.

Farahani, R. Z., Hekmatfar, M., Arabani, A. B., and Nikbakhsh, E. (2013). Hub location problems: A review of models, classification, solution techniques, and applications. *Computers & Industrial Engineering*, 64(4):1096 – 1109.

Gelareh, S. and Pisinger, D. (2011). Fleet deployment, network design and hub location of liner

shipping companies. *Transportation Research Part E: Logistics and Transportation Review*, 47(6):947–964.

Hamacher, H. W., Labbé, M., Nickel, S., and Sonneborn, T. (2004). Adapting polyhedral properties from facility to hub location problems. *Discrete Applied Mathematics*, 145(1):104–116.

Karimi, H. (2018). The capacitated hub covering location-routing problem for simultaneous pickup and delivery systems. *Computers & Industrial Engineering*, 116:47–58.

Kartal, Z., Hasgul, S., and Ernst, A. T. (2017). Single allocation p-hub median location and routing problem with simultaneous pick-up and delivery. *Transportation Research Part E: Logistics and Transportation Review*, 108:141–159.

Kartal, Z., Krishnamoorthy, M., and Ernst, A. T. (2019). Heuristic algorithms for the single allocation p-hub center problem with routing considerations. *OR Spectrum*, 41(1):99–145.

Kimms, A. (2006). Economies of scale in hub & spoke network design models: We have it all wrong. In *Perspectives on operations research*, pages 293–317. Springer.

Labbé, M. and Yaman, H. (2008). Solving the hub location problem in a star–star network. *Networks: An International Journal*, 51(1):19–33.

Lopes, M. C., de Andrade, C. E., de Queiroz, T. A., Resende, M. G., and Miyazawa, F. K. (2016). Heuristics for a hub location-routing problem. *Networks*, 68(1):54–90.

López-Ibánez, M., Dubois-Lacoste, J., Stützle, T., and Birattari, M. (2011). The irace package, iterated race for automatic algorithm configuration. Technical report, Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles .

Martins de Sá, E., Contreras, I., Cordeau, J.-F., Saraiva de Camargo, R., and de Miranda, G. (2015). The hub line location problem. *Transportation Science*, 49(3):500–518.

Martins de Sá, E., de Camargo, R. S., and de Miranda, G. (2013). An improved benders decomposition algorithm for the tree of hubs location problem. *European Journal of Operational Research*, 226(2):185–202.

Nagy, G. and Salhi, S. (1998). The many-to-many location-routing problem. *Top*, 6(2):261–275.

O'Kelly, M. E. (1986). The location of interacting hub facilities. *Transportation science*, 20(2):92–106.

O'Kelly, M. E. and Bryan, D. (1998). Hub location with flow economies of scale. *Transportation Research Part B: Methodological*, 32(8):605 – 616.

Reinhardt, L. B. and Pisinger, D. (2012). A branch and cut algorithm for the container shipping network design problem. *Flexible Services and Manufacturing Journal*, 24(3):349–374.

Ribeiro, G. M., Desaulniers, G., Desrosiers, J., Vidal, T., and Vieira, B. S. (2014). Efficient heuristics for the workover rig routing problem with a heterogeneous fleet and a finite horizon. *Journal of Heuristics*, 20(6):677–708.

Rodríguez-Martín, I., Salazar-González, J.-J., and Yaman, H. (2014). A branch-and-cut algorithm for the hub location and routing problem. *Computers & Operations Research*, 50:161–174.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472.

Shaw, P. (1997). A new local search algorithm providing high quality solutions to vehicle routing problems. *APES Group, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK*.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In Maher, M. and Puget, J.-F., editors, *Principles and Practice of Constraint Programming — CP98*, pages 417–431, Berlin, Heidelberg. Springer Berlin Heidelberg.

Skorin-Kapov, D., Skorin-Kapov, J., and O'Kelly, M. (1996). Tight linear programming relaxations of uncapacitated p-hub median problems. *European journal of operational research*, 94(3):582–593.

Song, D.-P. and Dong, J.-X. (2013). Long-haul liner service route design with ship deployment and empty container repositioning. *Transportation Research Part B: Methodological*, 55:188–211.

Tanash, M., Contreras, I., and Vidyarthi, N. (2017). An exact algorithm for the modular hub location problem with single assignments. *Computers & Operations Research*, 85:32–44.

Wasner, M. and Zäpfel, G. (2004). An integrated multi-depot hub-location vehicle routing model for network planning of parcel service. *International Journal of Production Economics*, 90(3):403–419.