HUBBI: Iterative network design for incomplete hub location problems

Weibin Dai[a,b], Jun Zhang[a,b], Xiaoqian Sun[a,b,*], Sebastian Wandelt[a,b]

[a]*National Key Laboratory of CNS/ATM, School of Electronic and Information Engineering, Beihang University, 100191, Beijing, China*
[b]*National Engineering Laboratory of Multi-Modal Transportation Big Data, 100191, Beijing, China*

Abstract

In this paper, we design and implement HUBBI, a heuristic-based method to minimize the transportation/setup costs for *p*-hub location problems with incomplete hub networks. First, HUBBI pre-computes an estimation of node pair quality for establishing hub links. The quality ranking over all node pairs is then used to guide an iterative network design process: Given a solution for *p* hubs, two generic design patterns extend the hub network to *p+1* hubs. These networks are further refined by a variable neighborhood search method. In our evaluation on standard datasets (CAB, AP, TR, USA423 and URAND), HUBBI solves incomplete hub location problems with up to 200 nodes. For those instances that are solved with an exact method, HUBBI solves 90% to optimality and only one instance has a gap larger than 1%. HUBBI is also two to three orders of magnitudes faster than the exact method, while using two orders of magnitude less memory.

Keywords: Incomplete hub location problems, Hubbiness, Network design, Scalability

1. Introduction

Hub location problems have been studied for over three decades, since the seminal work by O'Kelly (1986). The problems have been applied in many fields successfully, including transportation (Alumur et al., 2012; O'Kelly, 2012; de Camargo et al., 2013) and telecommunication (Yaman and Carello, 2005; Kim and O'Kelly, 2009). In a network, hubs are installed to collect, transfer and distribute travel demands between pairs of nodes. When transporting large flows, economies of scale provide cost discounts for the transportation through inter-hub links. In early hub location problems, each spoke node is connected to exactly one hub, called single allocation (O'Kelly, 1987), or more than one hub, called multiple allocation (Campbell, 1994). The objective function takes into account the variable transportation cost, and in some cases the cost of building hubs. However, the cost of establishing links was not considered in early models. The hub networks were also assumed to be fully connected, i.e., hubs are connected with each other via a direct link. Although this assumption simplifies the problem, these models are often far away from realistic problem instances. Therefore, incomplete hub networks have been studied in recent years. In incomplete models, often only a few hub pairs are connected because of constraints on the number of hub links (Alumur et al., 2009) or setup/total costs (Contreras et al., 2010; O'Kelly et al., 2015; Campbell et al., 2015). Apart from few exceptions, hub location problems are NP-hard (Alumur and Kara, 2008; Campbell and O'Kelly, 2012; Sun et al., 2017). A number of solution techniques have been proposed, including Benders decomposition (de Camargo et al., 2008; de Camargo and Miranda, 2012), tabu search (Calık et al., 2009), evolutionary algorithm (Kratica et al., 2011), Lagrangian relaxation (Contreras et al., 2009a), local search (Rodríguez-Martín and Salazar-González, 2006), variable neighborhood search (Ilić et al., 2010; Todosijević et al., 2017), branch-and-price (Contreras et al., 2011), branch-and-cut (Rodríguez-Martín et al., 2014), clustering-based methods (Peker et al., 2016), strategic oscillation (Corberán et al., 2016), genetic algorithms (Kratica et al., 2005; Stanimirović, 2012; Azizi et al., 2016) and other heuristics (Randall, 2008; Hoff et al., 2017).

*Corresponding author.
Email addresses: daiweibin@buaa.edu.cn (Weibin Dai), buaazhangjun@vip.sina.com (Jun Zhang), sunxq@buaa.edu.cn (Xiaoqian Sun), wandelt@buaa.edu.cn (Sebastian Wandelt)

The hub location model targeted in our study is a recent extension of the multiple allocation incomplete hub networks modeled by de Camargo et al. (2017) with exactly p hubs (the p-hub constraint) but without hop constraint, i.e., limits on the number of arcs in a path. Fixed costs are associated with establishing all types of links and variable costs are considered for satisfying travel demands. Moreover, direct links can be established between spoke nodes so that the travel demands between selected node pairs do not need to go through hubs. de Camargo et al. (2017) developed a Benders decomposition with Benders feasibility cuts. However, this algorithm does not scale up well to large networks: Our own re-implementation needs around 230 hours to solve the problem with 80 nodes and its run time increases with n^6, where n is the number of nodes. Given the large search space and complicated interactions between variable and fixed costs, it is rather difficult to design efficient heuristics for this problem.

In this paper, we propose and implement HUBBI, a heuristic-based method which pre-computes rankings of possible hub pairs in the network, the so-called hubbiness, and then uses certain network design patterns to generate good solutions for many incomplete hub location problems with short computation time. For a given network design problem with a fixed number of hubs p, we first compute the hubbiness for each pair of nodes, i.e., the reciprocal of approximate total cost in the case that only two hubs are established on this pair of nodes. The node pair with the highest hubbiness is used as the initial design. Then, we iteratively design a hub network by adding hubs one at a time until the network reaches p hubs. Then, the solutions obtained by this procedure are used as initial solutions for VNS and improved further by exploring their neighborhood. In our evaluation on real-world networks and random networks (CAB, AP, TR, USA423 and URAND) of varying sizes (25–200 nodes), we can solve over 90% of the cases to optimality and provide solution gaps less than 1% for all instances except for one case. The run times and memory usage of our method are much less than that for the the enhanced Benders decomposition method. Thus, our work contributes towards efficiently solving large and realistic hub location problems with acceptable gaps. The major contributions of this paper are:

1. We propose and implement a heuristic-based method, called HUBBI, which provides high-quality solutions for incomplete hub location problems using a set of network design rules.
2. We discuss the factors that affect the topology of hub networks.
3. We solve incomplete hub location problems with up to 200 nodes.

The remainder of this paper is organized as follows. We provide a literature review on hub location problems in Section 2. The mathematical formulation of the problem is presented in Section 3. The methodology of HUBBI is proposed in Section 4. Experiments on five representative datasets (CAB, AP, TR, USA423 and URAND) are reported in Section 5. The paper concludes with Section 6.

2. Literature Review

In this section, we review literature related to our study. O'Kelly (1986) introduced hub location problems and the first mathematical formulation for the p-hub median problem was proposed by O'Kelly (1987). Campbell (1994) defined four fundamental hub location problems (p-hub median, uncapacitated hub location, p-hub center and hub covering) and proposed the corresponding formulations with $O(n^4)$ variables. Based on existing models, a new single allocation p-hub median problem with $O(n^3)$ variables was formulated by Ernst and Krishnamoorthy (1996). They used a simulated annealing algorithm together with a novel branch-and-bound algorithm to solve the problem.

Early research on hub location problems generally assumed that hub networks are completely connected, which makes the models too simplistic to be used in many real-world applications. Therefore, incomplete models were proposed in recent years. Alumur et al. (2009) proposed several models for four types of single allocation hub location problems (p-hub median, hub location with fixed costs, hub covering and p-hub center) with incomplete hub networks. The connections of hub links are constrained by a given number q, i.e., there must be q connected hub links in all solutions. A heuristic based on tabu search was proposed to solve the hub covering problem (Calık et al., 2009). The connections of hub links can also be constrained by the fixed cost for establishing hub links. Gelareh and Nickel (2011) proposed a 4-index formulation for the incomplete multiple allocation problem for public transport and liner shipping. A primal Benders decomposition method and a greedy heuristic were used to solve instances with up to 50 nodes. The optimal solutions were obtained with the Benders decomposition method, but the greedy heuristic provided solutions with gaps of up to 8.73%. O'Kelly et al. (2015) analyzed the role of fixed cost in the design of hub networks.

In addition, a 3-index formulation for the incomplete problem was proposed. Note that with some modifications, the 3-index formulation can be deduced by the 4-index model. Based on the previous models, de Camargo et al. (2017) proposed new models for incomplete hub location problems with and without hop-constraints. In their model with hop-constraint, for a given number S, at most S links can be used for satisfying the travel demand for each OD pair. Recently, incomplete hub location problems with robustness have been studied. A variant of the incomplete hub location model with uncertainties on travel demands and fixed setup cost, but without direct/access link cost has been solved by de Sá et al. (2018a), using Benders decomposition and a constructive heuristic method. They also proposed two specialized Benders decomposition algorithms to solve another robust incomplete hub location problem with uncertainties on service time requirements (de Sá et al., 2018b).

Since it is intrinsically difficult to solve large-scale hub location problems with exact algorithms, many heuristic algorithms have been proposed. These studies are summarized as follows. Skorin-Kapov and Skorin-Kapov (1994) proposed a heuristic method based on tabu search for the uncapacitated single allocation p-hub median problem. The CAB dataset with 25 nodes were solved efficiently with their algorithm. For the uncapacitated single allocation hub location problem without a p-hub constraint, Chen (2007) proposed two methods based on simulated annealing, tabu list and improvement procedures. The experimental results showed the good performance of their method compared with a genetic algorithm and a simulated annealing method. Abyazi-Sani and Ghanbari (2016) proposed an efficient tabu search with some new tabu rules for the same problem. The experimental results showed that their algorithm could find good solutions within shorter computation time compared to the tabu search proposed by Silva and Cunha (2009). Kratica et al. (2007) proposed two genetic algorithms to solve an uncapacitated single allocation p-hub median problem. Their methods were compared to tabu search (Skorin-Kapov and Skorin-Kapov, 1994), simulated annealing (Ernst and Krishnamoorthy, 1996) and path relinking (Pérez et al., 2004). Azizi et al. (2016) proposed a genetic algorithm for solving the hub location problem with the consideration of hub failure. Two approaches for selecting backup hubs were proposed. Experiments with up to 81 nodes showed good performance of their algorithm. Since proposed by Mladenović and Hansen (1997), variable neighborhood search has also been commonly used to solve hub location problems. Ilić et al. (2010) proposed a general variable neighborhood search to solve the uncapacitated single allocation p-hub median problem with up to 1000 nodes. Yet, the incomplete hub location problem targeted in this paper have not been solved with VNS so far, given difficulties to find good initial solution with large search space. Peiró et al. (2014) proposed GRASP for solving the uncapacitated r-allocation p-hub median problem. The algorithm consists of three local search procedures for the three phases of the problem: location, assignment and routing. Compared to previous methods, their heuristic can find high-quality solutions on large instances in short run times. For the same problem, Martí et al. (2015) proposed a scatter search method which consists of a diversification generator, a path-relinking method and two local search. The experimental results shows that their method outperforms the previous GRASP in datasets with up to 200 nodes. Todosijević et al. (2017) also proposed a new VNS for this problem. In order to explore more possible solutions, the full nested variable neighborhood descent was applied as a local search for the first time. The experimental results showed that their VNS outperforms the GRASP. Serper and Alumur (2016) proposed a single allocation model for the capacitated intermodal hub network design problem. A variable neighborhood search algorithm consisting of the local search on hubs and links was proposed to solve the model. Although the computation time of their algorithm is short, the gaps of solutions obtained by their algorithms are around or larger than 1% for many instances. de Sá et al. (2015) studied the q-hub line location problem in which the hubs are located in a set of interconnecting lines. In addition to a Benders-branch-and-cut algorithm, three heuristics based on neighborhood search, greedy randomized adaptive search and adaptive large neighborhood search, were proposed to solve it.

In recent years, some novel methods with the consideration of spatial properties of networks have also been developed. Contreras et al. (2010) proposed a model for the tree of hub location problem where p hubs have to be located by means of a non-directed tree, i.e., there have to be p-1 hub links (Contreras et al., 2009b). An integer programming formulation and some valid inequalities were used to solve the problem. Figueiredo et al. (2014) proposed a two-stage method for hub location. In their method, p regional hubs are obtained by solving a p-median problem in the first stage; then a q-hub location problem is solved in the network with these p regional hubs. Another similar approach was proposed by Peker et al. (2016). Based on the spatial properties and travel demands of nodes, a clustering-based potential hub set is generated, which can help to reduce the computational complexity of the problems by narrowing the solution sets. Contreras et al. (2016) proposed a branch-and-cut algorithm for the cycle hub location problem which connects hubs by means of cycles. A meta-heuristic based on greedy search was used to find feasible

Table 1: Parameters for the incomplete hub location problem.

Parameter	Description
V	Set of nodes ($\|V\| = n$)
w_{ij}	Travel demand between node i and node j ($i, j \in V, i \neq j$)
c_{ij}	Distance between node i and j ($i, j \in V, i \neq j$)
f^0, f^1, f^2, f^3	Fixed cost per unit distance for establishing four types of links
b^0, b^1, b^2, b^3	Variable cost per unit distance per unit travel demand for four types of links
A_{ij}	Link specific fixed cost per unit distance for four types of links ($i, j \in V, i \neq j$)
f_k^H	Fixed cost for constructing a hub in node k ($k \in V$)

Table 2: Decision variables for the incomplete hub location problem.

Variable	Domain	Description
z_k	$\{0, 1\}$	Decide the construction of a hub in node $k \in V$
y_{ij}^0	$\{0, 1\}$	Decide the construction of direct link between node $i, j \in V, i \neq j$
y_{ik}^1	$\{0, 1\}$	Decide the construction of collection tributary link between node $i, k \in V, i \neq k$
y_{km}^2	$\{0, 1\}$	Decide the construction of hub link between node $k, m \in V, k \neq m$
y_{mj}^3	$\{0, 1\}$	Decide the construction of distribution link between node $m, j \in V, m \neq j$
h_{ijk}	$[0, 1]$	Fraction of flow from $i \in V$ to $j \in V, i \neq j$ on collection access link $i - k, k \in V, k \neq i$
x_{ijkm}	$[0, 1]$	Fraction of flow from $i \in V$ to $j \in V, i \neq j$ on hub link $k - m, k, m \in V, k \neq m, i \neq m, j \neq k$
t_{ijm}	$[0, 1]$	Fraction of flow from $i \in V$ to $j \in V, i \neq j$ on ending distribution link $m - j, m \in V, m \neq j$

solutions.

3. Problem Formulation

The hub location model in this section was proposed by de Camargo et al. (2017). Specifically, four types of links are used in this model: Direct links, collection links, hub links and distribution links, denoted by symbols *0,1,2,3*,

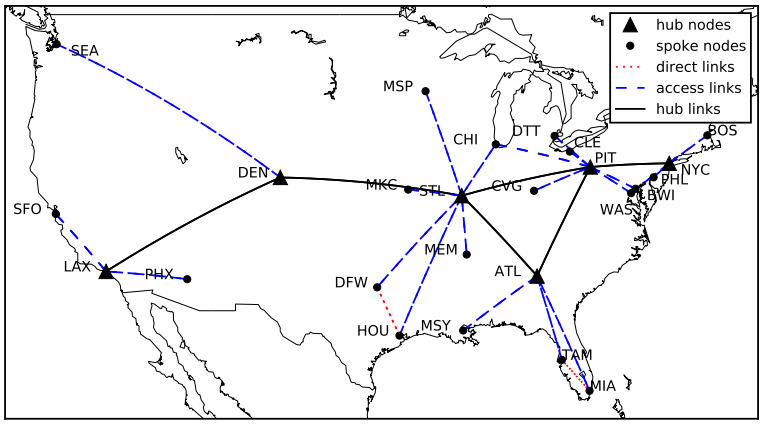


Figure 1: Visualization of a solution for the incomplete hub location problem. The black bold triangles and circles stand for hub nodes and spoke nodes respectively. Solid black lines refer to hub links, dashed blue lines are access links and dotted red lines represent direct links. Cost for installing hubs $f_k^H = 1.0 \times 10^7$, fixed cost $f = [2500, 3000, 3500, 3000]$, variable cost $b = [0.08, 0.04, 0.03, 0.04]$.

respectively. Direct links serve the direct traffic demand between two spoke nodes. Collection links represent links from spoke nodes to hub nodes. Hub links are established to transfer the high volume of traffic between hub nodes. Distribution links represent the traffic from hub nodes to spoke nodes. Collection links and distribution links are called access links together. As the hub nodes are not forced to be fully interconnected and each spoke node is allowed to be allocated to multiple hub nodes, the model represents quite a wide range of topologies. Based on a set of parameters (Table 1) and variables (Table 2), a multiple allocation hub location problem with incomplete inter-hub network can be modeled as follows:

$$\min \sum_{k\in V} f_k^H z_k + \sum_{i,j\in V, j\neq i} \tilde{c}_{ij}^0 y_{ij}^0 + \sum_{i,k\in V,k\neq i} \tilde{c}_{ik}^1 y_{ik}^1 + \sum_{k,m\in V,m\neq k} \tilde{c}_{km}^2 y_{km}^2 + \sum_{j\,m\in V,m\neq j} \tilde{c}_{mj}^3 y_{mj}^3 \tag{1}$$

$$+ \sum_{i,j\in V,j\neq i} w_{ij} \left(\sum_{k\in V,k\neq i} \hat{c}_{ik}^1 h_{ijk} + \sum_{k,m\in V,j\neq k,m\neq i,k\neq m} \hat{c}_{km}^2 x_{ijkm} + \sum_{m\in V,m\neq j} \hat{c}_{mj}^3 t_{ijm} \right)$$

$$\text{s.t.} \sum_{m\in V,m\neq j} t_{ijm} + \sum_{k\in V,k\neq j} x_{ijkj} + h_{ijj} + y_{ij}^0 = 1,\ \forall i, j \in V, i \neq j \tag{2}$$

$$h_{ijm} + \sum_{k\in V,k\neq j,k\neq m} x_{ijkm} = \sum_{k\in V,k\neq i,k\neq m} x_{ijmk} + t_{ijm},\ \forall i, j, m \in V, i \neq j, i \neq m, j \neq m \tag{3}$$

$$t_{iji} + \sum_{m\in V,m\neq i} x_{ijim} = z_i,\ \forall i, j \in V, i \neq j \tag{4}$$

$$h_{ijk} + \sum_{m\in V,m\neq k,m\neq j} x_{ijmk} \leq z_k,\ \forall i, j, k \in V, i \neq j, i \neq k, j \neq k \tag{5}$$

$$h_{ijj} + \sum_{k\in V,k\neq j} x_{ijkj} = z_j,\ \forall i, j \in V, i \neq j \tag{6}$$

$$h_{ijk} \leq y_{ik}^1,\ \forall i, j, k \in V, i \neq j, k \neq i \tag{7}$$

$$x_{ijkm} \leq y_{km}^2,\ \forall i, j, k, m \in V, i \neq j, k \neq j, m \neq i, k \neq m \tag{8}$$

$$t_{ijm} \leq y_{mj}^3,\ \forall i, j, m \in V, i \neq j, m \neq j \tag{9}$$

$$\sum_{k\in V} z_k = p \tag{10}$$

where $\tilde{c}_{ij}^0 = c_{ij}(f^0 + A_{ij} + b^0 w_{ij})$, $\tilde{c}_{ik}^1 = c_{ik}(f^1 + A_{ik})$, $\tilde{c}_{km}^2 = c_{km}(f^2 + A_{km})$, $\tilde{c}_{mj}^3 = c_{mj}(f^3 + A_{mj})$, $\hat{c}_{ik}^1 = c_{ik}b^1$, $\hat{c}_{km}^2 = c_{km}b^2$, $\hat{c}_{mj}^3 = c_{mj}b^3$. The objective function (1) is to find the most cost-efficient network structure with proper assignment of hubs, links and the corresponding traffic flows through these links. Constraint (2) forces that for each OD pair, one of the four links must be used to serve the traffic flow for its destination node j. Constraint (3) is the flow conservation equation for intermediate hub node m. Constraints (4–6) ensure that hub nodes can be connected to access links and hub links only. Constraints (7–9) provide the infrastructure for feasible traffic flow paths. Finally, constraint (10) limits the number of hubs to p.

Note that our model has demands and variable transportation cost between each pair of nodes as input. An optimal solution of the incomplete hub location problem for the CAB dataset is visualized in Figure 1. There are six hubs (LAX, DEN, STL, ATL, PIT and NYC). Most spoke nodes are assigned to only one hub, except CHI and WAS. Two pairs of spoke nodes are connected by direct links: (DFW, HOU) and (TAM, MIA). If there is a direct link between a pair of spoke nodes, the travel demands can be satisfied through the link directly. Otherwise, a path consisting of a collection link, a distribution link and one or more hub links, is determined.

4. Methodology: Hubbiness of node pairs and network design patterns

This section introduces a novel solution algorithm for the hub location model presented in Section 3. In a nutshell, our algorithm consists of three steps. The first step identifies the interesting node pairs in the network, based on their *hubbiness* value. The hubbiness of a given pair of nodes estimates their quality for providing the only two hubs in a

Algorithm 1 Overview on HUBBI

Input: Incomplete hub location problem with n nodes and p hubs.
Output: Location of hubs and direct/access/hub links.
▷ STEP 1: Design an initial solution for p=2 (see Section 4.1).
Compute hubbiness *hubbi* for all pairs of nodes in the network .
Choose the node pair *(k,m)* with the highest hubbiness $hubbi^{km}$, let $k - m$ and $m - k$ be the only hub links, assign direct access arcs and spoke links as suggested by $hubbi^{km}$.
▷ STEP 2: Incrementally improve the solution for larger p (see Section 4.2)
Let $p^c = 2$.
while $p^c < p$ **do**
    Perform operator *Tree-Extension* on the current assignment (see Section 4.2.1).
    Perform operator *Cycle-Extension* on the current assignment (see Section 4.2.2).
    $p^c = p^c + 1$.
end while
▷ STEP 3: Explore neighborhood of currently best solution
Perform VNS on the current assignment (see Section 4.3).

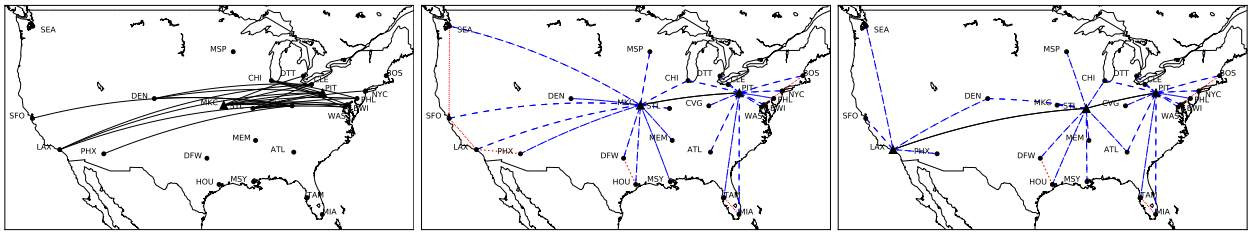
Algorithm 2 Computation of hubbiness

Input: A network instance with n nodes, the set of nodes V, a pair of nodes *(k,m)*.
Output: The value of hubbiness of *(k,m)*.
Install hubs on nodes k and m and establish hub links *k-m,m-k*.
Fully connect the direct links and access links.
Compute the total cost for the current assignment as a reference (see Appendix A.1).
Perform the "**Removing**" operator on existing direct links (see Appendix A.2, Algorithm 6).
Perform the "**Removing**" operator on existing access links (see Appendix A.3, Algorithm 8).
Perform the "**Replacing**" operator on access links (see Appendix A.3,, Algorithm 10).
The finally total cost is represented by TC^{km}.
The hubbiness of node pair *(k,m)* is $hubbi^{km} = \frac{1}{TC^{km}}$.

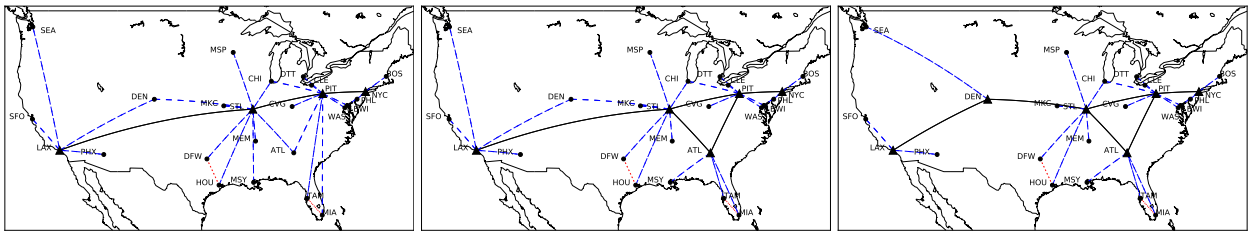
given network. The formal definition of hubbiness is introduced in Section 4.1. Using our ranking of node pairs, we obtain an initial solution for the case of 2 hubs. In a second step, we propose an iterative network design algorithm, which increases the number of hubs up to p (see Section 4.2). We propose two operators which attempt to rewrite and expand the previous solution: Tree-Extension and Cycle-Extension. After the second step, we obtain an initial solution with p hubs. Finally, the third step consists of performing variable neighborhood search (VNS) to improve the solution further by exploring its neighborhood (see Section 4.3). The overall structure of our network design algorithm is summarized in Algorithm 1. Our algorithm is called *HUBBI*, since it is based on the usage of hubbiness for selecting initial node pairs and guiding the iterative network design.

4.1. Hubbiness

We propose a quality estimation for node pairs in a network, called *hubbiness*. For a given node pair *(k,m)*, the value of $hubbi^{km}$ approximates the reciprocal of the total cost of a solution where k and m are the only two hubs in the network, including fixed (setup) cost and variable (transportation) costs. The hubbiness of node pairs induces a high-quality ranking of hub pairs in the network and is used to generate an initial solution for $p = 2$. The process for computing hubbiness of node pair *(k,m)* is shown in Algorithm 2. It simulates the installment of hubs at nodes k and m, together with heuristic decisions on the installment of direct links and access links. Overall, we implement a greedy search algorithm which starts with a fully-connected assignment. Such an assignment is usually very expensive in terms of setup costs. Therefore, we first remove greedily direct links, ordered by increasing travel demands, which reduce the total cost of the solution. The intuition is that pairs of spoke nodes with lower travel demands are less likely to be connected directly. Second, we remove greedily all access links, ordered by decreasing distances to hubs, which reduce the total cost of the solution, with the intuition that spoke nodes are less likely to be assigned to farther hubs. Third, we try to replace greedily all access links, by connecting them to not-yet connected hubs, sorting the hub candidates in increasing order of distances. The details of these operators are omitted here for brevity, but they can be found in Appendix A.2 and Appendix A.3, along with pseudo-codes. We define the hubbiness of the node pair *(k,m)*



(a) The top n node pairs with the best hubbiness. (b) The optimal solution with $p = 2$. (c) The optimal solution with $p = 3$.



(d) The optimal solution with $p = 4$. (e) The optimal solution with $p = 5$. (f) The optimal solution with $p = 6$.

Figure 2: The visualization of the top n node pairs with the best approximate hubbiness (a) and the optimal solutions for different $p \in \{2, 3, 4, 5, 6\}$ (b–f). The black bold triangles and circles stand for hub nodes and spoke nodes respectively. Solid black lines refer to hub links, dashed blue lines are access links and dotted red lines represent direct links. Cost for installing hubs $f_k^H = 1.0 \times 10^7$, fixed cost $f = [2500, 3000, 3500, 3000]$, variable cost $b = [0.08, 0.04, 0.03, 0.04]$.

as follows:

$$hubbi^{km} = \frac{1}{TC^{km}} \tag{11}$$

where TC^{km} is the total cost of the final assignment with node pair *(k,m)*, as obtained by Algorithm 2. Finally, the node pair with the highest hubbiness is connected with bi-directional hub links for the case of *p=2*. As an example, the top n links with the highest hubbiness for the CAB dataset are shown in Figure 2(a) and the optimal solutions for $p \in \{2, 3, 4, 5, 6\}$ (Figure 2(b–f)). In Section 4.2, we describe how to obtain solutions for $p > 2$.

4.2. Incremental Network Design

In this section, we aim to derive initial solutions for $p > 2$, by using several network design patterns (**Tree-Extension** and **Cycle-Extension**).

4.2.1. Operator Tree-Extension

In this subsection, we propose the first operator for network design, called **Tree-Extension**. For a given network with p hubs, we exchange one existing hub and, in addition, build another new hub, effectively creating branches on the hub network. The process of Tree-Extension is shown in Algorithm 3. Let Nei^h be the set of neighbor hubs of hub h. We sort all spoke nodes with increasing deviation of hubbiness to h which is defined in Equation (12) and select the top cn of them for a determined number cn. The experiments show that $cn = \sqrt{n}$ is sufficient for obtaining good solutions.

$$Dev^{sh} = \sum_{X \in Nei^h} \left\| hubbi^{sX} - hubbi^{hX} \right\| \tag{12}$$

From each hub link l, we select two hub nodes h_1 and h_2 (it is possible that they are the same). If the case of node pair (h_1, h_2) has not appeared before (seen pairs are tracked in set $\mathcal{G}$), for each spoke node s_1 from the top cn nodes with the smallest values of Dev^{sh_1} and each spoke node s_2, we switch the role between the hub h_1 and spoke s_1, build a new hub on s_2, establish new hub links $s_2 - h_2, h_2 - s_2$ and do some assignment on access/direct links, as shown

Algorithm 3 Tree-Extension from p hubs to $p+1$ hubs

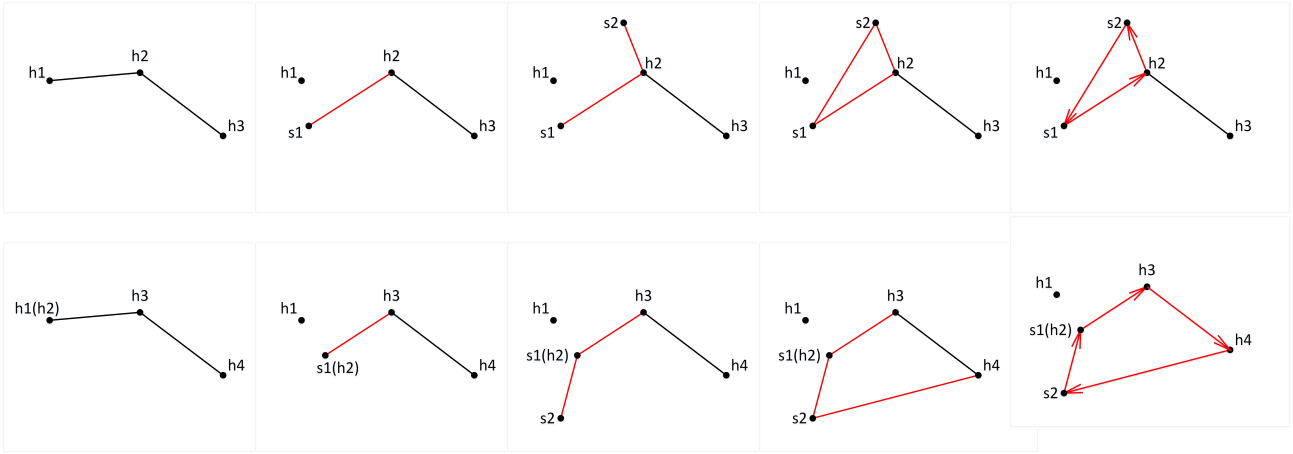
Input: A network instance with n nodes, the set of nodes V, the set of hub nodes $\mathcal{H}^o(|\mathcal{H}^o| = p)$, the set of four types of links $\mathcal{L}^o$, initial total cost $Obj^o = inf$.
Output: The new set of hub nodes $\mathcal{H}$, the new set of four types of links $\mathcal{L}$, final total cost Obj

Let $\mathcal{H} = \mathcal{H}^o$, $\mathcal{L} = \mathcal{L}^o$, Obj=Obj^o. Let $\mathcal{L}^2$ be the set of hub links in $\mathcal{L}^o$. Let $\mathcal{G} = \emptyset$
for each hub link $l \in \mathcal{L}^2$ **do**
  for each hub node $h_1 \in l$ **do**
    for each hub node $h_2 \in l$ **do**
      if $(h_1, h_2) \notin \mathcal{G}$ **then**
        Let $\mathcal{G} = \mathcal{G} \cup \{(h_1, h_2)\}$.
        Sort spoke nodes with increasing order of deviation of hubbiness to hub h_1 and obtain a set $N_{h_1}^{cn}$ of first cn nodes. Let $N_{h_1}^{cn} = \{h_1\} \cup N_{h_1}^{cn}$.
        Let N_{h_2} be the set of spoke nodes.
        for each $s_1 \in N_{h_1}^{cn}$ **do**
          if $h_1 == h_2$ **then**
            Let $h_2 = s_1$.
          end if
          for each $s_2 \in N_{h_2}$ **do**
            Based on the current $\mathcal{H}^o$, $\mathcal{L}^o$, do the following operations:
            Exchange the roles between h_1 and s_1, i.e., letting s_1 be a new hub and h_1 be a spoke, connecting s_1 to all neighbor hubs of h_1 and all spoke nodes that were connected to h_1.
            Build a new hub on s_2. Establish bi-directional hub links between hub h_2 and s_2. Connect all spoke nodes to hub s_2. Establish direct links between spoke h_1 and all other spoke nodes.
            Compute the total cost for the current assignment as a reference (see Appendix A.1).
            Perform the "**Removing**" operator on existing direct links (see Appendix A.2, Algorithm 6).
            Perform the "**Removing**" operator on existing access links (see Appendix A.3, Algorithm 8).
            Perform the "**Replacing**" operator on access links (see Appendix A.3, Algorithm 10).
            Perform the **Tree-Close** (see Algorithm 4).
            The current total cost, set of hubs and set of hub links are represented by Obj^o, $\mathcal{H}^o$ and $\mathcal{L}^o$.
            if Obj^o <Obj **then**
              Let Obj=Obj^o, $\mathcal{H} = \mathcal{H}^o$ and $\mathcal{L} = \mathcal{L}^o$.
            end if
            Return the status of Obj^o, $\mathcal{H}^o$ and $\mathcal{L}^o$ to Step 8.
          end for all
        end for all
      end if
    end for all
  end for all
end for all

Algorithm 4 Tree-Close: A sub-operator of Tree-Extension

Input: The initial set of four types of links $\mathcal{L}^o$, new hub s_2, the set of hub nodes $\mathcal{H}^{s_2}$ that are not connected to hub s_2, current total cost Obj^o.
Output: The new set of four types of links $\mathcal{L}$, final total cost Obj

Let $\mathcal{L} = \mathcal{L}^o$, Obj=Obj^o.
for each hub $h \in \mathcal{H}^{s_2}$ **do**
  Simulate establishing bi-directional hub links between s_2 and h. The updated set of four types of links is represented by $\mathcal{L}^o$.
  Compute the total cost Obj^o for the current assignment.
  if Obj^o <Obj **then**
    Let Obj=Obj^o and $\mathcal{L} = \mathcal{L}^o$.
  end if
  if There exist hub cycles with reversed hub links **then**
    Let Cy be the set of hub cycles that have reversed hub links.
    for each $cycle \in Cy$ **do**
      Simulate removing all reversed hub links of $cycle$. The updated set of four types of links is represented by $\mathcal{L}^o$.
      Compute the total cost Obj^o for the current assignment.
      if Obj^o <Obj **then**
        Let Obj=Obj^o and $\mathcal{L} = \mathcal{L}^o$.
      end if
    end for all
  end if
end for all



(a) An original hub network.

(b) Replace hub h_1 with node s_1.

(c) Install a new hub s_2 and connect it with hub h_2.

(d) Simulate connecting hub s_2 with other hubs.

(e) If there exists a hub cycle, simulate removing all reversed hub links of this cycle.

Figure 3: The process of Tree-Extension. The sub-figures in the upper part are for the case that $h_1 \neq h_2$ and the sub-figures in the lower part are for the case that $h_1 = h_2$. We perform the above operations for all hub links in the original network with p hubs. Black links are original hub links and red links are new hub links.

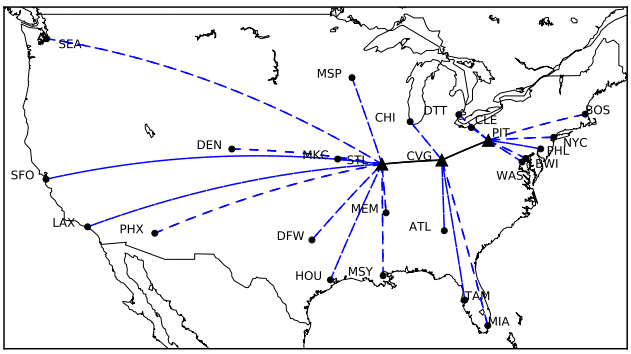
in Steps 15–16 in Algorithm 3. Afterwards, we implement a greedy search algorithm on this initial assignment: We first remove greedily direct links, ordered by increasing travel demands, which reduce the total cost of the solution. Second, we remove greedily all access links, ordered by decreasing distances to hubs, which also reduce the total cost of the solution. Third, we try to replace greedily all access links, by connecting them to not-yet connected hubs, sorting the hub candidates in increasing order of distances. The details of operator Tree-Extension are shown in Figure 3.

When computing hubbiness for a pair of nodes, there are only two hub links between both nodes. However, the cases of hub links for more than two hubs are more complex. In addition, bi-directional links between hubs are established in the above operators. If the fixed cost for establishing hub links is very expensive, it is better to establish a unidirectional cycle in the hub network (see the unidirectional hub cycle in Figure 4(b)). Therefore, we propose a sub-operator of Tree-Extension, called **Tree-Close**. After connecting new hub s_2 to h_2, we greedily establish hub links between s_2 and other hubs while the total cost is reduced. For each discovered hub cycle with reversed hub links, we simulate removing all reversed hub links of the cycle and generating a unidirectional cycle. If the total cost is reduced, we remove the reversed hub links. Note that we try to replace all bi-directional links in all cycles in the hub network by unidirectional links and select the cheapest option as the sub-solution. The details of sub-operator Tree-Close are shown in Algorithm 4.

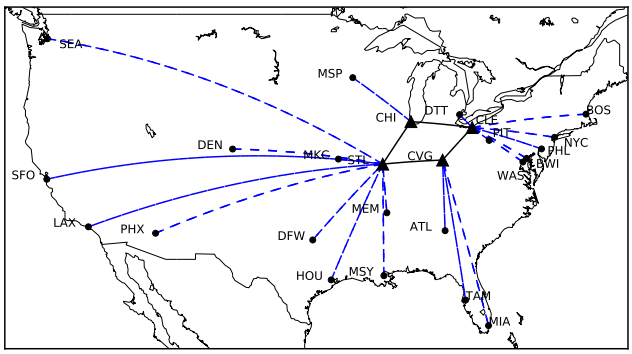
4.2.2. Operator Cycle-Extension

In the last section, Tree-Extension extends the hub network mainly based on the bi-directional connection between hubs to guarantee the connectivity of the hub network. Although the sub-operator Tree-Close is proposed, this sub-operator addresses the special case that a cycle is newly generated from a tree. It does not cover the case of extending a small cycle to a large cycle. Therefore, in this section, we propose a new operator called **Cycle-extension**. As shown in Figure 5, the procedure of this operator is as follows:

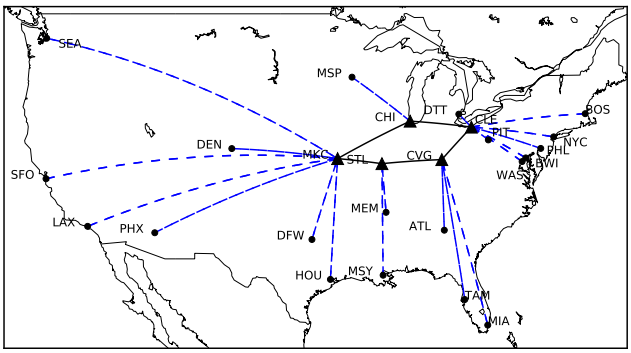
1. For each hub link $h_1 - h_2$, we select each of its vertices (e.g., h_1). For each node (e.g., s_1) from the top cn spoke nodes sorted by Dev^{sh}, we simulate replacing hub h_1 by node s_1.
2. For each possible spoke node s_2, we simulate installing a new hub s_2 and replace hub link $s_1 - h_2$ with a hub path $s_1 - s_2 - h_2$.
3. We implement a greedy search algorithm on the current assignment, similar to the computation of hubbiness: We first remove greedily direct links, ordered by increasing travel demands, which reduce the total cost of the



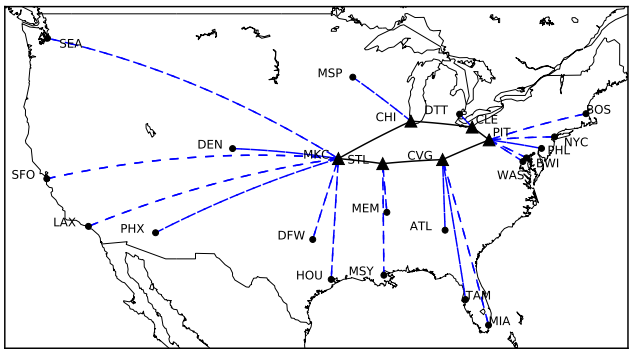
(a) Two bi-directional hub links consisting of four unidirectional hub links.



(b) A unidirectional hub cycle consisting of four unidirectional hub links.

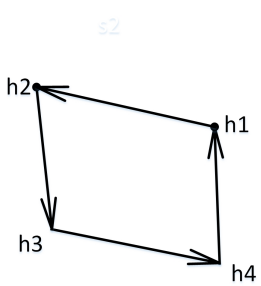


(c) A unidirectional hub cycle consisting of five unidirectional hub links.

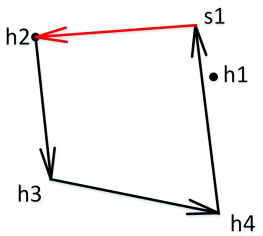


(d) A unidirectional hub cycle consisting of six unidirectional hub links.

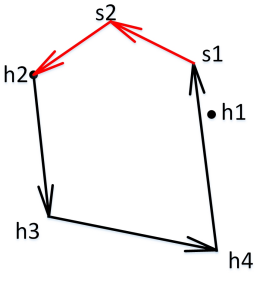
Figure 4: The comparison between Tree-Close and Cycle-Extension ($p \in \{3,4,5,6\}$): Tree-Close is used in the network design from (a) to (b); Cycle-Extension is used in the network design from (b) to (c) and from (c) to (d). Cost for installing hubs $f_k^H = 0$, fixed cost $f = [8000, 50000, 400000, 50000]$, variable cost $b = [0.8, 0.04, 0.01, 0.04]$.



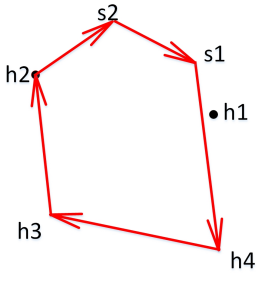
(a) An original hub cycle.



(b) Hub h_1 is replaced by node s_1.



(c) A new hub s_2 is installed and link s_1-h_2 is replaced by path $s_1-s_2-h_2$.



(d) Reverse the directions of all links.

Figure 5: The process of Cycle-extension. We perform the above operations for all hub links in the original network with p hubs. Black links are original hub links and red links are new hub links. Because the directions of hub links in the optimal solutions might be reversed with the increasing value of p, we reverse the hub links in (d) and see whether the total cost is reduced.

solution. Second, we remove greedily all access links, ordered by decreasing distances to hubs, which reduce the total cost of the solution. Third, we try to replace greedily all access links, by connecting them to not-yet connected hubs, sorting the hub candidates in increasing order of distances.

4. Note that with the increasing value of p, the directions of hub links in the optimal solutions might be reversed. Thus, we try to reverse the direction of all hub links and update the total cost.
5. After exploring all possible cases, the solution with the minimum total cost is finally selected.

Algorithm 5 Variable Neighborhood Search

Input: The set of nodes V, the set of hub nodes $\mathcal{H}^o$, the set of four types of links $\mathcal{L}^o$, current total cost Obj^o, variable cost $Cost^o_{ij}$, transportation paths $Path^o_{ij}$ $(i, j \in V)$, the maximum number of iterations $iter_{max}$, the maximum number of iterations without improvement $iter_{impr}$, $r = 1$, $flag = 0$.

Output: The new set of hub nodes $\mathcal{H}$, final total cost Obj

1: Let $\mathcal{H} = \mathcal{H}^o$, $\mathcal{L} = \mathcal{L}^o$, Obj=Obj^o, $Cost_{ij}$=$Cost^o_{ij}$, $Path_{ij}$=$Path^o_{ij}$.
2: **while** r <$iter_{max}$. **do**
3: Let $Spokes$=$V \backslash \mathcal{H}^o$.
4: **for each** $h \in \mathcal{H}^o$ **do**
5: **for each** $i \in Spokes$ **do**
6: Replace the hub h with node i. Let $\mathcal{H}^c$ and $\mathcal{L}^c$ be the updated sets of hubs and four types of links.
7: Compute the new total cost Obj^c, $Cost^c_{ij}$ and $Path^c_{ij}$ $(i, j \in V)$.
8: Let $j = 1$.
9: **while** $j < 4$ **do**
10: **if** $j == 1$ **then**
11: Perform the operators "**Removing**" and "**Adding**" on direct links (see Appendix A.2, Algorithms 6–7).
12: **end if**
13: **if** $j == 2$ **then**
14: Perform the operators "**Removing**", "**Adding**" and **Replacing** on access links (see Appendix A.3, Algorithms 8–10).
15: **end if**
16: **if** $j == 3$ **then**
17: Perform the operators "**Removing**", "**Adding**" and **Replacing** on hub links (see Appendix A.4, Algorithms 11–13).
18: **end if**
19: Let $j = j + 1$. Let Obj^n, $\mathcal{L}^n$, $Cost^n_{ij}$ and $Path^n_{ij}$ $(i, j \in V)$ be the current total cost, set of four types of links, variable cost and paths.
20: **if** Obj^n <Obj^c **then**
21: Let Obj^c=Obj^n, $\mathcal{L}^c = \mathcal{L}^n$, $Cost^c_{ij}$=$Cost^n_{ij}$, $Path^c_{ij}$=$Path^n_{ij}$, and $j = 1$.
22: **end if**
23: **end while**
24: **if** Obj^c <Obj **then**
25: Let $\mathcal{H} = \mathcal{H}^c$, $\mathcal{L} = \mathcal{L}^c$, Obj=Obj^c, $Cost_{ij}$=$Cost^c_{ij}$, $Path_{ij}$=$Path^c_{ij}$, $flag = 0$.
26: **else**
27: $flag = flag + 1$.
28: **end if**
29: **if** $flag \geq iter_{imp}$ **then**
30: **return**
31: **end if**
32: Return the status of Obj^o, $\mathcal{H}^o$ and $\mathcal{L}^o$ to Step 3.
33: **end for all**
34: **end for all**
35: Let $\mathcal{H}^o = \mathcal{H}$ and $\mathcal{L}^o = \mathcal{L}$.
36: **end while**

Tree-Close and Cycle-extension have different effects on the network construction. As an example, the optimal solutions for several instances with $p \in \{3, 4, 5, 6\}$ in CAB dataset are shown in Figure 4. Tree-Close is applied to generate a new circle from a bi-directional tree (Figure 4(b)), while Cycle-extension is used to extend a small cycle to a large cycle (Figure 4(c–d))

4.3. Variable Neighborhood Search (VNS)

The solutions obtained by the incremental network design are quite good (see Section 5), but for further improvement we employ variable neighborhood search (VNS) to seek better solutions. According to Ilić et al. (2010), there are three types of VNS strategies for hub location problems: Sequential strategy (Seq-VNS), nested strategy (Nest-VNS) and mixed strategy (Mix-VNS). Seq-VNS needs the shortest time but explores the fewest neighborhoods. Nest-VNS explores a large set of neighborhoods, but its run time is not acceptable. Mix-VNS allows a trade-off between the number of neighborhoods and run time. Therefore, we use Mix-VNS in this study.

Our algorithm is shown in Algorithm 5. For a given initial solution with the hub set $\mathcal{H}^o$ obtained by the incremental network design, we generate all hub sets that have one hub different from $\mathcal{H}^o$ in the nested level and these hub sets are called neighborhoods (Step 6 in Algorithm 5). Then, other local search methods (removing/adding direct links (Step

Table 3: The gaps of solutions and run time of three types of local search and variable neighborhood search for the CAB25 dataset with $p = 5$.

Operator	Maximum gap (%)	Minimum gap (%)	Average gap (%)	Median gap (%)	Average run time (s)
Modifying direct links	51.56	14.09	26.26	25.28	0.03
Modifying access links	39.18	19.44	28.05	28.16	0.05
Modifying hub links	35.19	7.96	17.93	15.99	0.35
Variable neighborhood search	8.18	0.57	2.58	1.84	26.2

11), removing/adding/replacing access links (Step 14) and removing/adding/replacing hub links (Step 17)) are applied on each neighborhood greedily (the detailed procedures for these operators are shown in Appendix A.2–Appendix A.4). If there is no improvement for a certain number of continuous iterations, the algorithm will be terminated (Steps 29–31). Note that in Step 3 of Algorithm 5, the set *Spokes* can be a subset of the spoke nodes that are close to hub h to reduce the run time.

5. Computational Results

5.1. Datasets and experimental setup

To evaluate the performance of HUBBI proposed in Section 4, several well-known datasets are used as case studies: CAB with 25 nodes (O'Kelly, 1987) and CAB with 100 nodes (as provided by O'Kelly[1]), TR with 81 nodes (Çetiner, 2003), AP with 200 nodes (Ernst and Krishnamoorthy, 1996), USA423 with 423 nodes (Peiró et al., 2014), as well as URAND with 100 and 200 nodes (Meyer et al., 2009). In order to generate datasets with smaller sizes, inspired by de Camargo et al. (2017), a selection of the first n nodes from the datasets CAB100, TR81, USA423 is adopted for CAB dataset with 30 and more nodes, as well as TR dataset and USA dataset with 25 and more nodes. All AP datasets are generated by publicly available code[2]. The enhanced Benders decomposition method, which is the fastest exact method (de Camargo et al., 2017), is implemented as a reference benchmark using CPLEX. We also tried to solve a few instances with VNS and LocalSolver[3], a black-box optimization solver based on heuristics, but the results were poor.

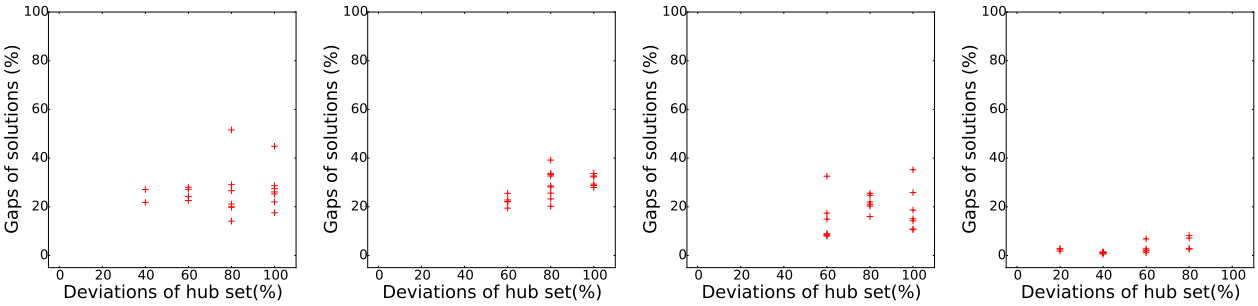
All experiments were executed on a server with 40 cores and 450 GB RAM, running Fedora 24. For a fair comparison, all programs were run using a single thread only. Regarding the setup of VNS, we make a few design decisions as follows. In Ilić et al. (2010), the maximum number of iterations without improvement and the maximum number of all iterations in the VNS were set to $\frac{n}{2}$ and $5n$ for the uncapacitated single allocation p-hub median problem, where n is the number of nodes in the networks. Because the complexity of the incomplete hub location problem is significantly higher, we set the values of these two parameters to $2n$ and $5n$, respectively. Moreover, we have tested different sizes of the set *Spokes* for VNS (see Algorithm 5) and different values of cn for HUBBI (i.e., we only try to use the top cn nodes ordered by the parameter Dev^{sh}, see Section 4.2.1). There are no significant differences for run time and solution quality, except from very special cases, which could appear for either choice of values. Therefore, we use medium settings here: Let $|Spokes| = \frac{n}{5}$ and $cn = \sqrt{n}$. We use the following variant of the link specified fixed cost (A_{ij}) as implemented by de Camargo et al. (2017), which was kindly provided by the authors:

$$A_{ij} = \begin{cases} 1, & \text{if } c_{ij} < 1.1498 * 1500 \\ 2, & \text{otherwise} \end{cases}$$

5.2. The need for problem-specific heuristics

In the first series of experiments, we compare the standard local search strategies in three neighborhood structures, i.e., modifying direct links (Appendix A.2), modifying access links (Appendix A.3), and modifying hub links (Appendix A.4), together with variable neighborhood search (VNS, Section 4.3). The CAB25 dataset with the number

[1] `https://www.researchgate.net/project/Studies-in-Hub-Location-and-Network-Design`
[2] `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/phub1.txt`
[3] `http://www.localsolver.com/`



(a) Modifying direct links. (b) Modifying access links. (c) Modifying hub links. (d) Variable neighborhood search.

Figure 6: The gaps of solutions (y-axis) and deviations of hub sets (x-axis) obtained by three types of local search and variable neighborhood search for the CAB25 dataset with $p = 5$.

Table 4: The performance of the enhanced Benders decomposition method, LocalSolver, VNS (20 times) and HUBBI on the CAB25 datasets. The fixed cost and variable cost are $f = [2500, 3000, 3500, 3000]$ and $b = [0.08, 0.04, 0.03, 0.04]$. The cost for establishing a hub is $f_k^H = 10^7$ $(k \in V)$. LocalSolver cannot provide any feasible solution within 3600 seconds for CAB25 dataset. HUBBI outperforms VNS on both solution qualities and computation times.

Datasets	Size	p	Solution of Benders (optimal)	Time of Benders (s)	Max gap of VNS	Average gap of VNS	Min gap of VNS	Average time of VNS (s)	Gap of HUBBI	Time of HUBBI (s)
CAB	25	2	511,711,559	238	3.26%	1.13%	0.01%	3.76	0.00%	1.1
CAB	25	3	477,904,481	367.5	5.76%	1.45%	0.02%	6.68	0.00%	6.2
CAB	25	4	468,438,577	586.9	6.29%	1.80%	0.55%	14.23	0.00%	11.1
CAB	25	5	461,450,560	628.1	8.18%	2.58%	0.57%	26.20	0.00%	16.5

Table 5: The groups of fixed costs and variable costs.

Test	Cost for installing hubs (f_k^H)	Fixed cost $[f^0, f^1, f^2, f^3]$	Variable cost $[b^0, b^1, b^2, b^3]$
1	1.0×10^7	[2500,3000,3500,3000]	[0.08,0.04,0.03,0.04]
2	1.0×10^7	[1000,1000,1000,1000]	[0.10,0.04,0.02,0.04]

of hubs $p = 5$ is used as an initial case study. The fixed cost and variable cost are $f = [2500, 3000, 3500, 3000]$ and $b = [0.08, 0.04, 0.03, 0.04]$. The cost for establishing a hub is set to $f_k^H = 10^7$ $(k \in V)$. Twenty initial hub sets are generated randomly and all spoke nodes are connected to the closest hubs. There are no direct links in the initial solutions. Each type of local search is performed individually. The gaps of solutions (y-axis) together with the deviation of hub sets between the local optimum and the optimal solutions (x-axis) are visualized in Figure 6. The results are summarized in Table 3. Although operators modifying direct/access links need the shortest run time, the solutions obtained by them are the worse in terms of solutions qualities. VNS provides a median gap of 1.84%, the minimum gap with 20 experiments is 0.57%. The maximum gap of 8.18% shows the dependency of VNS on good initial solutions.

We also tried to use LocalSolver[4], a black-box optimization solver based on heuristics, to solve the incomplete hub location problems. As shown in Table 4, we test the performance of the enhanced Benders decomposition method, LocalSolver, VNS and HUBBI on the rather small CAB25 dataset with fixed cost and variable cost $f = [2500, 3000, 3500, 3000]$ and $b = [0.08, 0.04, 0.03, 0.04]$. The cost for establishing a hub is set to $f_k^H = 10^7$ $(k \in V)$.

5.3. Comparison of HUBBI against an enhanced Benders decomposition method

The solution quality of HUBBI is analyzed based on datasets AP and CAB with $n \in \{25, 30, 40, 50, 60, 70, 80\}$ and TR with $n \in \{25, 30, 40, 50, 60, 70, 81\}$, where n is the number of nodes. For small instances with $n \leq 30$, we use two groups of fixed costs $[f^0, f^1, f^2, f^3]$ and variable costs $[b^0, b^1, b^2, b^3]$, as shown in Table 5, following

[4] `http://www.localsolver.com/`

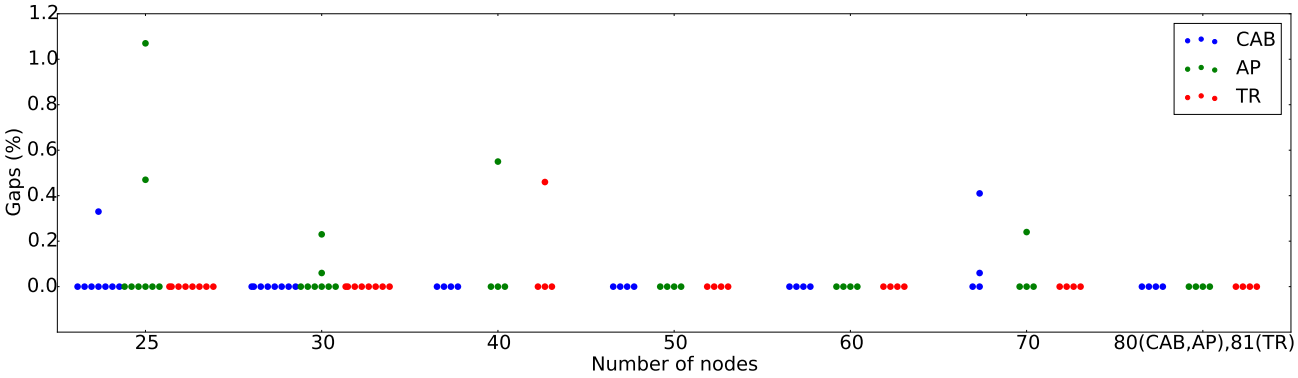
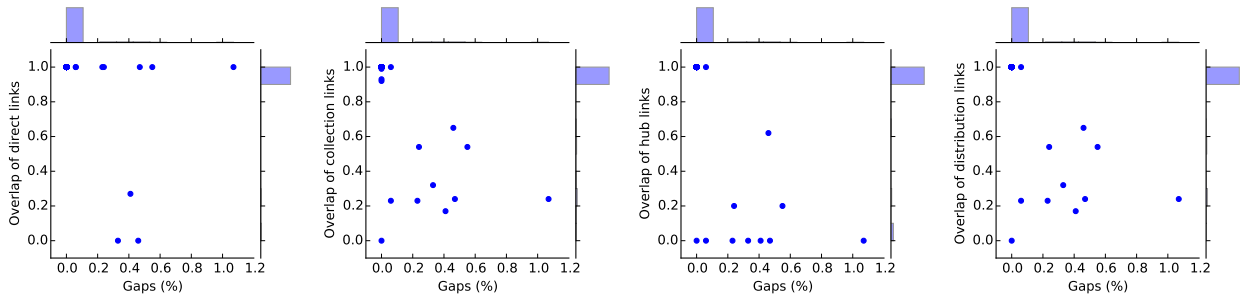


Figure 7: The gaps of solutions obtained by HUBBI method for the CAB and AP datasets for $n \in \{25, 30, 40, 50, 60, 70, 80\}$ and the TR dataset for $n \in \{25, 30, 40, 50, 60, 70, 81\}$. Here n is the number of nodes. Two groups of fixed costs and variable costs $(f, b) \in \{([2500, 3000, 3500, 3000], [0.08, 0.04, 0.03, 0.04]), ([1000, 1000, 1000, 1000], [0.10, 0.04, 0.02, 0.04])\}$ are used for the cases with $n = 25, 30$. For the cases with $n \geq 40$, only the second setting of cost is used, because of the long run time of the enhanced Benders decomposition method. The cost for installing a hub is $f_k^H = 10^7$ for all cases. The number of hubs are set to $p = 2, 3, 4, 5$. Note that except for one case with 25 nodes, the gaps of all other solutions are smaller than 1%. The detailed results regarding the solution quality for each dataset are shown in Tables B1–B3 in the Appendix.

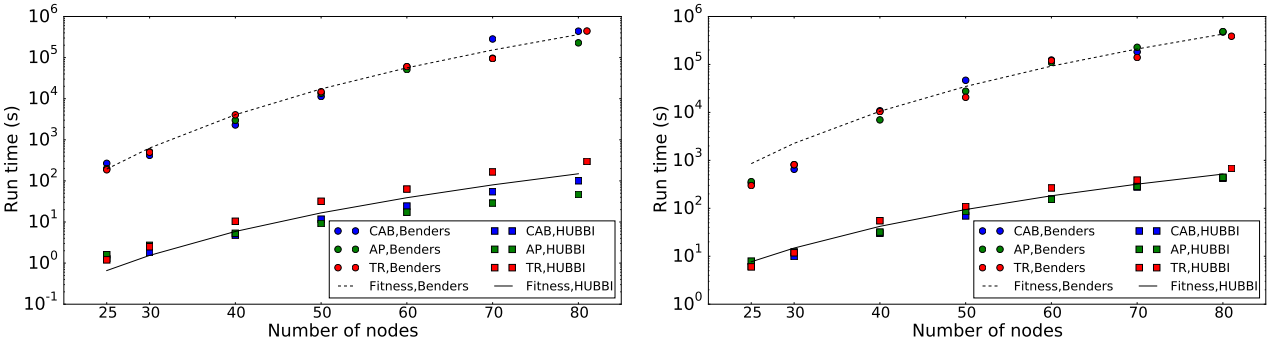


(a) Overlap of direct links. (b) Overlap of collection links. (c) Overlap of hub links. (d) Overlap of distribution links.

Figure 8: The joint plots between the quality of four types of links and the gaps of solutions, i.e., the x-axis is the percentage of solution gaps and the y-axis is the overlap between the link set obtained by HUBBI and the optimal link set. In most instances, the gaps are 0% and the overlap is 1. Interestingly, there are also a few special cases with very small solution gaps and also small overlap, which means that the values of objective functions can be very closed to the optimal solutions but the sets of links are quite different from the optimal ones in these cases.

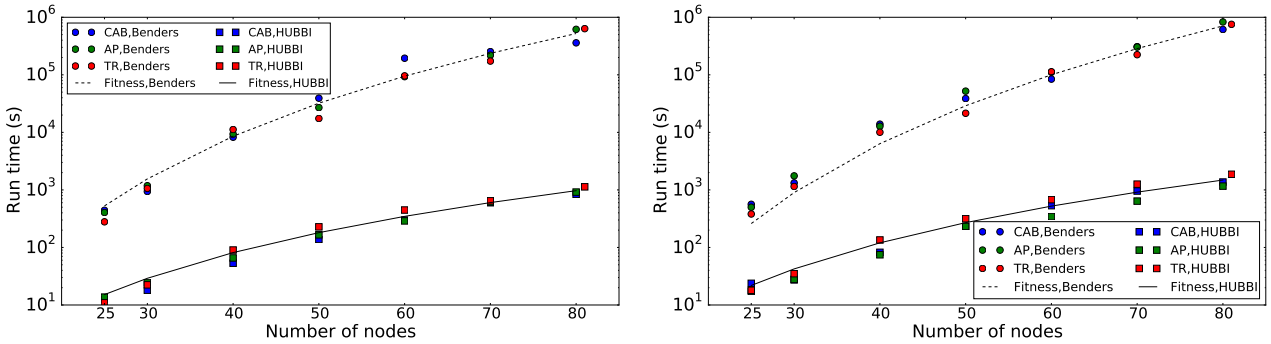
de Camargo et al. (2017). Given the long run time of the enhanced Benders decomposition method, larger instances are assessed based on the second cost setting only ($f = [1000, 1000, 1000, 1000], b = [0.10, 0.04, 0.02, 0.04]$). The cost for installing a hub k is $f_k^H = 10^7$ ($k \in V$) for all cases. The number of hubs are set to $p \in \{2, 3, 4, 5\}$. The gaps of HUBBI are computed based on the optimal solutions that are obtained by the enhanced Benders decomposition method. The swarm plot for the obtained gaps is shown in Figure 7. For each size group, the gaps for three datasets are shown individually. HUBBI provides high-quality solutions (gap<1%) for almost all cases (107/108) and the optimal solutions for over 90% (99/108) of all cases. The median gap is close/equal to zero in all cases. Interestingly, the gaps of solutions that are obtained by HUBBI keep rather small even for larger networks, which allows us to compute high-quality solutions on real-world problem instances. The detailed numerical results of our experiments regarding solution quality, for each dataset, are shown in Tables B1–B3 in the Appendix.

In order to further analyze the solution assignment obtained by HUBBI, the overlap of four link types obtained by HUBBI versus the enhanced Benders decomposition method are plotted together with the gaps of solutions for all instances (see Figure 8). The link quality is represented by the fractional overlap between the link set obtained by HUBBI and the optimal link set for each type of link. For solutions with gaps of 0%, the link overlap is one in the majority of cases, which means that HUBBI's solution coincides with the enhanced Benders decomposition method.



(a) Run time and fitness curves for $p = 2$.

b) Run time and fitness curves for $p = 3$.



(c) Run time and fitness curves for $p = 4$.

(d) Run time and fitness curves for $p = 5$.

Figure 9: The comparison of run time between HUBBI and the enhanced Benders decomposition method for the CAB and AP datasets for $n \in \{25, 30, 40, 50, 60, 70, 80\}$ and the TR dataset for $n \in \{25, 30, 40, 50, 60, 70, 81\}$. Here n is the number of nodes. One group of fixed cost $f = [1000, 1000, 1000, 1000]$ and variable cost $b = [0.10, 0.04, 0.02, 0.04]$ is used. The cost for installing a hub is $f_k^H = 10^7$ for all cases. The number of hubs are set to $p = 2, 3, 4, 5$. Note that the y-axis is in log scale. In each sub-figure, the run time for the CAB, AP and TR datasets are shown with blue, green and red colors, respectively. The black curves are the fitness curves for two methods. The enhanced Benders decomposition method and HUBBI are represented by circles and squares, respectively. The detailed run time of each method for each instance is shown in Tables B1–B3 in the Appendix.

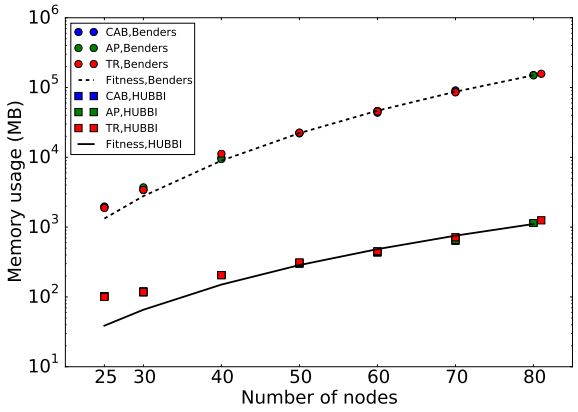
Interestingly, there are also a few special cases with very small solution gaps and also small overlap, which means that the values of objective functions can be very closed to the optimal solutions but the sets of links are quite different from the optimal ones in these cases.

Given that HUBBI often computes high-quality solutions, we compare the run time between HUBBI and the enhanced Benders decomposition method (de Camargo et al., 2017) next. The results for the cases with fixed cost $f = [1000, 1000, 1000, 1000]$ and variable cost $b = [0.10, 0.04, 0.02, 0.04]$ are shown in Figure 9. The run time of the enhanced Benders decomposition method and HUBBI for three datasets and with increasing number of nodes with $p \in \{2, 3, 4, 5\}$ is visualized in each sub-figure. Note that the y-axis is in log scale. HUBBI is 2–3 orders of magnitude faster than the exact method. For one of the largest datasets in our study, the enhanced Benders decomposition method requires around 230 hours to compute the solution, while HUBBI finds an optimal solution within 20 minutes. We also plot the fitness curves for the run time of the enhanced Benders decomposition method and HUBBI for each sub-figure. The corresponding fitness functions and R^2 are listed in Table 6. Based on our analysis, the computational complexity of the enhanced Benders decomposition method is $O(n^6)$, while HUBBI is only around $O(n^4)$. This makes HUBBI applicable for solving the hub location problems in larger networks. The detailed results regarding run time, for each instance, are shown in Tables B1–B3 in the Appendix.

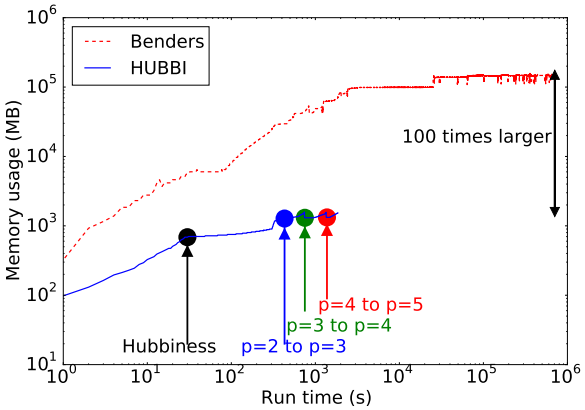
Finally, we report the memory usage of both algorithms. In Figure 10(a), the memory usage for different sizes of networks with $p = 2$ is reported. The memory usage of HUBBI increases much slower than for the enhanced Benders decomposition method. HUBBI uses only about 1% of the memory of the latter for the cases with around 80 nodes. In Figure 10(b), we show an exemplary curve for memory usage throughout solution exploration for a single instance:

Table 6: The fitness functions for the run time of the enhanced Benders decomposition method and HUBBI. Here, the run time and the number of nodes are represented by T and n, respectively.

Number of hubs p	Enhanced Benders decomposition		HUBBI	
	Fitness function	R^2	Fitness function	R^2
2	$T_{Benders} = 1.69 \times 10^{-7} \times n^{6.48}$	0.873	$T_{HUBBI} = 1.99 \times 10^{-7} \times n^{4.66}$	0.561
3	$T_{Benders} = 2.86 \times 10^{-5} \times n^{5.35}$	0.961	$T_{HUBBI} = 6.65 \times 10^{-5} \times n^{3.62}$	0.929
4	$T_{Benders} = 2.71 \times 10^{-6} \times n^{5.93}$	0.926	$T_{HUBBI} = 1.53 \times 10^{-4} \times n^{3.58}$	0.976
5	$T_{Benders} = 8.13 \times 10^{-8} \times n^{6.80}$	0.977	$T_{HUBBI} = 1.87 \times 10^{-4} \times n^{3.63}$	0.920



(a) Memory usage for different sizes of networks with $p = 2$. Y-axis is in log scale. The fitness curves for the enhanced Benders decomposition method and HUBBI are also visualized. The corresponding fitness functions are $M_{Benders} = 2.77 \times 10^{-3} \times n^{4.06}$ ($R^2_{Benders} = 0.9993$) and $M_{HUBBI} = 3.59 \times 10^{-3} \times n^{2.88}$ ($R^2_{HUBBI} = 0.9728$).



(b) The evolution of memory usage with time for TR81 dataset with $p = 5$. Both the x-axis and the y-axis are in log scale. Four dots from left to right are the time points that computation of hubbiness, network design from $p = 2$ to $p = 3$, from $p = 3$ to $p = 4$ and from $p = 4$ to $p = 5$ are complete.

Figure 10: The memory usage for different sizes of network with $p = 2$ and the evolution of memory usage with time for TR81 dataset with $p = 5$. The fixed cost $f = [1000, 1000, 1000, 1000]$ and variable cost $b = [0.10, 0.04, 0.02, 0.04]$ are used. The cost for installing a hub is $f_k^H = 10^7$.

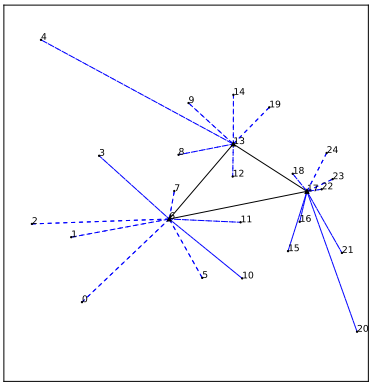
TR81 dataset with $p = 5$.

5.4. Analysis of solutions with large gaps

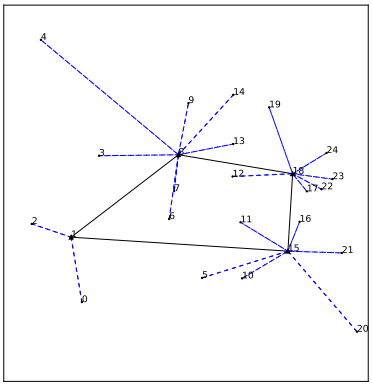
Although the solutions provided by HUBBI are close-to-optimal in many cases, it is interesting to scrutinize the solutions with large gaps further. As shown in Figure 7, there is a case for AP25 dataset with a gap of around 1.1%. This case is with $p = 4$, fixed cost $f = [1000, 1000, 1000, 1000]$ and variable cost $b = [0.10, 0.04, 0.02, 0.04]$ for AP25 dataset (see the 7th case in Table B2 in the Appendix). For further analysis, we visualize the solutions obtained by HUBBI and the optimal solutions obtained by the enhanced Benders decomposition method for AP25 dataset with $p \in \{3, 4, 5\}$ in Figure 11. There is a unidirectional hub cycle among hubs $6 \rightarrow 17 \rightarrow 13$ in the optimal solution for $p = 3$ (see Figure 11(a)). When the number of hubs increases from $p = 3$ to $p = 4$, the hub set in the optimal solution changes completely (see Figure 11(b)). The new optimal hub set becomes $[8, 1, 15, 18]$, which is not identified by HUBBI. HUBBI performs the operator Cycle-Extension instead. After the old hub 17 is replaced by new hub 22, another new hub 15 is built (see Figure 11(e)). The gap of this solution is 1.07%. Interestingly, when the number of hubs increases further to $p = 5$, the optimal solution is obtained again by HUBBI based on the poor solution for $p = 4$ (see Figure 11(f)). This emphasizes the limitations and also the strengths of incremental network design.

5.5. Results for large instances

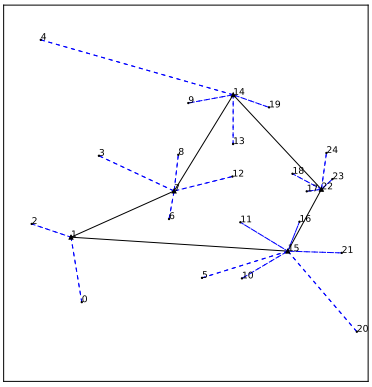
In this section, we test HUBBI on large network instances: The CAB dataset with $n = 90, 100$, as well as the AP dataset, USA423 dataset and URAND dataset with $n = 100, 200$. For all instances, the fixed cost and variable cost are set to [1000,1000,1000,1000] and [0.1,0.04,0.02,0.04], respectively. The cost for establishing a hub is set to



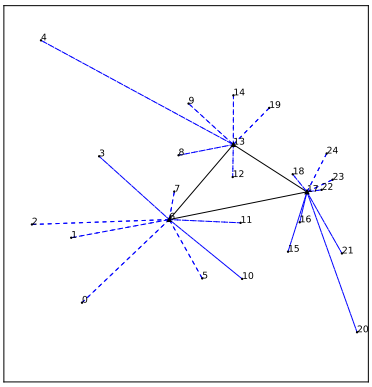
(a) The optimal solution obtained by the enhanced Benders decomposition method for $p = 3$. There is a unidirectional hub cycle among hubs $6 \rightarrow 17 \rightarrow 13$.



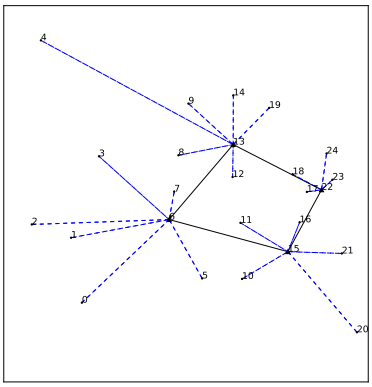
(b) The optimal solution obtained by the enhanced Benders decomposition method for $p = 4$. There is a unidirectional hub cycle among hubs $8 \rightarrow 1 \rightarrow 15 \rightarrow 18$.



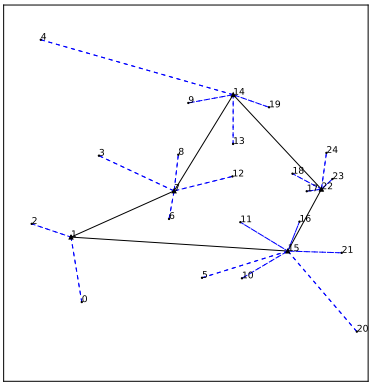
(c) The optimal solution obtained by the enhanced Benders decomposition method for $p = 5$. There is a unidirectional hub cycle among hubs $7 \rightarrow 1 \rightarrow 15 \rightarrow 22 \rightarrow 14$.



(d) The solution obtained by HUBBI for $p = 3$. There is a unidirectional hub cycle among hubs $6 \rightarrow 17 \rightarrow 13$. This solution is the optimal.



(e) The solution obtained by HUBBI for $p = 4$. There is a unidirectional hub cycle among hubs $6 \rightarrow 15 \rightarrow 22 \rightarrow 13$. The gap is 1.07%.



(f) The solution obtained by HUBBI for $p = 5$. There is a unidirectional hub cycle among hubs $7 \rightarrow 1 \rightarrow 15 \rightarrow 22 \rightarrow 14$. This solution is the optimal.

Figure 11: The visualization for the solutions for AP25 with $p = 3, 4, 5$ with fixed cost $f = [1000, 1000, 1000, 1000]$, variable cost $b = [0.10, 0.04, 0.02, 0.04]$ and cost for installing a hub $f_k^H = 10^7$. The solutions of the enhanced Benders decomposition method are shown on the top (sub-figures (a)–(c)) and solutions of HUBBI are shown at the bottom (sub-figures (d)–(f)). The gap of solution obtained by HUBBI for $p = 4$ is 1.07%. All other solutions are optimal.

10^7 and the number of hubs is set to $p \in \{2, 3, 4, 5\}$. The enhanced Benders decomposition method cannot provide any feasible solution within acceptable run time for these instances. Therefore, we only have the solutions that are obtained by HUBBI here. The results are presented in Table 7. First, note that HUBBI provides solutions for the largest instance (200 nodes) within 11 hours. Furthermore, we report the total cost for establishing four types of links and total cost for transporting flows, as shown in columns "Total cost for links" and "Total cost for flows", respectively. In the column "Hub network" of Table 7, "Bi-directional connection" means that all hub links are bi-directionally connected and "Unidirectional cycle" means that the hub network is constructed by one or more cycles with end-to-end unidirectional hub links. In the AP dataset and URAND dataset, the hub cycles with end-to-end unidirectional hub links are constructed in order to reduce the total cost (especially cost for establishing links, see the column "Number of hub links") even if this network structure will increase the cost for transporting flows. We will discuss this observation in the next section. The sets of hubs are shown in the column "Hubs". One interesting thing is that after adding one hubs, the location of more hubs is changed when the hub network is changed from "Bi-directional connection" to "Unidirectional cycle".

Table 7: Results for large instances as obtained by HUBBI. The fixed cost and variable cost are set to [1000,1000,1000,1000] and [0.1,0.04,0.02,0.04], respectively. The cost for establishing a hub is 10^7. The total cost for establishing four types of links is shown in column "Total cost for links". The total cost for transporting flows is shown in column "Total cost for flows". In the column "Hub network", "Bi-directional connection" means that all hub links in the hub bi-directionally connected and "Unidirectional cycle" means that the hub network is constructed by one or more cycles with end-to-end unidirectional hub links.

Datasets	Size	p	Solution of HUBBI	Time of HUBBI	Total cost for links	Total cost for flows	Hubs	Number of hub links	Hub network
CAB	90	2	699,137,666	156.6	100,108,008	579,029,658	21 50	2	Bi-directional connection
CAB	90	3	622,124,994	663.0	89,175,086	502,949,908	45 50 64	4	Bi-directional connection
CAB	90	4	579,909,174	1312.9	74,962,888	464,946,286	12 20 50 64	10	Bi-directional connection
CAB	90	5	545,619,347	2320.7	67,991,924	427,627,423	4 20 27 50 64	14	Bi-directional connection
CAB	100	2	815,226,014	245.1	113,405,292	681,820,722	28 50	2	Bi-directional connection
CAB	100	3	721,506,754	1040.3	93,189,096	598,317,658	3 35 50	4	Bi-directional connection
CAB	100	4	669,541,155	1945.4	87,763,170	541,777,985	3 4 20 50	10	Bi-directional connection
CAB	100	5	628,578,120	3459.0	74,802,728	503,775,392	3 4 20 27 50	14	Bi-directional connection
AP	100	2	2,372,688,531	94.6	2,349,371,148	3,317,383	33 67	2	Bi-directional connection
AP	100	3	2,012,898,792	1011.3	1,979,473,016	3,425,777	5 36 67	3	Unidirectional cycle
AP	100	4	1,761,856,862	2152.0	1,718,195,367	3,661,496	6 36 43 91	4	Unidirectional cycle
AP	100	5	1,622,617,647	2577.2	1,568,528,552	4,089,095	6 16 43 53 91	5	Unidirectional cycle
AP	200	2	4,751,446,007	1253.3	4,728,160,697	3,285,310	44 139	2	Bi-directional connection
AP	200	3	3,989,552,242	16414.3	3,956,098,772	3,453,470	36 82 157	3	Unidirectional cycle
AP	200	4	3,510,839,590	31905.8	3,467,232,472	3,607,118	29 74 117 149	4	Unidirectional cycle
AP	200	5	3,208,883,388	38924.7	3,154,774,305	4,109,083	23 29 95 117 160	5	Unidirectional cycle
USA	100	2	589,646,094	162.4	116,878,144	452,767,950	77 89	2	Bi-directional connection
USA	100	3	501,668,349	1129.9	110,831,724	360,836,625	25 62 89	6	Bi-directional connection
USA	100	4	479,357,476	2385.2	106,030,428	333,327,048	25 51 62 89	8	Bi-directional connection
USA	100	5	455,307,390	3730.3	101,003,562	304,303,828	25 33 62 83 89	14	Bi-directional connection
USA	200	2	2,684,903,589	4065.0	293,821,689	2,371,081,900	77 100	2	Bi-directional connection
USA	200	3	2,384,644,807	16735.7	219,574,120	2,135,070,687	77 106 107	6	Bi-directional connection
USA	200	4	2,129,545,203	26729.2	209,055,096	1,880,490,107	25 106 107 174	12	Bi-directional connection
USA	200	5	1,970,547,697	37587.8	193,491,304	1,727,056,393	25 62 101 106 107	18	Bi-directional connection
URAND	100	2	5,905,819,865	115.3	5,884,398,134	1,421,732	4 73	2	Bi-directional connection
URAND	100	3	4,826,131,562	1145.6	4,794,632,720	1,498,843	31 64 86	3	Unidirectional cycle
URAND	100	4	4,145,901,880	2713.7	4,104,310,083	1,591,797	0 31 65 68	4	Unidirectional cycle
URAND	100	5	3,536,241,325	3634.6	3,484,674,459	1,566,866	0 52 65 84 89	5	Unidirectional cycle
URAND	200	2	11,628,555,837	1204.1	11,602,817,192	5,738,645	28 32	2	Bi-directional connection
URAND	200	3	9,418,531,756	17107.9	9,382,630,909	5,900,847	67 110 183	3	Unidirectional cycle
URAND	200	4	8,053,464,679	34686.0	8,007,284,267	6,180,412	20 84 109 110	4	Unidirectional cycle
URAND	200	5	7,063,015,577	37943.4	7,006,791,208	6,224,369	65 79 84 142 182	5	Unidirectional cycle

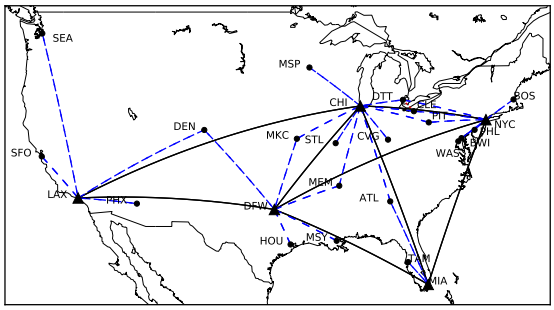
5.6. Interdependence between the hub network topology and cost settings

If the total cost for establishing (hub) links is too expensive, compared to the total variable cost, the final hub network will preferably be a cycle with end-to-end unidirectional hub links. The ratio between total variable cost and total fixed cost for links depends on not only the travel demands w, but also the choice of f and b. In the AP dataset and URAND dataset in Section 5.5, the travel demands of the AP dataset are rather small, although the other datasets have the same cost setting. Therefore, the hub networks in the AP dataset become unidirectional cycles for the cases with $p > 2$. In addition, the settings of fixed cost for establishing hubs also influence the topology of hub networks via the location of hubs.

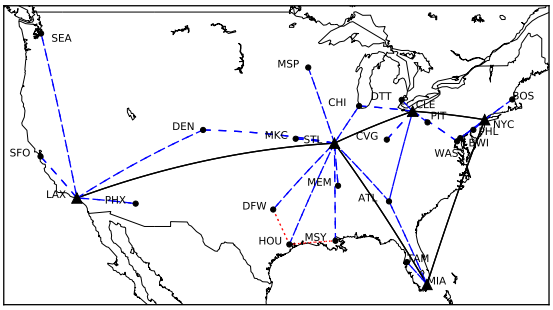
We discuss the correlation between the topology of hub networks and the cost settings based on Figure 12: We set the fixed cost for establishing hub links to different values $f^2 \in \{100, 1000, 10000, 100000\}$ and the fixed cost for establishing hubs with two alternatives: $f_k^H \in \{10^7, F_k^d\}$; other parameters remain unchanged ($f = [1000, 1000, f^2, 1000]$, $b = [0.10, 0.04, 0.02, 0.04]$) for the case with $p = 5$ on the CAB25 dataset. The parameter F_k^d was proposed by Ebery et al. (2000) as follows:

$$F_k^d = \left(1 - \frac{3c_{kh}}{5 \max_{i \in V} c_{ih}}\right)\left(\sum_{i \in V, j \in V} (b^1 c_{ih} + b^3 c_{hj} - b^2 c_{ij})\right)/p \tag{13}$$

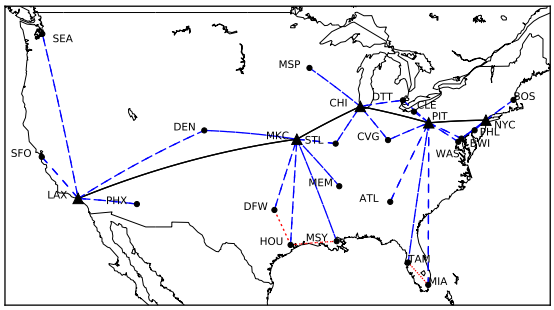
where node h is the node that is the closest to the center of mass. The obtained fixed cost for establishing hubs is



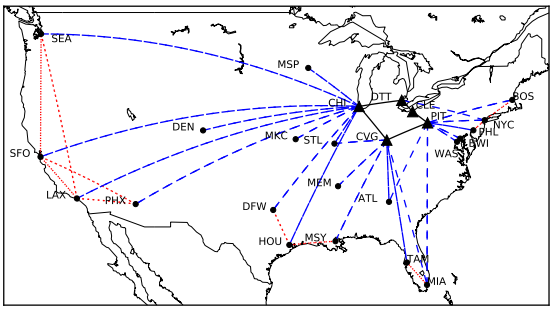
(a) Solution for $f = [1000, 1000, 100, 1000]$ and $f_k^H = 10^7$.



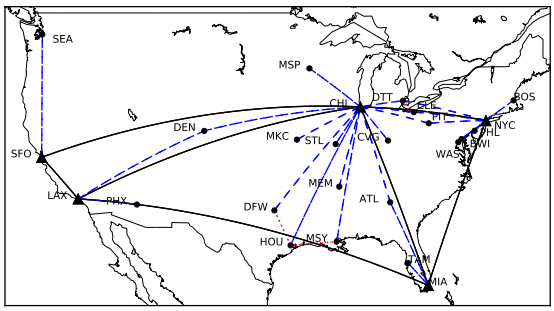
(b) Solution for $f = [1000, 1000, 1000, 1000]$ and $f_k^H = 10^7$.



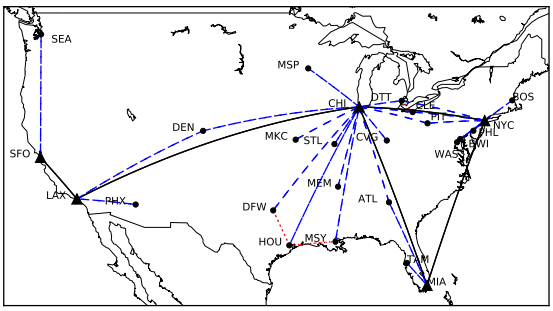
(c) Solution for $f = [1000, 1000, 10000, 1000]$ and $f_k^H = 10^7$.



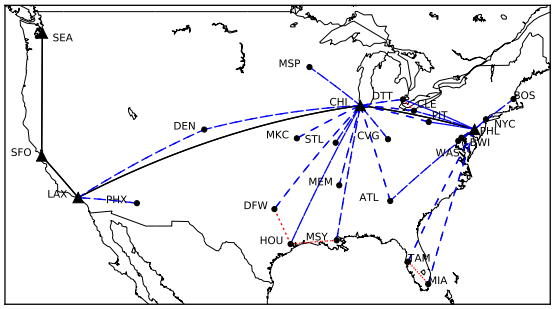
(d) Solution for $f = [1000, 1000, 100000, 1000]$ and $f_k^H = 10^7$.



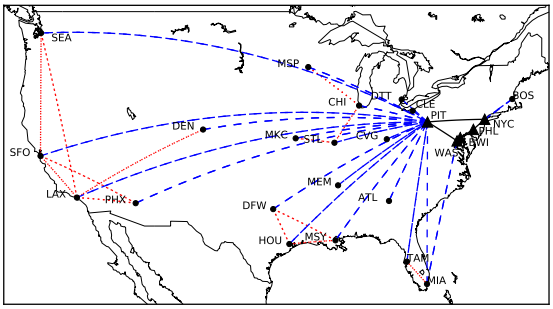
(e) Solution for $f = [1000, 1000, 100, 1000]$ and $f_k^H = F_k^d$.



(f) Solution for $f = [1000, 1000, 1000, 1000]$ and $f_k^H = F_k^d$.



(g) Solution for $f = [1000, 1000, 10000, 1000]$ and $f_k^H = F_k^d$.



(h) Solution for $f = [1000, 1000, 100000, 1000]$ and $f_k^H = F_k^d$.

Figure 12: Eight instances of incomplete hub location problems for the case with $p = 5$ on the CAB25 dataset with different values of fixed costs for establishing hub links (i.e., f^2), leaving other parameters unchanged: (a) With the smallest value of f^2, all hub links are bi-directionally and densely connected; (b) With increased value of f^2, the hub links are sparser but still bi-directionally connected; (c) With further increased value of f^2, the hub links are much sparser and the hub network becomes a line with bi-directional links; (d) With the largest value of f^2, a unidirectional cycle is established, i.e., there are only five hub links, *DTT-CHI, CHI-CVG, CVG-PIT, PIT-CLE, CLE-DTT*, in the whole network. In subfigures (e–h), the effect of using F_k^d is visible; nodes with lower fixed costs are preferred.

distributed in the range $[3.11 \times 10^7, 7.78 \times 10^7]$.

In the first four subfigures of Figure 12, the fixed hub cost is constant at 10^7. In Figure 12(a), with the smallest value of f^2, all hub links are bi-directional and densely connected. In this case, the operator Tree-Extension contributes mainly to the improvement of the solutions. In Figure 12(b), with increased value of f^2, the hub links are sparser but still bi-directional connected. In Figure 12(c), with further increased value of f^2, the hub links are much sparser and the hub network becomes a line with bi-directional links. In Figure 12(d), with the largest value of f^2, a unidirectional cycle is established, i.e., there are only five hub links, *DTT-CHI, CHI-CVG, CVG-PIT, PIT-CLE, CLE-DTT*, in the whole network. In this case, the operator Cycle-Extension is preferred. When the fixed cost of hubs is computed as F_k^d, nodes with lower values of F_k^d preferably become hubs (see Figures 12(e–h)).

6. Conclusions

In this paper, we proposed a heuristic-based method, called HUBBI. This method first computes a ranking of possible hub pairs in the network, the so-called hubbiness. Then it uses two network design patterns (Tree-Extension and Cycle-extension) and VNS to generate high-quality solutions for incomplete hub location problems with short computation times. In our computational results, five datasets (CAB, AP, TR, USA423 and URAND) with variable sizes were used as case studies. Experimental results showed that our HUBBI provides optimal solutions for over 90% of all instances. The gaps of solutions are all less than 1% except for one instance. The run time of our method increases with the order of n^4 while the run time for an exact enhanced Benders decomposition algorithm increases with n^6, where n is the number of nodes. Moreover, HUBBI only uses about 1% of the memory of the enhanced Benders decomposition method, which makes it available for solving larger networks with limited computing resources. We also discussed the factors that affect the topology of hub network, finding that cycles with unidirectional hub links are more likely to appear with either larger values of fixed cost for establishing hub links, or with lower travel demands in the network.

Since HUBBI constructs networks by adding hubs iteratively, its computation time increases with the size of p. In addition to the p-hub constraint, another type of model with hop-constraints to limit the maximum number of links or hops used by the path of each OD pair might improve the service level of a network (de Camargo et al., 2017). Different from the p-hub constraint, the hop-constraints do not limit the establishment of hubs or links directly. Thus, the implementation of HUBBI together with variable neighborhood search for hop-constraints might be quite different. The constraint on costs for infrastructure networks can also be studied in future work. For a given network, the cost for constructing the infrastructure (hubs and links) are limited. The objective function could be to minimize the total variable cost under this constraint.

Acknowledgement

This study is supported by the Research Fund from National Natural Science Foundation of China (Grants No. 61601013, No. 61650110516, No. 61521091, and No. 91538204).

References

Abyazi-Sani, R., Ghanbari, R., 2016. An efficient tabu search for solving the uncapacitated single allocation hub location problem. Computers & Industrial Engineering 93, 99 – 109. URL: http://www.sciencedirect.com/science/article/pii/S0360835215005082, doi:https://doi.org/10.1016/j.cie.2015.12.028.

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

Alumur, S.A., Kara, B.Y., Karasan, O.E., 2009. The design of single allocation incomplete hub networks. Transportation Research Part B: Methodological 43, 936–951.

Alumur, S.A., Kara, B.Y., Karasan, O.E., 2012. Multimodal hub location and hub network design. Omega 40, 927–939.

Azizi, N., Chauhan, S., Salhi, S., Vidyarthi, N., 2016. The impact of hub failure in hub-and-spoke networks: Mathematical formulations and solution techniques. Computers & Operations Research 65, 174–188.

Calık, H., Alumur, S.A., Kara, B.Y., Karasan, O.E., 2009. A tabu-search based heuristic for the hub covering problem over incomplete hub networks. Computers & Operations Research 36, 3088–3096.

de Camargo, R.S., Miranda, G., 2012. Single allocation hub location problem under congestion: Network owner and user perspectives. Expert Systems with Applications 39, 3385–3391.

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

de Camargo, R.S., Miranda, G.d., Luna, H., 2008. Benders decomposition for the uncapacitated multiple allocation hub location problem. Computers & Operations Research 35, 1047–1064.

de Camargo, R.S., de Miranda Jr, G., O'Kelly, M.E., 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.

Campbell, J.F., 1994. Integer programming formulations of discrete hub location problems. European Journal of Operational Research 72, 387–405.

Campbell, J.F., De Miranda, G., De Camargo, R.S., O'Kelly, M.E., 2015. Hub location and network design with fixed and variable costs, in: System Sciences (HICSS), 2015 48th Hawaii International Conference on, IEEE. pp. 1059–1067.

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

Çetiner, S., 2003. An iterative hub location and routing problem for postal delivery systems. master's thesis, department of industrial engineering. Middle East Technical University, Ankara, Turkey .

Chen, J.F., 2007. A hybrid heuristic for the uncapacitated single allocation hub location problem. Omega 35, 211–220.

Contreras, I., Díaz, J.A., Fernández, E., 2009a. Lagrangean relaxation for the capacitated hub location problem with single assignment. OR spectrum 31, 483–505.

Contreras, I., Díaz, J.A., Fernández, E., 2011. Branch and price for large-scale capacitated hub location problems with single assignment. INFORMS Journal on Computing 23, 41–55.

Contreras, I., Fernández, E., Marín, A., 2009b. Tight bounds from a path based formulation for the tree of hub location problem. Computers & Operations Research 36, 3117–3127.

Contreras, I., Fernández, E., Marín, A., 2010. The tree of hubs location problem. European Journal of Operational Research 202, 390–400.

Contreras, I., Tanash, M., Vidyarthi, N., 2016. Exact and heuristic approaches for the cycle hub location problem. Annals of Operations Research , 1–23.

Corberán, Á., Peiró, J., Campos, V., Glover, F., Martí, R., 2016. Strategic oscillation for the capacitated hub location problem with modular links. Journal of Heuristics 22, 221–244. URL: https://doi.org/10.1007/s10732-016-9308-7, doi:10.1007/s10732-016-9308-7.

Ebery, J., Krishnamoorthy, M., Ernst, A., Boland, N., 2000. The capacitated multiple allocation hub location problem: Formulations and algorithms. European Journal of Operational Research 120, 614 – 631. URL: http://www.sciencedirect.com/science/article/pii/S0377221798003956, doi:https://doi.org/10.1016/S0377-2217(98)00395-6.

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

Figueiredo, R.M.A., O'Kelly, M.E., Pizzolato, N.D., 2014. A two-stage hub location method for air transportation in Brazil. International Transactions in Operational Research 21, 275–289.

Gelareh, S., Nickel, S., 2011. Hub location problems in transportation networks. Transportation Research Part E: Logistics and Transportation Review 47, 1092–1111.

Hoff, A., Peiró, J., Corberán, Á., Martí, R., 2017. Heuristics for the capacitated modular hub location problem. Computers & Operations Research 86, 94 – 109. URL: http://www.sciencedirect.com/science/article/pii/S0305054817301144, doi:https://doi.org/10.1016/j.cor.2017.05.004.

Ilić, A., Urošević, D., Brimberg, J., Mladenović, N., 2010. A general variable neighborhood search for solving the uncapacitated single allocation p-hub median problem. European Journal of Operational Research 206, 289–300.

Kim, H., O'Kelly, M.E., 2009. Reliable p-hub location problems in telecommunication networks. Geographical Analysis 41, 283–306.

Kratica, J., Milanović, M., Stanimirović, Z., Tošić, D., 2011. An evolutionary-based approach for solving a capacitated hub location problem. Applied Soft Computing 11, 1858 – 1866. URL: http://www.sciencedirect.com/science/article/pii/S1568494610001365, doi:https://doi.org/10.1016/j.asoc.2010.05.035. the Impact of Soft Computing for the Progress of Artificial Intelligence.

Kratica, J., Stanimirović, Z., Tošić, D., Filipović, V., 2005. Genetic algorithm for solving uncapacitated multiple allocation hub location problem. 24, 415–426.

Kratica, J., Stanimirović, Z., Tošić, D., Filipović, V., 2007. Two genetic algorithms for solving the uncapacitated single allocation p-hub median problem. European Journal of Operational Research 182, 15–28.

Martí, R., Corberán, A., Peiró, J., 2015. Scatter search for an uncapacitated p-hub median problem. Computers & Operations Research 58, 53 – 66. URL: http://www.sciencedirect.com/science/article/pii/S030505481400330X, doi:https://doi.org/10.1016/j.cor.2014.12.009.

Meyer, T., Ernst, A.T., Krishnamoorthy, M., 2009. A 2-phase algorithm for solving the single allocation p-hub center problem. Comput. Oper. Res. 36, 3143–3151. URL: http://dx.doi.org/10.1016/j.cor.2008.07.011, doi:10.1016/j.cor.2008.07.011.

Mladenović, N., Hansen, P., 1997. Variable neighborhood search. Computers & Operations Research 24, 1097 – 1100. URL: http://www.sciencedirect.com/science/article/pii/S0305054897000312, doi:https://doi.org/10.1016/S0305-0548(97)00031-2.

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

O'Kelly, M.E., 1987. A quadratic integer program for the location of interacting hub facilities. European Journal of Operational Research 32, 393–404.

O'Kelly, M.E., 2012. Fuel burn and environmental implications of airline hub networks. Transportation Research Part D: Transport and Environment 17, 555–567.

O'Kelly, M.E., Campbell, J.F., Camargo, R.S., Miranda, G., 2015. Multiple allocation hub location model with fixed arc costs. Geographical Analysis 47, 73–96.

Peiró, J., Corberán, Á., Martí, R., 2014. GRASP for the uncapacitated r-allocation p-hub median problem. Computers & Operations Research 43, 50–60.

Peker, M., Kara, B.Y., Campbell, J.F., Alumur, S.A., 2016. Spatial analysis of single allocation hub location problems. Networks and Spatial Economics 16, 1075–1101. URL: https://doi.org/10.1007/s11067-015-9311-9, doi:10.1007/s11067-015-9311-9.

Pérez, M.P., Rodríguez, F.A., Vega, J.M.M., 2004. On the use of path relinking for the ρ–hub median problem, in: European Conference on

Evolutionary Computation in Combinatorial Optimization, Springer. pp. 155–164.
Randall, M., 2008. Solution approaches for the capacitated single allocation hub location problem using ant colony optimisation. Computational Optimization and Applications 39, 239–261. URL: https://doi.org/10.1007/s10589-007-9069-1, doi:10.1007/s10589-007-9069-1.
Rodríguez-Martín, I., Salazar-González, J.J., 2006. An iterated local search heuristic for a capacitated hub location problem, in: Almeida, F., Blesa Aguilera, M.J., Blum, C., Moreno Vega, J.M., Pérez Pérez, M., Roli, A., Sampels, M. (Eds.), Hybrid Metaheuristics, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 70–81.
Rodríguez-Martín, I., Salazar-González, J.J., Yaman, H., 2014. A branch-and-cut algorithm for the hub location and routing problem. Computers & Operations Research 50, 161–174.
de Sá, E.M., Contreras, I., Cordeau, J.F., 2015. Exact and heuristic algorithms for the design of hub networks with multiple lines. European Journal of Operational Research 246, 186–198.
de Sá, E.M., Morabito, R., de Camargo, R.S., 2018a. Benders decomposition applied to a robust multiple allocation incomplete hub location problem. Computers & Operations Research 89, 31–50.
de Sá, E.M., Morabito, R., de Camargo, R.S., 2018b. Efficient benders decomposition algorithms for the robust multiple allocation incomplete hub location problem with service time requirements. Expert Systems with Applications 93, 50 – 61. URL: http://www.sciencedirect.com/science/article/pii/S0957417417306802, doi:https://doi.org/10.1016/j.eswa.2017.10.005.
Serper, E.Z., Alumur, S.A., 2016. The design of capacitated intermodal hub networks with different vehicle types. Transportation Research Part B: Methodological 86, 51 – 65. URL: http://www.sciencedirect.com/science/article/pii/S0191261516000205, doi:https://doi.org/10.1016/j.trb.2016.01.011.
Silva, M.R., Cunha, C.B., 2009. New simple and efficient heuristics for the uncapacitated single allocation hub location problem. Computers & Operations Research 36, 3152 – 3165. URL: http://www.sciencedirect.com/science/article/pii/S0305054809000033, doi:https://doi.org/10.1016/j.cor.2008.12.019. new developments on hub location.
Skorin-Kapov, D., Skorin-Kapov, J., 1994. On tabu search for the location of interacting hub facilities. European Journal of Operational Research 73, 502–509.
Stanimirović, Z., 2012. A genetic algorithm approach for the capacitated single allocation p-hub median problem. Computing and Informatics 29, 117–132.
Sun, X., Dai, W., Zhang, Y., Wandelt, S., 2017. Finding p-hub median locations: An empirical study on problems and solution techniques. Journal of Advanced Transportation 2017, Article ID 9387302. doi:https://doi.org/10.1155/2017/9387302.
Todosijević, R., Urošević, D., Mladenović, N., Hanafi, S., 2017. A general variable neighborhood search for solving the uncapacitated r-allocation p-hub median problem. Optimization Letters 11, 1109–1121. URL: https://doi.org/10.1007/s11590-015-0867-6, doi:10.1007/s11590-015-0867-6.
Yaman, H., Carello, G., 2005. Solving the hub location problem with modular link capacities. Computers & Operations Research 32, 3227–3245.

Appendix A. Computation of costs and local search

This section presents a function for computing the costs of solutions (Appendix A.1). In addition, local search strategies for modifying direct links (Appendix A.2), access links (Appendix A.3) and hub links (Appendix A.4) are proposed and formalized. The function and operators are used frequently in the computation of hubbiness (Section 4.1) and network design patterns (Section 4.2), and variable neighborhood search (Section 4.3).

Appendix A.1. Computation of costs

In this section, we introduce a function for computing the cost of a current solution. It provides the variable cost and the corresponding transportation path for each pair of nodes (which are needed when performing local search strategies).

Assume that the sets of hub nodes, hub links and direct links are represented by H, $Link_0$ and $Link_2$, respectively. The variable cost per unit flow $Cost^o_{km}$ between each pair of hubs *(k,m)* is obtained by Dijkstra algorithm with computational complexity of $O(t + plog(p))$, where *p, t* are the numbers of hubs and hub links, respectively. The current sets of hubs that are connected to spoke node i by collection links and distribution links are represented by Hub^{1o}_i and Hub^{3o}_i. Based on the variable cost between hub nodes, the variable costs between hub node k and spoke node i are computed as follows:

$$Cost^o_{ik} = \min\left\{b^1 c_{im} + Cost^o_{mk} \mid m \in Hub^{1o}_i\right\}$$
$$Cost^o_{ki} = \min\left\{Cost^o_{km} + b^3 c_{mi} \mid m \in Hub^{3o}_i\right\}$$

Thus, the current minimum variable cost from spoke node i to spoke node j is obtained by:

$$Cost^o_{ij} = \min\left\{b^1 c_{ik} + Cost^o_{kj} \mid k \in Hub^{1o}_i\right\}$$

If nodes *i,j* are connected by a direct link, then we have $Cost^o_{ij} = b^0 c_{ij}$. The computational complexity for all variable costs $Cost^o_{ij}, (i, j \in V, i \neq j)$ is $O(n^2 p)$. Assume that $Path^o_{ij} = [i, k_1, k_2, ..., k_s, j]$ is the corresponding path between i and j. If the direct link between *(i,j)* is established, then $Path^o_{ij} = [i, j]$. The total cost TC is computed as follows:

$$TC = \sum_{i \in V, j \in V, i \neq j} Cost^o_{ij} w_{ij} + \text{Fixed cost for establishing hubs and links}$$

Appendix A.2. Modifying direct links

The first local search strategy only modifies the connection of direct links between spoke nodes. We try to **remove** or **add** the direct link between each pair of spoke nodes, leaving access links, hub links and hub location unchanged. Assume that the set of direct links are represented by $Link_0$. Before the modification, we sort all direct links by increasing travel demands and spoke pairs without direct links by decreasing travel demands, respectively. The intuition is that pairs of spoke nodes with larger travel demands are more likely to be connected by direct links. For a directed link *(i,j)*, assume that we remove the direct link between them. After the removal, a path through the hub network is used to satisfy the travel demands from node i to node j. The new cost is computed as follows:

$$Cost^c_{ij} = \min \left\{ b^1 c_{ik} + Cost^o_{kj} \mid k \in Hub^{1o}_i \right\}$$

Therefore, the difference of the total cost for removing the direct link is:

$$\begin{aligned} dC^{ij}_{rem} &= \left(Cost^c_{ij} - Cost^o_{ij} \right) w_{ij} - (f^0 + A_{ij}) c_{ij} \\ &= \left(\min \left\{ b^1 c_{ik} + Cost^o_{kj} \mid k \in Hub^{1o}_i \right\} - b^0 c_{ij} \right) w_{ij} - (f^0 + A_{ij}) c_{ij} \end{aligned}$$

If $dC^{ij}_{rem} < 0$, we update the link connection $Link_0$, variable cost $Cost^c_{ij}$ and used path $Path^c_{ij}$.

Algorithm 6 Local search for modifying direct links, Operator **Removing**

Input: A network instance with the set of nodes V, current total cost Obj^o, variable costs $Cost^o_{ij}$, transportation paths $Path^o_{ij}$ $(i, j \in V)$
Output: The final total cost Obj, updated variable costs $Cost_{ij}$ and transportation paths $Path_{ij}$ $(i, j \in V)$.
Let Obj=Obj^o, $Cost_{ij}$=$Cost^o_{ij}$, $Path_{ij}$=$Path^o_{ij}$.
Let DL be the list of direct links sorted by increasing travel demands.
for each $link \in DL$ **do**
    Simulate removing *link* and compute the new total cost Obj^o and new $Cost^o_{ij}$ and $Path^o_{ij}$.
    if $Obj^o < Obj$ **then**
        Remove *link*. Let Obj=Obj^o, $Cost_{ij}$=$Cost^o_{ij}$, $Path_{ij}$=$Path^o_{ij}$ $(i, j \in V)$.
    end if
end for all

Algorithm 7 Local search for modifying direct links, Operator **Adding**

Input: A network instance with the set of nodes V, current total cost Obj^o, variable costs $Cost^o_{ij}$, transportation paths $Path^o_{ij}$ $(i, j \in V)$
Output: The final total cost Obj, updated variable costs $Cost_{ij}$ and transportation paths $Path_{ij}$ $(i, j \in V)$.
Let Obj=Obj^o, $Cost_{ij}$=$Cost^o_{ij}$, $Path_{ij}$=$Path^o_{ij}$.
Let DN be the list of all pairs of spoke nodes that are not directly connected sorted by decreasing travel demands.
for each $link \in DN$ **do**
    Simulate adding *link* and compute the new total cost Obj^o and new $Cost^o_{ij}$ and $Path^o_{ij}$.
    if $Obj^o < Obj$ **then**
        Add *link*. Let Obj=Obj^o, $Cost_{ij}$=$Cost^o_{ij}$, $Path_{ij}$=$Path^o_{ij}$ $(i, j \in V)$.
    end if
end for all

For a spoke pair *(i,j)* without direct link, the difference of the total cost for adding the direct link is computed as follows:

$$dC_{add}^{ij} = \left(b^0 c_{ij} - Cost_{ij}^o\right) w_{ij} + (f^0 + A_{ij}) c_{ij}$$

If $dC_{add}^{ij} < 0$, we update the link connection $Link_0$, variable costs $Cost_{ij}^c$ and used path $Path_{ij}^c$. The pseudo-code for removing/adding direct links is shown in Algorithm 6–7.

Property 1. *The computational complexity of one iteration for removing/adding direct links is* $O(n^2 p)$.

Appendix A.3. Modifying access links

The second local search strategy modifies the connections of access links including collection links and distribution links. We only explain the operator on collection links, since the distribution links are handled identically. For each spoke node i, let the sets of hubs that are connected to spoke node i by collection links and distribution links be denoted by Hub_i^{1o} and Hub_i^{3o}, respectively. The hubs in Hub_i^{1o} and Hub_i^{3o} are ordered by decreasing distances to spoke node i (because, in general, spoke nodes are less likely to be assigned to farther hubs). For each collection hub node $h_o \in Hub_i^{1o}$ and each alternative hub $h_c \in H \backslash Hub_i^{1o}$ sequentially, three operators, **"removing"**, **"adding"** and **"replacing"**, can be performed.

In the 'removing' operator, a collection link from spoke i to hub h_o is removed. For a destination node j, if the path Pah_{ij}^o uses the link $i - h_o$, we need to find a new path through the remaining hubs in Hub_i^{1o}. Its cost per unit flow is:

$$Cost_{ij}^c = \min\left\{b^1 c_{ik} + Cost_{kj}^o \mid k \in Hub_i^{1o} \backslash \{h_o\}\right\}$$

Thus, the difference of the total cost for removing the collection link $i - h_o$ is:

$$\begin{aligned} dC_{rem}^{ih_o} &= \sum_{j \in V, j \neq i} \left(Cost_{ij}^c - Cost_{ij}^o\right) w_{ij} - (f^1 + A_{ih_o}) c_{ih_o} \\ &= \sum_{j \in V, j \neq i} \left(\min\left\{b^1 c_{ik} + Cost_{kj}^o \mid k \in Hub_i^{1o} \backslash \{h_o\}\right\} - Cost_{ij}^o\right) w_{ij} - (f^1 + A_{ih_o}) c_{ih_o} \end{aligned}$$

In the 'adding' operator, a collection link from spoke i to hub h_c is established. The difference of total cost for installing the collection link $i - h_c$ is:

$$dC_{add}^{ih_c} = \sum_{j \in V, j \neq i} \min\left\{b^1 c_{ih_c} + Cost_{h_c j}^o - Cost_{ij}^o,\ 0\right\} w_{ij} + (f^1 + A_{ih_c}) c_{ih_c}$$

Finally, regarding the 'replacing' operator, the difference of total cost for replacing the collection link $i - h_o$ with $i - h_c$ is computed as follows:

$$dC_{rep}^{ih_o h_c} = \sum_{j \in V, j \neq i} \left(\min\left\{b^1 c_{ik} + Cost_{kj}^o \mid k \in \{h_c\} \cup Hub_i^{1o} \backslash \{h_o\}\right\} - Cost_{ij}^o\right) w_{ij} - (f^1 + A_{ih_o}) c_{ih_o} + (f^1 + A_{ih_c}) c_{ih_c}$$

For each spoke node i and its collection hub node h_o, we select the best operator and alternate hub node h_c with the minimum new total cost (if the 'removing' operator is selected, then $h_c = \emptyset$). If the current solution is improved, the set of hubs Hub_i^1, variable costs $Cost_{ij}^o$ and paths $Path_{ij}^o$ for $j \in V$ are updated. The pseudo-code for removing/adding/replacing access links is shown in Algorithm 8–10.

Property 2. *The computational complexity of one iteration for removing/adding/replacing access links are* $O(n^2 p^2)$, $O(n^2 p)$ *and* $O(n^2 p^3)$, *respectively.*

Algorithm 8 Local search for modifying access links, Operator **Removing**

Input: A network instance with n nodes, the set of nodes V, current total cost Obj^o, variable costs $Cost^o_{ij}$ and transportation paths $Path^o_{ij}$ $(i, j \in V)$.
Output: The final total cost Obj, updated variable cost $Cost_{ij}$ and transportation paths $Path_{ij}$ $(i, j \in V)$.
Let Obj=Obj^o, $Cost_{ij}$=$Cost^o_{ij}$, $Path_{ij}$=$Path^o_{ij}$.
for each spoke node $i \in V$ **do**
  Sort the hubs that are connected to node i as a list H^c_i with the order of decreasing distances
  for each $h_1 \in H^c_i$ **do**
    Simulate removing the access link between h_1 and i. Compute the new total cost Obj^o and new $Cost^o_{ij}$ and $Path^o_{ij}$.
    if Obj^o <Obj **then**
      Remove the access link. Let Obj=Obj^o, $Cost_{ij}$=$Cost^o_{ij}$, $Path_{ij}$=$Path^o_{ij}$ $(i, j \in V)$.
    end if
  end for all
end for all

Algorithm 9 Local search for modifying access links, Operator **Adding**

Input: A network instance with n nodes, the set of nodes V, current total cost Obj^o, variable costs $Cost^o_{ij}$ and transportation paths $Path^o_{ij}$ $(i, j \in V)$.
Output: The final total cost Obj, updated variable cost $Cost_{ij}$ and transportation paths $Path_{ij}$ $(i, j \in V)$.
Let Obj=Obj^o, $Cost_{ij}$=$Cost^o_{ij}$, $Path_{ij}$=$Path^o_{ij}$.
for each spoke node $i \in V$ **do**
  Sort the hubs that are not connected to node i as a list H^n_i with the order of increasing distances
  for each $h_1 \in H^n_i$ **do**
    Simulate adding the access link between h_1 and i. Compute the new total cost Obj^o and new $Cost^o_{ij}$ and $Path^o_{ij}$.
    if Obj^o <Obj **then**
      Let Obj=Obj^o, $Cost_{ij}$=$Cost^o_{ij}$, $Path_{ij}$=$Path^o_{ij}$ $(i, j \in V)$.
    end if
  end for all
end for all

Algorithm 10 Local search for modifying access links, Operator **Replacing**

Input: A network instance with n nodes, the set of nodes V, current total cost Obj^o, variable costs $Cost^o_{ij}$ and transportation paths $Path^o_{ij}$ $(i, j \in V)$.
Output: The final total cost Obj, updated variable costs $Cost_{ij}$ and transportation paths $Path_{ij}$ $(i, j \in V)$.
Let Obj=Obj^o, $Cost_{ij}$=$Cost^o_{ij}$, $Path_{ij}$=$Path^o_{ij}$.
for each spoke node $i \in V$ **do**
  Sort the hubs that are connected to node i as a list H^c_i with the order of decreasing distances
  Sort the hubs that are not connected to node i as a list H^n_i with the order of increasing distances
  for each $h_1 \in H^c_i$ **do**
    Let $Obj^c = inf$.
    for each $h_2 \in H^n_i$ **do**
      Simulate adding the access link between h_2 and i and removing the access link between h_1 and i. Compute the new total cost Obj^o and new $Cost^o_{ij}$ and $Path^o_{ij}$.
      if Obj^o <Obj^c **then**
        Let $\hat{h} = h_2$.. Let Obj^c=Obj^o, $Cost^c_{ij}$=$Cost^o_{ij}$, $Path^c_{ij}$=$Path^o_{ij}$ $(i, j \in V)$.
      end if
    end for all
    if Obj^c <Obj **then**
      Add the access link between $\hat{h}$ and i and remove the access link between h_1 and i. Let Obj=Obj^c, $Cost_{ij}$=$Cost^c_{ij}$, $Path_{ij}$=$Path^c_{ij}$ $(i, j \in V)$.
    end if
  end for all
end for all

Appendix A.4. Modifying hub links

The third local search strategy modifies the connection of hub links and this strategy is only used in variable neighborhood search. We sort all connected hub links by decreasing length and all unconnected hub pairs by increasing length, respectively. Given that we do not have a hop-constraint, longer hub links are added only under rare

Algorithm 11 Local search for modifying hub links, Operator **Removing**

Input: A network instance with the set of nodes V, current total cost Obj^o, variable costs $Cost^o_{ij}$, transportation paths $Path^o_{ij}$ $(i, j \in V)$
Output: The final total cost Obj, updated variable costs $Cost_{ij}$ and transportation paths $Path_{ij}$ $(i, j \in V)$.
Let Obj=Obj^o, $Cost_{ij}$=$Cost^o_{ij}$, $Path_{ij}$=$Path^o_{ij}$.
Let HL be the list of hub links sorted by decreasing length.
for each $link \in HL$ **do**
    Simulate removing $link$ and compute the new total cost Obj^o and new $Cost^o_{ij}$ and $Path^o_{ij}$.
    if Obj^o <Obj **then**
        Remove $link$. Let Obj=Obj^o, $Cost_{ij}$=$Cost^o_{ij}$, $Path_{ij}$=$Path^o_{ij}$ $(i, j \in V)$.
    end if
end for all

Algorithm 12 Local search for modifying hub links, Operator **Adding**

Input: A network instance with the set of nodes V, current total cost Obj^o, variable costs $Cost^o_{ij}$, transportation paths $Path^o_{ij}$ $(i, j \in V)$
Output: The final total cost Obj, updated variable costs $Cost_{ij}$ and transportation paths $Path_{ij}$ $(i, j \in V)$.
Let Obj=Obj^o, $Cost_{ij}$=$Cost^o_{ij}$, $Path_{ij}$=$Path^o_{ij}$.
Let HN be the list of all pairs of hub nodes that are not connected sorted by increasing length.
for each $link \in HN$ **do**
    Simulate adding $link$ and compute the new total cost Obj^o and new $Cost^o_{ij}$ and $Path^o_{ij}$.
    if Obj^o <Obj **then**
        Add $link$. Let Obj=Obj^o, $Cost_{ij}$=$Cost^o_{ij}$, $Path_{ij}$=$Path^o_{ij}$ $(i, j \in V)$.
    end if
end for all

Algorithm 13 Local search for modifying hub links, Operator **Replacing**

Input: A network instance with the set of nodes V, current total cost Obj^o, variable costs $Cost^o_{ij}$, transportation paths $Path^o_{ij}$ $(i, j \in V)$
Output: The final total cost Obj, updated variable costs $Cost_{ij}$ and transportation paths $Path_{ij}$ $(i, j \in V)$.
Let Obj=Obj^o, $Cost_{ij}$=$Cost^o_{ij}$, $Path_{ij}$=$Path^o_{ij}$.
Let HL be the list of all hub links sorted by decreasing length.
Let HN be the list of all pairs of hub nodes that are not connected sorted by increasing length.
for each $link_1 \in HL$ **do**
    Let $Obj^c = inf$
    for each $link_2 \in HN$ **do**
        Simulate removing $link_1$, adding $link_2$ and compute the new total cost Obj^o and new $Cost^o_{ij}$ and $Path^o_{ij}$.
        if Obj^o <Obj^c **then**
            Let $\hat{link} = link_2$. Let Obj^c=Obj^o, $Cost^c_{ij}$=$Cost^o_{ij}$, $Path^c_{ij}$=$Path^o_{ij}$ $(i, j \in V)$.
        end if
    end for all
    if Obj^c <Obj **then**
        Remove $link_1$ and add $\hat{link}$. Let Obj=Obj^c, $Cost_{ij}$=$Cost^c_{ij}$, $Path_{ij}$=$Path^c_{ij}$ $(i, j \in V)$.
    end if
end for all

circumstances, as long as multiple-hop paths of similar lengths exist in the hub network already. For each pair of hub nodes, we try to **remove**, **add** or **replace** the link between them. Different from the direct links and access links, the modification of hub links might affect any pair of origin node and destination node.

First, after removing an hub link *k-m*, the paths with the minimum costs need to be found again for the OD pairs that used hub link *k-m* with the method presented in Appendix A.1. Second, for an unconnected hub pair *(k,m)*, the difference of total cost after adding a link between them is computed as follows:

$$dC^{km}_{add} = \sum_{i,j \in V, i \neq j} \min\left\{Cost^o_{ik} + b^2 c_{km} + Cost^o_{mj} - Cost^o_{ij},\ 0\right\} w_{ij} + (f^2 + A_{ih_c})c_{km}$$

In the operator 'replacing', for each connected hub links $k_0 - m_0$ and each unconnected hub pairs (k_1, m_1), we try

to remove the link $k_0 - m_0$ and establish the link $k_1 - m_1$. If the total cost is decreased, we perform this operator. If the current solution is improved, the link connection $Link_2$, variable costs $Cost^o_{ij}$ and paths $Path^o_{ij}$ for $i, j \in V, i \neq j$ are updated. The pseudo-code for removing/adding/replacing hub links is shown in Algorithm 11–13.

Property 3. *The computational complexity of one iteration for removing/adding/replacing hub links are* $O(n^2pt)$*,* $O(n^2p^4)$ *and* $O(n^2p^3t)$*, respectively. Here* p *and* t *are the numbers of hubs and hub links.*

Appendix B. Numerical results for Section 5.3

Table B1: The comparison of solutions between the enhanced Benders decomposition method (de Camargo et al., 2017) and HUBBI on the CAB dataset.

Datasets	Size	Fixed cost & Variable cost	Number of hubs p	Solution of Benders	Time of Benders (s)	Solution of HUBBI	Time of HUBBI (s)	Gap (%)
CAB	25		2	511,711,559	238.0	511,711,559	1.1	0.00
CAB	25	[2500,3000,3500,3000] &	3	477,904,481	367.5	477,904,481	6.2	0.00
CAB	25	[0.08,0.04,0.03,0.04]	4	468,438,577	586.9	468,438,577	11.1	0.00
CAB	25		5	461,450,560	628.1	461,450,560	16.5	0.00
CAB	25		2	420,299,769	267.2	420,299,769	1.3	0.00
CAB	25	[1000, 1000, 1000, 1000] &	3	373,381,181	336.5	373,381,181	6.1	0.00
CAB	25	[0.10, 0.04, 0.02, 0.04]	4	353,596,483	441.7	353,596,483	12.5	0.00
CAB	25		5	345,296,628	561.6	346,425,661	23.8	0.33
CAB	30		2	134,092,822	470.0	134,092,822	1.7	0.00
CAB	30	[2500,3000,3500,3000] &	3	130,825,592	1265.6	130,825,592	9.6	0.00
CAB	30	[0.08,0.04,0.03,0.04]	4	130,500,123	1005.6	130,500,123	16.9	0.00
CAB	30		5	131,482,612	1246.5	131,482,612	23.4	0.00
CAB	30		2	81,648,551	418.5	81,648,551	1.8	0.00
CAB	30	[1000, 1000, 1000, 1000] &	3	84,162,120	650.6	84,162,120	9.9	0.00
CAB	30	[0.10, 0.04, 0.02, 0.04]	4	87,765,039	939.4	87,765,039	18.1	0.00
CAB	30		5	92,598,192	1321.5	92,598,192	27.3	0.00
CAB	40		2	103,388,853	2294.0	103,388,853	4.8	0.00
CAB	40	[1000, 1000, 1000, 1000] &	3	104,862,318	10804.7	104,862,318	30.2	0.00
CAB	40	[0.10, 0.04, 0.02, 0.04]	4	106,854,604	8233.4	106,854,604	53.2	0.00
CAB	40		5	110,113,426	13879.1	110,113,426	82.3	0.00
CAB	50		2	140,071,617	11301.7	140,071,617	11.7	0.00
CAB	50	[1000, 1000, 1000, 1000] &	3	137,600,147	46684.4	137,600,147	69.4	0.00
CAB	50	[0.10, 0.04, 0.02, 0.04]	4	134,873,311	39268.9	134,873,311	138.0	0.00
CAB	50		5	137,579,008	38669.4	137,579,008	233.2	0.00
CAB	60		2	273,409,717	59346.4	273,409,717	24.4	0.00
CAB	60	[1000, 1000, 1000, 1000] &	3	255,000,118	123450.1	255,000,118	155.7	0.00
CAB	60	[0.10, 0.04, 0.02, 0.04]	4	241,106,048	194266.4	241,106,048	298.7	0.00
CAB	60		5	233,530,171	84196.7	233,530,171	532.5	0.00
CAB	70		2	440,806,160	283498	440,806,160	54.7	0.00
CAB	70	[1000, 1000, 1000, 1000] &	3	398,179,132	182982.2	399,809,129	277.9	0.41
CAB	70	[0.10, 0.04, 0.02, 0.04]	4	361,973,046	253073.7	362,172,993	598.8	0.06
CAB	70		5	341,191,927	308427.4	341,191,927	955.6	0.00
CAB	80		2	538,916,337	438901.3	538,916,337	100.8	0.00
CAB	80	[1000, 1000, 1000, 1000] &	3	473,982,039	473471.3	473,982,039	423.8	0.00
CAB	80	[0.10, 0.04, 0.02, 0.04]	4	434,713,733	359767.4	434,713,733	838.1	0.00
CAB	80		5	414,202,998	613328.5	414,202,998	1370.1	0.00

Table B2: The comparison of solutions between the enhanced Benders decomposition method (de Camargo et al., 2017) and HUBBI on the AP dataset.

Datasets	Size	Fixed cost & Variable cost	Number of hubs p	Solution of Benders	Time of Benders (s)	Solution of HUBBI	Time of HUBBI (s)	Gap (%)
AP	25		2	1,760,477,796	203.9	1,760,477,796	1.4	0.00
AP	25	[2500,3000,3500,3000] &	3	1,540,035,763	319.9	1,540,035,763	7.9	0.00
AP	25	[0.08,0.04,0.03,0.04]	4	1,396,507,965	377.2	1,403,077,812	13.7	0.47
AP	25		5	1,292,015,281	444.7	1,292,015,281	16.2	0.00
AP	25		2	596,879,564	199.8	596,879,564	1.6	0.00
AP	25	[1000, 1000, 1000, 1000] &	3	528,515,107	359.7	528,515,107	7.9	0.00
AP	25	[0.10, 0.04, 0.02, 0.04]	4	482,733,233	405.4	487,899,274	13.7	1.07
AP	25		5	452,974,383	499.9	452,974,383	17.4	0.00
AP	30		2	2,161,139,148	511.1	2,161,139,148	2.5	0.00
AP	30	[2500,3000,3500,3000] &	3	1,866,724,032	872.7	1,866,724,032	12.3	0.00
AP	30	[0.08,0.04,0.03,0.04]	4	1,699,507,758	1277.8	1,703,343,083	24.8	0.23
AP	30		5	1,593,740,162	1602.4	1,593,740,162	27.6	0.00
AP	30		2	730,896,391	506.8	730,896,391	2.7	0.00
AP	30	[1000, 1000, 1000, 1000] &	3	637,268,637	811.8	637,268,637	12.3	0.00
AP	30	[0.10, 0.04, 0.02, 0.04]	4	584,920,385	1182.3	585,281,722	24.6	0.06
AP	30		5	554,411,017	1752.1	554,411,017	27.9	0.00
AP	40		2	950,762,322	2974.0	950,762,322	5.3	0.00
AP	40	[1000, 1000, 1000, 1000] &	3	827,786,171	6985.3	827,786,171	31.9	0.00
AP	40	[0.10, 0.04, 0.02, 0.04]	4	744,814,010	9342.3	744,814,010	65.7	0.00
AP	40		5	701,658,008	12769.9	705,521,151	74.6	0.55
AP	50		2	1,176,312,740	13520.4	1,176,312,740	9.2	0.00
AP	50	[1000, 1000, 1000, 1000] &	3	1,016,706,255	27627.4	1,016,706,255	85.8	0.00
AP	50	[0.10, 0.04, 0.02, 0.04]	4	889,216,748	26928.5	889,216,748	165.8	0.00
AP	50		5	835,527,844	52023.2	835,527,844	231.9	0.00
AP	60		2	1,385,736,507	51089.4	1,385,736,507	17.3	0.00
AP	60	[1000, 1000, 1000, 1000] &	3	1,198,319,519	109477.2	1,198,319,519	153.6	0.00
AP	60	[0.10, 0.04, 0.02, 0.04]	4	1,060,190,998	93018.6	1,060,190,998	289.2	0.00
AP	60		5	974,812,578	110775.7	974,812,578	343.5	0.00
AP	70		2	1,687,850,457	96426.9	1,687,850,457	28.9	0.00
AP	70	[1000, 1000, 1000, 1000] &	3	1,430,761,480	227529.4	1,430,761,480	287.0	0.00
AP	70	[0.10, 0.04, 0.02, 0.04]	4	1,251,319,135	219192.9	1,251,319,135	614.2	0.00
AP	70		5	1,158,450,719	303828.2	1,161,223,813	638.6	0.24
AP	80		2	1,844,646,986	228716.8	1,844,646,986	46.8	0.00
AP	80	[1000, 1000, 1000, 1000] &	3	1,566,411,361	485130.6	1,566,411,361	443.8	0.00
AP	80	[0.10, 0.04, 0.02, 0.04]	4	1,379,464,866	616853.6	1,379,464,866	916.8	0.00
AP	80		5	1,255,718,409	829368.4	1,255,718,409	1163.2	0.00

Table B3: The comparison of solutions between the enhanced Benders decomposition method (de Camargo et al., 2017) and HUBBI on the TR dataset.

Datasets	Size	Fixed cost & Variable cost	Number of hubs p	Solution of Benders	Time of Benders (s)	Solution of HUBBI	Time of HUBBI (s)	Gap (%)
TR	25		2	273,822,824	183.6	273,822,824	1.1	0.00
TR	25	[2500,3000,3500,3000] &	3	262,054,430	291.2	262,054,430	5.5	0.00
TR	25	[0.08,0.04,0.03,0.04]	4	257,959,765	333.5	257,959,765	9.8	0.00
TR	25		5	261,026,538	416.4	261,026,538	15.1	0.00
TR	25		2	216,517,536	186.4	216,517,536	1.2	0.00
TR	25	[1000, 1000, 1000, 1000] &	3	203,834,757	298.3	203,834,757	6.0	0.00
TR	25	[0.10, 0.04, 0.02, 0.04]	4	197,010,925	279.0	197,010,925	10.9	0.00
TR	25		5	197,595,574	381.5	197,595,574	18.0	0.00
TR	30		2	362,997,492	480.8	362,997,492	2.2	0.00
TR	30	[2500,3000,3500,3000] &	3	349,521,392	845.4	349,521,392	10.8	0.00
TR	30	[0.08,0.04,0.03,0.04]	4	344,297,964	1032.1	344,302,034	19.3	0.00
TR	30		5	343,031,781	1372.2	343,035,851	32.0	0.00
TR	30		2	289,569,897	496.2	289,569,897	2.5	0.00
TR	30	[1000, 1000, 1000, 1000] &	3	273,351,870	811.5	273,351,870	11.8	0.00
TR	30	[0.10, 0.04, 0.02, 0.04]	4	264,608,513	1061.7	264,608,513	22.4	0.00
TR	30		5	256,670,023	1147.4	256,670,023	35.0	0.00
TR	40		2	874,941,740	4050.9	874,941,740	10.4	0.00
TR	40	[1000, 1000, 1000, 1000] &	3	785,921,235	10388.5	785,921,235	55.0	0.00
TR	40	[0.10, 0.04, 0.02, 0.04]	4	720,557,545	11198.5	720,557,544	90.2	0.00
TR	40		5	667,368,174	10046.7	670,413,080	134.9	0.46
TR	50		2	1,218,645,558	14742.4	1,218,645,558	31.9	0.00
TR	50	[1000, 1000, 1000, 1000] &	3	1,096,381,415	20624.9	1,096,381,415	108.5	0.00
TR	50	[0.10, 0.04, 0.02, 0.04]	4	1,000,474,005	17422.6	1,000,474,005	228.0	0.00
TR	50		5	933,121,175	21458.3	933,143,216	316.5	0.00
TR	60		2	1,519,149,133	60389.4	1,519,149,133	63.3	0.00
TR	60	[1000, 1000, 1000, 1000] &	3	1,388,169,186	120311.2	1,388,169,186	265.7	0.00
TR	60	[0.10, 0.04, 0.02, 0.04]	4	1,279,763,150	95900.9	1,279,763,150	447.0	0.00
TR	60		5	1,205,363,138	113056.0	1,205,363,137	677.0	0.00
TR	70		2	1,869,856,077	94478.9	1,869,856,077	165.3	0.00
TR	70	[1000, 1000, 1000, 1000] &	3	1,697,685,138	139709.8	1,697,685,138	387.0	0.00
TR	70	[0.10, 0.04, 0.02, 0.04]	4	1,576,487,744	173331.2	1,576,487,744	646.2	0.00
TR	70		5	1,487,520,419	224004.2	1,487,520,419	1258.8	0.00
TR	81		2	2,102,164,638	442593.0	2,102,164,638	297.6	0.00
TR	81	[1000, 1000, 1000, 1000] &	3	1,894,998,242	387055.7	1,895,006,222	680.1	0.00
TR	81	[0.10, 0.04, 0.02, 0.04]	4	1,771,026,963	634818.8	1,771,026,963	1131.2	0.00
TR	81		5	1,660,915,673	753291.2	1,660,915,673	1868.1	0.00