Ann Oper Res (2017) 258:655–677
DOI 10.1007/s10479-015-2091-2

Exact and heuristic approaches for the cycle hub location problem

Ivan Contreras[1] · **Moayad Tanash**[1] · **Navneet Vidyarthi**[1]

Published online: 6 January 2016

Abstract In this paper, we present solution algorithms for the cycle hub location problem (CHLP), which seeks to locate p hub facilities that are connected by means of a cycle, and to assign non-hub nodes to hubs so as to minimize the total cost of routing flows through the network. This problem is useful in modeling applications in transportation and telecommunications systems, where large setup costs on the links and reliability requirements make cycle topologies a prominent network architecture. We present a branch-and-cut algorithm that uses a flow-based formulation and two families of mixed-dicut inequalities as a lower bounding procedure at nodes of the enumeration tree. We also introduce a metaheuristic based on greedy randomized adaptive search procedure to obtain initial upper bounds for the exact algorithm and to obtain feasible solutions for large-scale instances of the CHLP. Numerical results on a set of benchmark instances with up to 100 nodes and 8 hubs confirm the efficiency of the proposed solution algorithms.

Keywords Hub location · Cycles · Branch-and-cut · GRASP

1 Introduction

Hub location problems (HLPs) arise in the design of hub-and-spoke networks. They have a wide variety of applications in airline transportation, freight transportation, rapid transit systems, trucking industries, postal operations, and telecommunications networks. These systems serve demand for transportation of passengers, commodities, and/or transmission of information (data, voice, video) between multiple origins and destinations. Instead of connecting every origin-destination (O–D) pair directly, hub-and-spoke networks serve customers via a small number of links, where hub facilities consolidate the flows from many origins, transfer them through the hub level network, and eventually distribute them to their

✉ Ivan Contreras
icontrer@encs.concordia.ca

[1] Concordia University and Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal H3G 1M8, Canada

final destinations. The use of fewer links in the network concentrates flows at the hub facilities, allowing economies of scale to be applied on routing costs, besides helping to reduce setup costs and to centralize commodity handling and sorting operations. Broadly speaking, HLPs consider the location of a set of hubs and the design of the hub-and-spoke network so as to minimize the total flow cost.

Since the seminal work by O'Kelly (1986), several classes of fundamental discrete HLPs, such as p-hub median problems, uncapacitated hub location problems, p-hub center problems, and hub covering problems, have been studied in the literature. For a detailed classification and review of discrete HLPs, the readers are referred to Alumur and Kara (2008), Campbell and O'Kelly (2012), Zanjirani Farahani et al. (2013), and Contreras (2015). For the case of continuous HLPs, we refer the readers to Iyigun (2013) and references therein. Even though these problems are different on a number of characteristics, mainly due to their particular applications, the vast majority of them share in common four assumptions. The first one is that flows have to be routed via hubs and thus, paths between O–D nodes must include at least one hub. Second, it is possible to connect hubs with more effective pathways that allow a constant discount factor to be applied to the flow cost between hubs. The third assumption is that hub arcs have no setup cost and thus, hub facilities can be connected at no additional cost. The last one is that distances between nodes satisfy the triangle inequality.

To some extent, the above mentioned assumptions and their implications simplify the network design decisions. For example, the last two assumptions allow the backbone network to be fully interconnected (i.e. a complete graph), whereas the access network is determined by the allocation pattern of O–D nodes to hub facilities. Moreover, the combination of the first, third and fourth assumptions results in O–D paths containing at least one hub or at most two hub nodes. This results in HLPs having a number of attractive theoretical features, which have given rise to various mathematical models (Ernst and Krishnamoorthy 1998a; Labbé and Yaman 2004; Hamacher et al. 2004; Contreras and Fernández 2014) and specialized solution algorithms (Ernst and Krishnamoorthy 1998b; Labbé et al. 2005b; Cetiner et al. 2010; Contreras et al. 2011; Martins de Sá et al. 2015b) that exploit the structure of the hub-and-spoke network to solve real-size instances. In several applications, these assumptions are reasonable and provide a good approximation to reality. However, in other applications, they can lead to unrealistic results.

It is known that fully interconnected networks may be prohibitive in applications where there is a considerable setup cost associated with the hub arcs (see, for instance, O'Kelly and Miller 1994; Klincewicz 1998). To overcome this deficiency, several models considering incomplete hub networks have been introduced. The so-called hub arc location problems (Campbell et al. 2005a, b) relax the assumption of full interconnection between hubs and deals with the location of a set of hub arcs that may (or may not) require a particular topological structure in the induced network. Some of these models do not even require the hub arcs to define a single connected component. Alumur et al. (2009) and Calık et al. (2009) study the design of incomplete hub networks with single assignments in which no network structure other than connectivity is imposed on the backbone network. Other works have also proposed different models that consider an incomplete backbone network with a particular topological structure. For example, Contreras et al. (2009, 2010) and Martins de Sá et al. (2013) study the design of tree-star hub networks in which the hubs have to be connected by means of a tree and the O–D nodes follow a single allocation pattern to hubs. These papers focus on the minimization of the total flow cost whereas Kim and Tcha (1992), Lee et al. (1996), and Lee et al. (1993) consider minimizing the setup costs associated with the design of tree-star networks. Labbé and Yaman (2008) and Yaman (2008) consider the design of star-star networks in which hub nodes are directly connected to a central node (i.e. star backbone

network) and the O–D nodes are assigned to exactly one hub node. Yaman (2009) studies the problem of designing a three-layer hub-and-spoke network, where the top layer consists of a complete network connecting the central hubs, and the second and third layers are unions of star networks connecting the remaining hubs to central hubs and the O–D nodes to hubs, respectively. Yaman and Elloumi (2012) consider the design of two-level star networks, while taking into consideration the service quality in terms of the length of paths between pair of O–D nodes. Martins de Sá et al. (2015a, b) study the problem of designing a hub network in which hubs are connected by means of a set of lines where the objective is to minimize the total weighted travel time between pairs of nodes.

In this paper, we study the *cycle hub location problem* (CHLP), which seeks to locate p hub facilities that are connected with a set of hub arcs by means of a cycle. Each O–D node must be allocated to exactly one hub (i.e. single assignment) and flows between pair of nodes have to be routed through the cycle-star network so as to minimize the total flow cost. The CHLP is a challenging NP-hard problem that combines location and network design decisions. The location decisions focus on the selection of the set of nodes to locate facilities, whereas the network design decisions focus on the design of the cycle-star network, by selecting the access and hub arcs as well as the routing of flows through the network. This problem was first introduced in Contreras and Fernández (2012) in the context of general network design problems, but to the best of our knowledge, there is no paper in the literature dealing with approximate or exact solution methods for solving it.

The CHLP shares some similarities with other network design problems in which a cycle-star network is sought. The so-called ring-star problem (RSP) that arises in the design of telecommunications networks, introduced by Labbé et al. (2004), aims to locate a simple cycle through a subset of nodes with the objective of minimizing the sum of setup costs of the cycle and assignment costs from the vertices not in the cycle to their closest vertex on the cycle. Another closely related problem is the median cycle problem (MCP), studied by Labbé et al. (2005a). This problem arises in the design of ring-shaped infrastructures and consists of finding a simple cycle that minimizes the setup costs for the cycle, such that the total assignment cost of the non-visited nodes do not exceed a given budget constraint. Current and Schilling (1994) and Gendreau et al. (1997) study covering versions of the RSP in which all nodes must be within a prespecified distance from the cycle. Baldacci et al. (2007) present the capacitated m-ring-star problem, which deals with the location of m cycles that pass through a central node and the assignment of nodes to cycles. Lee et al. (1998) and Xu et al. (1999) study the Steiner ring-star problem, in which the cycle only contains Steiner nodes chosen from a given set. Current and Schilling (1994) consider the median tour problem, where a cycle with p nodes has to be located. It is a bicriteria problem which consists of minimizing the setup cost of the cycle and of minimizing the total assignment cost of nodes to their closest facilities. Liefooghe et al. (2010) study a bi-objective ring star problem, in which the setup cost of the cycle and the assignment costs are considered.

All the above mentioned problems focus on the minimization of the setup cost for the design of the network and on the assignment of nodes to facilities. Service is given at or from the facilities, so that service demand occurs at nodes. In the case of HLPs, and in particular the CHLP, service demand is between pairs of nodes and the facilities are used as intermediate locations in the routes that connect node pairs. Therefore, in addition to the network design and assignment decisions considered in the above problems the CHLP considers additional routing decisions and focuses on the minimization of the total flow cost between many node pairs. This makes the problem more challenging as the O-D paths need to be known to compute the routing cost. In fact, even if the location of the hub facilities and the assignment

of O-D nodes to hubs is known, the problem remains NP-hard as it reduces to the minimum flow cost Hamiltonian cycle problem (see Ortiz-Astorquiza et al. 2015).

Potential applications where the location of a hub cycle is required arise in the design of telecommunications networks. In this case, hub facilities correspond to electronic equipment, such as concentrators, multiplexors and switches, and demand flows are data packages routed over a variety of physical media, such as coper cables, fiber-optic cables and telephone lines, or through the air by using satellite channels and microwave links. A common architecture of such networks consists of a number of tributary networks that connect nodes to the hubs, and a backbone network that interconnects the hubs. Due to the configuration of these networks, backbone links (i.e. hub arcs) usually have higher capacities and route larger volumes of flow than tributary links (i.e. access arcs). A discount on the costs of using a backbone link is thus perceived as compared to the cost of a tributary link. These usage (or communication) costs represent operations costs or fees for using a public network. Given the large setup costs incurred for the activation of links, network planners usually consider the design of a network containing the minimum number of links so as to minimize the total communication cost. In this sense, tree-star and line-star topologies are very attractive network topologies. However, these may not be appropriate for telecommunication networks where there are requirements for the backbone network to guarantee that there is more than one path between any pair of hubs, in the event that a backbone link fails. For that reason, a cycle-star topology may be preferred as it offers an alternative path between every pair of hubs whenever a link fails. Hence, a hub cycle guarantees connectivity of the remaining network while minimizing the setup cost. For more details about the design of telecommunications hub networks, we refer to the review paper by Klincewicz (1998).

Additional applications for the location of hub cycles arise in public transportation planning, in particular in the design of rapid transit systems. In this case, network planners may be interested in studying the impact of extending an already established public transportation network in a metropolitan region by locating a circular rapid transit line (i.e. hub cycle), such as a subway, a tram or an express bus lane. Examples of such circular lines are the Moscow Underground, the Melbourne Circular Tram Line, and some of the Montreal bus lines, among others. Hub facilities correspond to subway, tram or bus stations where a change of mode of transportation is usually possible. Nonhub nodes can be seen as bus stops, taxi stations or urban districts. Demand flow represents users traveling between O-D pairs and the goal is to improve the network's overall efficiency, measured in terms of the users' travel time. In this case, the discount factor represents the use of a faster transport technology to connect hub node pairs. In some situations, a circular line may be desirable not only due to reliability requirements but also because it offers an alternative path that can considerably reduce the travel time for some O-D pairs. The case of non-circular lines has been studied in detail in Martins de Sá et al. (2015a, b).

The main contribution of this paper is to propose exact and heuristic approaches for the CHLP. In particular, we present a flow based formulation for the CHLP which is used in a branch-and-cut (BC) algorithm to obtain optimal solutions for small to medium size instances and to provide lower bounds for larger instances. It uses two families of valid inequalities, which can be seen as an extension of the mixed-dicut inequalities for multi-commodity network design problems, to improve the linear programming (LP) relaxation bounds at some nodes of the enumeration tree. We develop separation heuristics to find violated inequalities efficiently. Moreover, we introduce a metaheuristic based on a greedy randomized adaptive search procedure (GRASP) to obtain initial upper bounds for the BC algorithm and to obtain feasible solutions for large-scale instances of the CHLP. In order to evaluate the efficiency

and limitations of our algorithms, extensive computational experiments were performed on benchmark instances with up to 100 nodes and 8 hubs.

The remainder of the paper is organized as follows. Section 2 provides a formal definition of the problem and presents a flow-based formulation. The BC algorithm and the metaheuristic are presented in Sects. 3 and 4, respectively. The computational results and the analysis are presented in Sect. 5. Conclusions follow in Sect. 6.

2 Definition and formulation of the problem

Let $G = (N, A)$ be a complete digraph, where $N = \{1, 2, \ldots, n\}$ represents the set of nodes as well as the potential sites for locating hubs and $A = N \times N$ is the set of arcs. For each ordered pair $(i, j) \in A$, let W_{ij} denote the amount of flow between origin i and destination j. Thus, $O_i = \sum_{j \in N} W_{ij}$ is the total flow originating at node $i \in N$, and $D_i = \sum_{j \in N} W_{ji}$, is the total flow destined to node $i \in N$. The distances, or flow costs d_{ij} between nodes i and j are assumed to be symmetric, however, they may not satisfy the triangle inequality property. Given that hub nodes are no longer fully interconnected, O–D paths on the solution network may contain more than two hub nodes. The per unit flow cost is then given by the length of the path between an origin and a destination, where the discount factor $0 < \alpha < 1$ is applied to all hub arcs contained on the path.

The CHLP seeks to determine the location of exactly p hubs which are connected by means of a cycle, and the routing of flows through the hub-and-spoke network. Each node has to be allocated to exactly one hub and if a node is selected as a hub, then it is self-assigned. The objective is to minimize the total flow cost. In every feasible solution to the CHLP: (1) there exists p hub arcs; (2) every hub node is connected with exactly two other hub nodes; (3)) the graph induced by the hubs does not contain subtours, and (4) there are exactly two paths between every pair of nodes on the solution network. This makes the CHLP more difficult to formulate and solve than classical HLPs, as the shortest path between O–D nodes, containing an undetermined number of hub nodes and hub arcs, needs to be determined to evaluate the objective function. Note that when $p \in \{1, 2, 3\}$, the CHLP reduces to a classical p-hub median problem in which hubs are fully interconnected.

Before presenting a mixed integer programming formulation, we first define the *graph of flows* $G_F = (N, E_F)$, as the undirected graph with node set N and an edge associated with each pair $(i, j) \in N \times N$ such that $W_{ij} + W_{ji} > 0$. We assume that G_F is made up of a single connected component since otherwise the problem can be decomposed into several independent CHLPs, one for each connected component in G_F. If a particular application requires a single cycle and the graph of flows contains more than one connected component, we can replace these flows of value zero with $W_{ij} = \epsilon > 0$ sufficiently small.

In order to keep track of the path that is used to send the flow between O–D nodes, we use flow variables that are commonly used in the hub location literature, for instance Ernst and Krishnamoorthy (1998b), Contreras et al. (2010) and Alumur et al. (2015). For each $i \in N$ and $(k, m) \in A$, we define x_{ikm} equal to the amount of flow with origin in node $i \in N$ that traverses hub arc (k, m) in the direction $k \rightarrow m$. For each $i, k \in N; i \neq k$ we also define binary location/allocation variables z_{ik} that equals one if and only if a non-hub node i is allocated to hub k. When $z_{kk} = 1$, node k is selected as a hub and assigned to itself. Finally, for each $k, m \in N, k < m$, we introduce binary hub arc variables y_{km} that equals one if and only if hub arc (k, m) is selected, zero otherwise. Following Contreras and Fernández (2012), the CHLP can be formulated as the following mixed integer program:

$$(P) \quad \text{minimize} \sum_{i\in N}\sum_{k\in N}(c_{ik}O_i + c_{ki}D_i)z_{ik} + \sum_{i\in N}\sum_{k\in N}\sum_{\substack{m\in N\\ m\neq k}}\alpha c_{km}x_{ikm} \tag{1}$$

$$\text{subject to} \sum_{k\in N} z_{ik} = 1 \qquad i \in N \tag{2}$$

$$\sum_{k\in N} z_{kk} = p \tag{3}$$

$$\sum_{k\in N}\sum_{m\in N} y_{km} = p \tag{4}$$

$$\sum_{m>k} y_{km} + \sum_{k>m} y_{mk} = 2z_{kk} \qquad k \in N \tag{5}$$

$$O_i z_{ik} + \sum_{\substack{m\in N\\ m\neq k}} x_{imk} = \sum_{\substack{m\in N\\ m\neq k}} x_{ikm} + \sum_{m\in N} W_{im} z_{mk} \quad i,k \in N; k \neq i \tag{6}$$

$$z_{km} + y_{km} \leq z_{mm} \qquad k,m \in N; m > k \tag{7}$$

$$z_{mk} + y_{km} \leq z_{kk} \qquad k,m \in N; m > k \tag{8}$$

$$x_{ikm} + x_{imk} \leq O_i y_{km} \qquad i,k,m \in N; m > k \tag{9}$$

$$x_{ikm} \geq 0 \qquad i,k,m \in N \tag{10}$$

$$z_{km} \in \{0,1\} \qquad k,m \in N \tag{11}$$

$$y_{km} \in \{0,1\} \qquad k,m \in N; m > k \tag{12}$$

The first term of the objective function represents the flow cost on the access arcs whereas the second term evaluates the discounted flow cost on the hub arcs. Constraints (2) ensure that each node is assigned to exactly one hub. Constraint (3) is a cardinality constraint on the number of hubs that must be opened, whereas constraint (4) state that the number of hub arcs in the cycle is equal to p. Constraints (5) guarantee that each hub node must be connected to exactly two other hub nodes. Constraints (6) are the flow conservation constraints. Constraints (7) and (8) ensure that both end nodes of a hub arc are opened hubs and also, they ensure that non-hub nodes are assigned to an open hub. Constraints (9) state that the flow between hubs moves through the hub cycle. Finally, constraints (10–12) are the standard nonnegativity and integrality constraints. As mentioned in Contreras and Fernández (2012), the assumption that the graph of flows G_F contains a single connected component, together with Constraints (2–12), eliminates the need for subtour elimination constraints.

3 An exact algorithm

In this section, we present an exact branch-and-cut (BC) algorithm that uses the linear programming (LP) relaxation of formulation P as a lower bounding procedure at nodes of the enumeration tree. The LP bounds from the formulation are strengthened with the incorporation of two families of valid inequalities that exploit the structure of the CHLP.

3.1 Valid inequalities

The first set of inequalities is an adaptation of the so-called *mixed-dicut inequalities*, first introduced by Ortega and Wolsey (2003) for the fixed-charge, single commodity, network

flow problem and later extended to the multi-commodity case for the tree of hubs location problem by Contreras et al. (2010). Let Z denote the set of feasible integer solutions of (2–12). The mixed-dicut inequalities can be defined as follows.

Proposition 1 *For $i, m \in N$, $F \subseteq N \setminus \{m\}$, $J \subseteq N \setminus \{i, m\}$, and $Q = \sum_{j \in J \cup \{m\}} W_{ij}$, the inequality*

$$\sum_{k \in N \setminus (F \cup \{m\})} x_{ikm} + Q \left(\sum_{\substack{k \in F \\ k<m}} y_{km} + \sum_{\substack{k \in F \\ k>m}} y_{mk} \right) \geq \sum_{j \in J \cup \{m\}} W_{ij} \left(z_{jm} - z_{im} \right) \tag{13}$$

is valid for Z.

The idea behind Constraints (13) is that when m is a hub node and i is a nonhub node not allocated to m, the term on the right-hand-side (RHS) $\sum_{j \in J \cup \{m\}} W_{ij} \left(z_{jm} - z_{im} \right)$, denotes the total flow originating at i and having as destination some non-hub node $j \in J$ or hub m itself. Therefore, the RHS of (13) is a lower bound on the total flow arriving to m from i. Note that the RHS can only be non-negative when m is a hub and i is not allocated to m. As for the left-hand-side (LHS), in any feasible solution that node i is not assigned to hub m, the flow arriving to m from i will enter m through one intermediate hub $\hat{k}$ (possibly node i). Thus, only one of the terms of the LHS will be positive, depending on whether or not $\hat{k} \in F$, and such a value will be an upper bound on the value of the RHS when m is a hub and i is not assigned to m.

We can generalize the mixed-dicut inequalities (13) by considering a set of candidate hub nodes $M \subseteq N$ and the set of O–D nodes assigned to them as follows. Let $\delta^-(M) = \{(i, j) \in A : i \in N \setminus M, j \in M\}$ denote the set of arcs entering the set M.

Proposition 2 *For $i \in N$, $M \subseteq N \setminus \{i\}$, $J \subseteq N \setminus (M \cup \{i\})$, $F \subseteq \delta^-(M)$, and $Q_0 = \sum_{j \in (J \cup M)} W_{ij}$, the generalized mixed-dicut inequality*

$$\sum_{(k,m) \in \delta^-(M) \setminus F} x_{ikm} + Q_0 \left(\sum_{\substack{(k,m) \in F \\ k>m}} y_{mk} + \sum_{\substack{(k,m) \in F \\ k<m}} y_{km} \right) \geq \sum_{j \in (J \cup M)} W_{ij} \left(\sum_{m \in M} z_{jm} - \sum_{m \in M} z_{im} \right) \tag{14}$$

is valid for Z.

Proof Observe that when $m \in M$ are open hubs and node i is not allocated to any node in M, the RHS $\sum_{j \in (J \cup M)} W_{ij} (\sum_{m \in M} z_{jm} - \sum_{m \in M} z_{im})$, denotes the total flow coming from i and destined to either the hub nodes $m \in M$ or the non-hub nodes $j \in J$ assigned to some hub $m \in M$. The RHS of (14) is thus a lower bound on the total flow arriving to the set of hub nodes M from i. Note that this RHS can only be non-negative when there is one or more nodes $m \in M$ which are open hubs and i is not assigned to any of them, otherwise the RHS would be less than or equal to zero. In the case of the LHS, we note that in any feasible solution in which node i is not allocated to a hub $m \in M$, any amount of flow routed from i to nodes $m \in M$ will arrive via a subset of hub arcs in the cut $\delta^-(M)$. If at least one open hub arc is in F, then the second term of the LHS provides an upper bound on the total amount of flow originated at i with destination $M \cup J$. If all hub arcs in F are closed, then the first term of the LHS provides an upper bound on the total amount of flow originated at i with destination $M \cup J$ entering via a subset of open hub arcs in $\delta^-(M) \setminus F$ and the result follows. □

3.2 Separating mixed-dicut inequalities

Given a fractional solution $(\bar{x}, \bar{y}, \bar{z})$ of the LP relaxation of formulation (1)-(12), the separation problem of inequalities (13) and (14) must be solved to determine whether there exist a violated inequality at $(\bar{x}, \bar{y}, \bar{z})$.

In the case of (13), for each pair of nodes $i, m \in N$, we want to find sets F and J such that

$$\sum_{k \in N \setminus (F \cup \{m\})} \bar{x}_{ikm} + Q \left(\sum_{\substack{k \in F \\ k<m}} \bar{y}_{km} + \sum_{\substack{k \in F \\ k>m}} \bar{y}_{mk} \right) - \sum_{j \in J \cup \{m\}} W_{ij} \left(\bar{z}_{jm} - \bar{z}_{im} \right) < 0.$$

Contreras et al. (2010) present an exact algorithm for solving the separation problem of Constraints (13) for the tree of hubs location problem. Given that for each $i, m \in N$, the proposed algorithm requires the solution of several 2-dimensional knapsack problems, the optimal solution of the separation problem requires a considerable amount of time, especially for large-scale instances. Therefore, we next present a fast heuristic to approximately solve the separation problem so as to find violated inequalities (13).

Note that the set $J \subseteq N \setminus \{i, m\}$ affects both the LHS and RHS of the inequality, whereas the set $F \subseteq N \setminus \{m\}$ affects only the LHS. Moreover, given a set J and its associated $Q = \sum_{j \in J \cup \{m\}} W_{ij}$, we can efficiently select the set F that minimizes the value of

$$L(Q) = \min_{F \subseteq N \setminus \{m\}} \sum_{k \in N \setminus (F \cup \{m\})} \bar{x}_{ikm} + Q \left(\sum_{k \in F : k<m} \bar{y}_{km} + \sum_{k \in F : k>m} \bar{y}_{mk} \right),$$

using the following result.

Proposition 3 (Contreras et al. 2010) *Let $i, m \in N$, $Q \geq 0$, and $(\bar{x}, \bar{y}, \bar{z})$ be given. Then, a set $\overline{F} \subseteq N \setminus \{m\}$ that minimizes the value of $L(Q)$ is given by $\overline{F} = F_< \cup F_>$, where*

$$F_< = \left\{ k \in N : k < m \text{ and } \frac{\bar{x}_{ikm}}{\bar{y}_{km}} \geq Q \right\},$$

and

$$F_> = \left\{ k \in N : k > m \text{ and } \frac{\bar{x}_{ikm}}{\bar{y}_{mk}} \geq Q \right\}.$$

The proposed heuristic works by iteratively evaluating different subsets $J \subseteq N \setminus \{i, m\}$ and computing $L(Q)$ to check whether the associated inequality is violated or not. First of all, it constructs an initial set J by considering all $j \in N$ such that $(\bar{z}_{jm} - \bar{z}_{im}) > 0$. Then, it modifies the set J by arbitrarily removing elements from it (one at a time) and evaluating the magnitude of the (possible) violation of the inequality. Let δ denote the smallest difference between the LHS and RHS of the constraint. If the output of the algorithm gives $\delta < 0$, it means that a violated inequality has been found. The procedure is outlined in Algorithm 1.

Algorithm 1: Separation of inequalities (13) for (i, m)

$J \leftarrow \left\{ j \in N : \bar{z}_{jm} - \bar{z}_{im} > 0 \right\}$
$\delta \leftarrow L(Q) - \sum_{j \in J \cup \{m\}} W_{ij}(\bar{z}_{jm} - \bar{z}_{im})$
for $(l \in J)$ **do**
  $J \leftarrow J \setminus \{l\}$
  if $\left(\delta > L(Q) - \sum_{j \in J \cup \{m\}} W_{ij}(\bar{z}_{jm} - \bar{z}_{im}) \right)$ **then**
    $\delta \leftarrow L(Q) - \sum_{j \in J \cup \{m\}} W_{ij}(\bar{z}_{jm} - \bar{z}_{im})$
  else
    $J \leftarrow J \cup \{l\}$
  end if
end for

In the case of inequalities (14), for each $i \in N$, we want to find sets M, J and F such that

$$\sum_{(k,m) \in \delta^-(M) \setminus F} \bar{x}_{ikm} + Q_0 \left(\sum_{\substack{(k,m) \in F \\ k > m}} \bar{y}_{mk} + \sum_{\substack{(k,m) \in F \\ k < m}} \bar{y}_{km} \right) - \sum_{j \in (J \cup M)} W_{ij} \left(\sum_{m \in M} \bar{z}_{jm} - \sum_{m \in M} \bar{z}_{im} \right) < 0.$$

Observe that, sets $M \subseteq N \setminus \{i\}$ and $J \subseteq N \setminus M \cup \{i\}$ affect both the LHS and RHS, whereas set $F \subseteq \delta^-(M)$ only affects the LHS. Therefore, for given sets M and J, we can efficiently select the set F that minimizes the value of

$$R(Q_0) = \min_{F \subseteq \delta^-(M)} \sum_{(k,m) \in \delta^-(M) \setminus F} \bar{x}_{ikm} + Q_0 \left(\sum_{\substack{(k,m) \in F \\ k > m}} \bar{y}_{mk} + \sum_{\substack{(k,m) \in F \\ k < m}} \bar{y}_{km} \right)$$

using a similar approach as in the case of Constraints (13).

Proposition 4 *Let $i \in N$, $Q_0 \geq 0$, and $(\bar{x}, \bar{y}, \bar{z})$ be a given LP solution. Then, a set $\overline{F} \subseteq \delta^-(M)$ that minimizes the value of $R(Q_0)$ is given by $\overline{F} = F_< \cup F_>$, where*

$$F_< = \left\{ (k, m) \in \delta^-(M) : k < m \text{ and } \frac{\bar{x}_{ikm}}{\bar{y}_{km}} \geq Q_0 \right\},$$

and

$$F_> = \left\{ (m, k) \in \delta^-(M) : k > m \text{ and } \frac{\bar{x}_{ikm}}{\bar{y}_{mk}} \geq Q_0 \right\}.$$

Proof A set $\overline{F} \subseteq \delta^-(M)$ that minimizes the value of $R(Q_0)$ is obtained by solving the optimization problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{(k,m)\in\delta^-(M)} \bar{x}_{ikm}(1-\mu_{km}) \\ & + Q_0 \left(\sum_{\substack{(k,m)\in\delta^-(M) \\ k>m}} \bar{y}_{mk}\mu_{km} + \sum_{\substack{(k,m)\in\delta^-(M) \\ k<m}} \bar{y}_{km}\mu_{km} \right) \qquad (15) \\ \text{subject to} \quad & \mu_{km} \in \{0,1\} \qquad \forall (k,m) \in \delta^-(M), \end{aligned}$$

where μ_{km} is a decision variable that denotes whether (k, m) belongs to F or not. The result follows form the fact that the objective function (15) can be restated as

$$\sum_{(k,m)\in\delta^-(M)} \bar{x}_{ikm} + \min \left\{ \sum_{\substack{(k,m)\in\delta^-(M) \\ k>m}} \left(Q_0\bar{y}_{mk} - \bar{x}_{ikm}\right)\mu_{km} + \sum_{\substack{(k,m)\in\delta^-(M) \\ k<m}} \left(Q_0\bar{y}_{km} - \bar{x}_{ikm}\right)\mu_{km} \right\}.$$

□

The proposed heuristic uses an iterative procedure to construct different subsets of $M \subseteq N\backslash\{i\}$ and $J \in N\backslash\{i, m\}$ and computes the associated $R(Q_0)$. We first order the variables $\bar{z}_{kk}$ non-increasingly and denote k_r the r-th element, i.e., $\bar{z}_{k_1k_1} \geq \bar{z}_{k_2k_2} \geq \cdots \geq \bar{z}_{k_nk_n}$. We then construct the set M by adding one element at a time with respect to this ordering. Every time a new element is added to M, an associated set J is constructed by considering all $j \in N$ such that $(\sum_{m\in M} \bar{z}_{jm} - \sum_{m\in M} \bar{z}_{im}) > 0$, and $R(Q_0)$ is computed to check whether the associated inequality is violated or not. If the violation obtained from the addition of the new element to M is higher that the violation at the previous iteration, the element is permanently added to M. Otherwise, it is removed and the next element in the sequence is selected as candidate. Once all candidates with $\bar{z}_{k_rk_r} > 0$ are considered, the algorithm tries to modify J by arbitrarily removing elements from it one at a time and evaluating the corresponding δ. The procedure is outlined in Algorithm 2.

3.3 A branch-and-cut algorithm

We present an exact branch-and-cut method for solving the CHLP. The idea is to solve the LP relaxation of P with a cutting-plane algorithm by initially including only Constraints (2–8) and (10–12) at the root node and iteratively adding Constraints (9), (13), and (14) only when violated by the current LP solution. When no more violated inequalities are found, we resort to CPLEX for solving the resulting formulation by enumeration, using a call-back function for generating additional violated Constraints (9),(13) and (14) at the nodes of the enumeration tree. The separation problem of inequalities (9) is solved by inspection at every node of the tree. The separation of inequalities (13) is carried out using Algorithm 1 at the root node and at every other node at a predetermined depth. The separation problem of inequalities (14) is carried out using Algorithm 2 and only at the root node of the enumeration tree. For Constraints (13) and (14), we limit the number of generated cuts at every iteration of the separation phase by using a threshold value ϵ for the minimum violation required for a cut to be added. We use a branching strategy in which the highest priority is given to the location variables (z_{kk}), followed by the hub arc variables (y), and least priority to the allocation variables (z_{ik}).

Algorithm 2: Separation of inequalities (14) for (i)

$M \leftarrow \emptyset, \delta_{min} \leftarrow 0, r \leftarrow 1$
Sort the values $\bar{z}_{kk}$ non-increasingly
while$(\bar{z}_{k_r k_r} > 0)$ **do**
  $M \leftarrow M \cup \{k_r\}$
  $J \leftarrow \left\{j \in N : \sum_{m \in M} \bar{z}_{jm} - \sum_{m \in M} \bar{z}_{im} > 0\right\}$
  $\delta \leftarrow R\,(Q_0) - \sum_{j \in J \cup M} W_{ij}(\sum_{m \in M} \bar{z}_{jm} - \sum_{m \in M} \bar{z}_{im})$
  if $(\delta < \delta_{min})$ **then**
    $\delta_{min} \leftarrow \delta$
  else
    $M \leftarrow M \setminus \{k_r\}$
  end if
  $r \leftarrow r + 1$
end while
for $(l \in J)$ **do**
  $J \leftarrow J \setminus \{l\}$
  $\delta \leftarrow R\,(Q_0) - \sum_{j \in J \cup M} W_{ij}(\sum_{m \in M} \bar{z}_{jm} - \sum_{m \in M} \bar{z}_{im})$
  if $(\delta < \delta_{min})$ **then**
    $\delta_{min} \leftarrow \delta$
  else
    $J \leftarrow J \cup \{l\}$
  end if
end for

4 A Heuristic approach

In this section we propose a GRASP for the CHLP. GRASP is a multi-start metaheuristic used for solving combinatorial optimization problems (Festa and Resende 2011). In GRASP, each iteration consists of two phases: a constructive phase and a local search phase.

For the CHLP we propose a constructive phase that comprise three steps. In the first step, a set of p hubs is randomly selected among a set of candidate nodes. The remaining nodes are then assigned to their closest open hub. Finally, a set of p hub arcs, associated with the selected hub nodes, are then chosen in such a way that they form a cycle on the backbone network. Later, a local search phase is used to improve the initial solution. In particular, a variable neighborhood descent (VND) method is used to systematically explore a set of neighborhoods that modify the structure of the network.

In what follows, solutions are represented by a set of hub nodes H, a set of hub arcs E, and an assignment mapping a. Therefore, solutions are designated by the form $S = (H, E, a)$,where $H \in N$ denotes the set of selected sites to locate hubs, i.e., $H(i) = 1$ if node $i \in N$ is selected to be a hub, $E : e \rightarrow R$ represents the set of arcs between hub nodes, i.e., $E(e) = 1$ if hub arc e exists, and $a : N \rightarrow H$ is the assigning mapping, i.e., $a(j) = m$ if node $j \in N$ is assigned to hub node $m \in H$.

4.1 Constructive phase

Let $S = (H, E, a)$ be a partial solution where $H(i) = null$, $E(e) = null$ and $a(j) = null$. To generate a feasible solution, three steps are considered; the selection of a set of hubs, the assignment of nodes to hubs and the connection of hubs so as to construct a cycle. A restricted candidate list (RCL) is built using a greedy function, where, at each iteration t, a sub-region $N_i^t(r) = \{j \in N^t : d_{ij} \leq r\}$ of candidate nodes N^t around a node i with a radius of size r is considered. We define the greedy function as

$$\psi_i^1 = \sum_{j \in N_i^1(r)} \left(W_{ij} + W_{ji}\right),$$

and

$$\psi_i^t = \sum_{j \in N_i^t(r)} \left(W_{ij} + W_{ji}\right) + \sum_{j \in N_i^t(r)} \sum_{k \in \{1,\ldots,t-1\}} \sum_{m \in N_{i(k)}^k(r)} (W_{jm} + W_{mj}),$$

for $t > 1$, where $i(k)$ denotes the node selected as a hub at iteration k. The first term of ψ_i^t represents the flow originating from node i with destination $N_i^t(r)$, and the total flow going into node i coming from nodes in $N_i^t(r)$. That is, node i is considered as a potential hub to serve nodes $j \in N_i^t(r)$. The second term of ψ_i^t represents the amount of flow that needs to be routed between nodes inside the sub-region $N_i^t(r)$ of a candidate hub node i and the nodes inside the sub-regions $N_{i(k)}^k(r)$ of the open hubs $i(k)$ selected in previous iterations $k = 1, \ldots, t-1$.

In order to achieve a trade-off between quality and diversity, a partially randomized greedy procedure is considered. At each iteration, one element is randomly selected from the RCL to become a hub. The RCL is updated at each iteration of the construction phase and contains the best candidate nodes N^t with respect to the greedy function. Let $\psi_{min}^t = \min\{\psi_i^t : i \in N^t\}$ and $\psi_{max}^t = \max\{\psi_i^t : i \in N^t\}$, then

$$RCL = \left\{i : \psi_i^t \leq \psi_{min}^t + \beta\left(\psi_{max}^t - \psi_{min}^t\right)\right\},$$

where $0 \leq \beta \leq 1$. Once a hub is located at a candidate node i, we remove all nodes in $N_i^t(r)$ from the set of candidate nodes N^{t+1}. When p hubs are opened, all the non-hub nodes are simply assigned to their closest opened hub. In order to construct a feasible solution, a *nearest neighbor algorithm* (see, Cook et al. 1998) is applied to connect the set of selected hubs by means of a cycle. It works by arbitrarily selecting a hub node and connecting it to the nearest hub not yet connected. The process continues until all hubs are connected. The constructive phase is outlined in Algorithm 3.

Algorithm 3: Constructive Phase of GRASP

$H \leftarrow \phi, t \leftarrow 0, N^t \leftarrow N$
while ($\mid H \mid \neq p$) **do**
  Evaluate ψ_i^t for all $i \in N^t$
  $RCL = \{i : \psi_i^t \leq \psi_{min}^t + \beta\left(\psi_{max}^t - \psi_{min}^t\right)\}$
  Select randomly $i^* \in RCL$
  $H \leftarrow H \cup i^*$
  $N^{t+1} \leftarrow N^t \backslash \{i^* \cup N_{i^*}^t(r)\}$
  $t \leftarrow t + 1$
end while
Assign each node $j \in N^t$ to its closest hub in H.
Connect hubs using the Nearest Neighbor Algorithm.

4.2 Local search phase

The local search phase is used to improve the initial solution obtained from the constructive phase. We use a local search procedure based on a VND method for the CHLP. To the best of our knowledge, the VND method was initially introduced by Brimberg and Mladenovic (1996) and is based on a systematic search in a set of k neighborhoods, $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k$. The

VND works by performing a local search in a neighborhood $\mathcal{N}_1$ until a local optimal solution is found. After that, the algorithm switches to neighborhoods $\mathcal{N}_2, \ldots, \mathcal{N}_k$, sequentially, until an improved solution is found. Each time the search improves the best known solution, the procedure restarts using the neighborhood $\mathcal{N}_1$. Our implementation of the VND algorithm explores three types of neighborhood structures. The first type consist of the classical shift and swap neighborhood. The former one considers all solutions that differ from the current one by changing the assignment of one node, whereas the latter one tries to improve the current solution by swapping the assignment of two nodes. Let $S = (H, E, a)$ be the current solution, then

$$\mathcal{N}_1(s) = \left\{ s' = \left(H, E, a'\right) : \exists! j \in N,\ a'(j) \neq a(j) \right\},$$

and

$$\mathcal{N}_2(s) = \left\{ s' = \left(H, E, a'\right) : \exists!(j_1, j_2),\ j_1' = a(j_2),\ j_2' = a(j_1),\ \forall j \neq j_1, j_2 \right\}.$$

To explore $\mathcal{N}_1(s)$, all pairs of the form (i, j) are considered, where $a(j) \neq i$ and for $\mathcal{N}_2(s)$ all pairs of the form (j_1, j_2) are considered, where $a(j_1) = a(j_2)$. In both cases, we use a best improvement strategy.

The second type of neighborhood structure affects the current set of open hubs. Let $S = (H, E, a)$ be the current solution and let $i \in N \backslash H$ be the nodes which are candidate to replace the open hubs located at site $m \in H$, then

$$\mathcal{N}_3(s) = \left\{ S' = \left(H', E', a'\right) : S' = H' \backslash \{m\} \cup \{i\}, m \in S',\ i \in N \backslash H \right\}.$$

To explore $\mathcal{N}_3(s)$ all nodes $m \in N \backslash H$ are considered, and a set of solutions is obtained from the current one by interchanging an open hub by a closed one and reassigning all the non-hub nodes to their closest open hub.

The third type of neighborhood structure is the so-called *2-opt* (Cook et al. 1998), commonly used in other optimization problems in which cycle structures are sought. The procedure works by deleting two hub arcs and reconnecting the network with a new cycle. Let $S = (H, E, a)$ be the current solution, then

$$\mathcal{N}_4(s) = \left\{ S = \left(H, E', a\right) : E' = E \backslash \{(i_1, j_1), (i_2, j_2)\} \cup \{(i_1, i_2), (j_1, j_2)\} \right\}.$$

In this neighborhood, a best improvement strategy is also considered.

5 Computational experiments

We conduct computational experiments to analyze and compare the performance of the flow-based formulation presented in Sect. 2 using the commercial solver CPLEX as well as the proposed solution approaches—the exact branch-and-cut algorithm and the GRASP metaheuristic. The formulation and solution algorithms were coded in C and run on a single possessor of an HP station with an Intel Xeon CPU E3-1240V2 processor at 3.40 GHz and 24 GB of RAM under Windows 7 environment. All integer programs were solved using the callback library of CPLEX 12.4. The numerical tests were performed using the Australian Post (AP) instances obtained from the OR library (http://mscmga.ms.ic.ac.uk/jeb/orlib/phubinfo.html). These instances comprise of postal flow and Euclidean distances between 200 postal districts in the metro Sydney area. In our experiments, we have selected instances with $|N| =$ 10, 20, 25, 40, 50, 60, 70, 75, 90, and 100 nodes. The number of hubs to be opened was set to $p = 4$, 6 and 8, and the value of discount factor was varied from α =0.2, 0.5 to 0.8.

In the first part of the computational experiments, we focus on analyzing the improvement of the linear programming (LP) relaxation bounds obtained when adding the cuts automatically generated by CPLEX and the two families of valid inequalities (13) and (14) introduced in Sect. 3 for the formulation P. In particular, we compare the results of the following experiments:

1. We solve the LP relaxation of formulation P and we do not allow CPLEX to add cuts.
2. We solve the LP relaxation of formulation P and we allow CPLEX to add cuts to improve the initial LP bounds. All the cuts parameters are set to their default values.
3. We solve the LP relaxation of formulation P and we dynamically add the mixed-dicut inequalities (13) using the separation heuristic presented in Sect. 3 to find violated inequalities. We set $\epsilon = 0.001$ for the minimum violation required for a cut to be added.
4. We solve the LP relaxation of formulation P and we dynamically add the generalized mixed-dicut inequalities (14) using the separation heuristic presented in Sect. 3 to find violated inequalities. We set $\epsilon = 0.01$ for the minimum violation required for a cut to be added.

The detailed results of these experiments are shown in Table 1. The first column lists the problem parameters such as the number of nodes $|N|$, the number of hubs to be opened p and the discount factor α for each instance. The second set of columns under the heading *CPLEX* reports the LP gap ($\%\,LP$), the LP gap after adding CPLEX cuts ($\%\,LP_{cuts}$), the number of cuts added by CPLEX ($\#cuts$), and the CPU time in seconds (CPU) to solve the LP and to add the cuts. The $\%\,LP$ gap is computed as $(UB - LP)/(UB) \times 100\,\%$, where UB denotes the best upper bound (or optimal solution value) and LP is the optimal value of the LP relaxation. The third set of columns under the heading *MDI* shows the results after adding the mixed-dicut inequalities (13) to P. The results include the LP gap ($\%LP$) after adding inequalities (13), the number of violated cuts added *(Cuts)*, and the CPU time. The last set of columns reports the LP gap ($\%LP$) after adding the generalized mixed-dicut inequalities (14), the number of violated inequalities *(Cuts)*, and the CPU time. In all cases, the CPU time include the time for separating and adding violated inequalities.

Results in Table 1 show that the average percent LP gap of formulation P is 6.64 % and ranges from 1.15 to 12.84 %. With the addition of CPLEX cuts, the LP gap is reduced to an average of 5.36 % and ranges from 0.48 to 10.58 %. However, after adding the mixed-dicut inequalities (13), the average LP gap is further reduced to 2.82 % with a range from 0.03 to 7.14 %. When adding the generalized mixed-dicut inequalities (14), the average LP gap is 1.94 % with a range from 0.00 to 5.86 %. In fact, constraints (14) are able to close the optimality gap, and obtain an integer optimal solution in 3 out of the 6 instances with 10 nodes. However, given that the number of cuts added is much larger compared to the number of cuts added by CPLEX, the CPU time to solve the associated LPs substantially increases. We also note that the quality of the obtained LP bounds seem to depend on the size and the number of hub facilities as well as the discount factor. For instance, the LP gap is worse as N and p increase. Also, the LP gaps tend to deteriorate as the value of the discount factor α increases. It is interesting to observe that the number of generated cuts also depend on these parameters.

We next compare the impact of separating both families of inequalities (13) and (14) and adding them to formulation P at the same time. In particular, we compare the results of the following experiments:

5. We solve the LP relaxation of formulation P and we first dynamically add Constraints (13) using the separation heuristic. When no more inequalities of this type can be found, Constraints (14) are then added.

Table 1 Comparison between CPLEX cuts and mixed-dicut inequalities

Instance	*CPLEX*				*MDI*			*GMDI*		
$\|N\|$-p-α	% LP	$\%\,LP_{cuts}$	*Cuts*	CPU	% LP	*Cuts*	CPU	% LP	*Cuts*	CPU
10-4-0.2	3.37	1.96	84	<1	0.92	182	<1	0.60	245	<1
10-4-0.5	5.34	2.18	75	<1	0.68	178	<1	0.07	232	<1
10-4-0.8	6.93	3.05	74	<1	0.64	225	<1	0.00	241	<1
10-6-0.2	5.62	2.64	69	<1	0.91	195	<1	0.00	265	<1
10-6-0.5	8.58	4.50	88	<1	1.54	289	<1	0.00	292	<1
10-6-0.8	10.62	7.21	71	<1	2.78	294	<1	0.09	467	<1
20-4-0.2	1.70	0.48	198	1	0.03	393	<1	0.10	432	1
20-4-0.5	4.33	3.32	288	2	1.48	1119	3	1.28	1449	9
20-4-0.8	5.11	3.41	215	2	1.47	843	2	0.83	1077	5
20-6-0.2	5.60	3.62	288	2	1.15	1218	5	0.72	2104	17
20-6-0.5	8.26	6.60	294	2	2.80	1643	5	1.69	2802	29
20-6-0.8	9.68	7.60	300	2	4.46	1388	5	2.87	2300	18
20-8-0.2	7.35	5.98	234	2	2.93	1517	4	1.98	2070	20
20-8-0.5	12.84	10.58	313	3	6.48	1868	7	4.46	3370	37
20-8-0.8	12.66	9.93	329	5	6.18	1671	6	3.87	3059	25
25-4-0.2	1.79	1.35	262	2	0.13	1198	3	0.26	1052	5
25-4-0.5	3.15	2.75	214	2	0.42	1327	5	0.33	1290	10
25-4-0.8	4.50	4.03	249	3	1.63	1211	5	0.76	1289	14
25-6-0.2	3.46	2.44	275	2	0.22	1528	6	0.15	1721	13
25-6-0.5	6.35	5.57	273	2	1.80	2059	6	1.04	2635	44
25-6-0.8	8.87	6.99	392	7	4.29	1476	7	2.72	2488	39
25-8-0.2	7.51	6.31	386	5	3.43	2527	14	2.82	4001	94
25-8-0.5	10.12	8.26	390	6	4.71	2279	11	3.15	3472	81
25-8-0.8	11.09	8.98	378	8	5.84	1851	10	3.66	3255	58
40-4-0.2	1.65	1.55	411	7	0.21	2613	25	0.44	2179	83
40-4-0.5	3.40	3.10	446	17	0.94	2922	66	0.80	3597	379
40-4-0.8	5.29	5.23	398	21	2.44	3164	76	1.60	3786	485
40-6-0.2	4.10	3.44	525	18	1.39	4958	88	1.47	6180	958
40-6-0.5	7.45	7.28	619	22	3.88	5619	124	3.10	8441	2389
40-6-0.8	8.34	8.01	511	27	4.80	3989	92	3.06	7543	2236
40-8-0.2	6.54	5.84	660	17	3.38	6369	96	2.81	11,774	2868
40-8-0.5	10.80	9.42	774	46	6.44	5039	125	4.82	10,104	4731
40-8-0.8	10.21	8.86	779	48	5.96	4211	106	3.83	9105	3427
50-4-0.2	1.15	1.02	392	23	0.08	2250	41	0.23	2304	144
50-4-0.5	2.57	2.58	610	51	0.52	3677	216	0.62	4091	1326
50-4-0.8	5.02	4.68	504	70	2.45	3838	305	1.66	4768	2184
50-6-0.2	3.56	2.96	631	49	1.06	5486	240	1.21	7448	3605
50-6-0.5	7.92	7.54	850	65	4.78	6132	386	4.13	11,016	11,900

Table 1 continued

Instance	*CPLEX*				*MDI*			*GMDI*		
$\|N\|$-p-α	% LP	$\%\, LP_{cuts}$	*Cuts*	CPU	% LP	*Cuts*	CPU	% LP	*Cuts*	CPU
50-6-0.8	8.20	8.08	756	74	5.18	5219	338	3.57	12,809	18,687
50-8-0.2	6.41	5.87	867	52	3.72	7381	326	3.51	12,827	11,965
50-8-0.5	10.76	9.82	1047	94	7.04	6984	393	5.86	14,684	27,641
50-8-0.8	10.60	10.03	866	96	7.14	5640	348	5.25	1278	18,706
Average	6.64	5.36	413.93	20	2.82	2713.57	83	1.94	4179.57	2720

6. We solve the LP relaxation of formulation P and we first dynamically add Constraints (14) using the separation heuristic. When no more inequalities of this type can be found, Constraints (13) are then added.

The results of these experiments are shown in Table 2. For every instance, we report the percent LP gap obtained after adding cuts from both families ($\%LP$), the number of added cuts from both families *(Cut1)* and *(Cut2)*, respectively, and the CPU time in seconds (CPU).

Table 2 shows that in the case of $MDI + GMDI$, the average percent LP gap is 1.75 % and ranges from 0 to 5.56 %, whereas in the case of $GMDI + MDI$, the average LP gap is slightly reduced to 1.68 % and ranges from 0 to 5.43 %. This is lower than the LP gaps reported in Table 1 for CPLEX, MDI, and $GMDI$. Moreover, Table 2 shows that in 34 out of 42 instances, $GMDI + MDI$ results in lower LP gap as compared to $MDI + GMDI$. Furthermore, by adding the two families of inequalities to formulation P, we are able to obtain an integer solution from the LP relaxation for 5 instances. In general, combining both families of valid inequalities provides significant improvement in the quality of the LP bounds. Although the $GMDI + MDI$ scheme provides, on average, the best LP bounds, the required CPU time is considerably larger than the other experiments performed. However, experiment 5 provides the best overall results in terms of the tradeoff between the quality of the LP bounds and the CPU time. Therefore, in the remainder of the experiments, we consider this scheme only.

In the second part of the computational experiments, we analyze the performance of our proposed exact and heuristic approaches. In particular, we compare the quality of the obtained solutions using both constraints (13) and (14) within a branch-and-cut framework and the GRASP algorithm introduced in Sect. 4. These experiments are performed on the same set of instances as before (ranging from 10 to 50 nodes). Throughout this experiment, we set a time limit to 86,400 s of CPU time. Instances that could not be solved to optimality within this time limit are marked with the label "Time". Preliminary computational experiments showed that the performance of the branch-and-cut algorithm was sensitive to the values of its various parameters. After some fine tuning, we fixed the parameters as follows. We set $\epsilon = 0.05$ for the minimum violation required for both families of cuts to be added. We also stop adding inequalities at a given node when the improvement of the LP bounds between the previous iteration and the current one is less that 0.08 %. The separation of inequalities (13) is carried out using Algorithm 1 at the root node and at all nodes for which the depth is multiple of 25.

The detailed results are reported in Table 3. The first column lists the problem parameters. The second set of columns under the label "$CPLEX$" reports the LP gap ($\%\, LP$), the percent deviation between final upper and lower bound ($\%\, Gap$), the CPU time (CPU) in seconds,

Table 2 Results when combining mixed-dicut inequalities

Instance	*MDI* + *GMDI*				*GMDI* + *MDI*			
$\|N\|$-p-α	% LP	#*Cut*1	#*Cut*2	CPU	% LP	#*Cut*1	#*Cut*2	CPU
10-4-0.2	0.59	197	77	<1	0.48	60	253	<1
10-4-0.5	0.00	178	78	<1	0.00	6	232	<1
10-4-0.8	0.00	225	60	<1	0.00	0	241	<1
10-6-0.2	0.00	195	54	<1	0.00	0	265	<1
10-6-0.5	0.00	289	91	<1	0.00	0	292	<1
10-6-0.8	0.27	302	203	<1	0.01	35	480	<1
20-4-0.2	0.00	398	14	<1	0.00	48	433	<1
20-4-0.5	0.99	1210	484	7	0.97	384	1449	11
20-4-0.8	0.67	934	481	5	0.62	225	1077	6
20-6-0.2	0.49	1383	624	16	0.45	446	2104	18
20-6-0.5	1.54	1821	763	15	1.38	472	2802	28
20-6-0.8	2.77	1583	1032	14	2.67	368	2300	17
20-8-0.2	1.99	1582	816	28	1.75	443	2070	19
20-8-0.5	4.43	2061	1356	27	4.27	418	3370	40
20-8-0.8	3.86	1834	1598	20	3.73	308	3059	29
25-4-0.2	0.04	1216	130	4	0.02	339	1052	7
25-4-0.5	0.05	1393	325	20	0.05	296	1290	13
25-4-0.8	0.54	1376	576	16	0.55	298	1289	17
25-6-0.2	0.04	1539	136	8	0.01	156	1823	25
25-6-0.5	0.66	2209	1042	26	0.69	529	2635	50
25-6-0.8	2.55	1680	1212	33	2.45	460	2488	41
25-8-0.2	2.69	2804	1324	38	2.50	938	4001	104
25-8-0.5	2.98	2490	1846	56	2.88	540	3472	75
25-8-0.8	3.66	2092	1579	46	3.51	382	3255	69
40-4-0.2	0.17	2661	120	31	0.04	985	2179	102
40-4-0.5	0.55	3131	778	156	0.46	1,025	3597	430
40-4-0.8	1.26	3609	1728	370	1.27	1,061	3786	520
40-6-0.2	1.33	5104	194	113	1.01	1,815	6180	1040
40-6-0.5	2.97	6269	2228	693	2.69	1,719	8441	2577
40-6-0.8	2.77	4478	4310	1365	2.78	1,190	7543	2247
40-8-0.2	2.57	6868	2912	705	2.52	1,654	11,774	3088
40-8-0.5	4.75	5516	4258	1590	4.53	1,464	10,104	4706
40-8-0.8	3.52	4675	5210	1946	3.55	1,122	9105	3439
50-4-0.2	0.03	2315	179	61	0.04	533	2304	182
50-4-0.5	0.19	3864	889	506	0.22	1,071	4091	1443
50-4-0.8	1.39	4436	1874	1391	1.36	1,142	4768	2396
50-6-0.2	0.80	5878	1403	822	0.73	2,110	7448	4125
50-6-0.5	3.75	6649	4312	4571	3.72	2,002	11,016	12,997
50-6-0.8	3.31	5783	6134	7365	3.30	1,481	12,809	19,314

Table 2 continued

Instance	*MDI* + *GMDI*				*GMDI* + *MDI*			
$\|N\|$-p-α	% LP	#*Cut*1	#*Cut*2	CPU	% LP	#*Cut*1	#*Cut*2	CPU
50-8-0.2	3.05	8113	3867	3226	3.05	2,635	12,827	12,592
50-8-0.5	5.56	7730	6523	8128	5.43	2,134	14,684	29,866
50-8-0.8	4.88	6182	8712	12348	5.00	1,331	14,882	21,671
Average	1.75	2958.38	1703	1090	1.68	801	4506.43	2936

and the number of explored nodes in the branching tree ($Nodes$). Note that, the final gap ($\%\, Gap$) is computed as $(UB - LB)/(UB) \times 100\,\%$, where UB and LB denote the best upper and lower bounds obtained at termination, respectively. The third set of columns under the heading "*Branch-and-Cut*" reports: ($\%\, LP_{cuts}$) the LP bound at the root node after adding valid inequalities (13) and (14), ($\%\, Gap$) the final percent deviation at termination, the CPU time (CPU) in seconds, and the number of explored nodes in the branching tree ($Nodes$).

The fourth set of columns reports the results of the GRASP. In order to assess the quality and robustness of the solution obtained from GRASP, the algorithm was run 30 times for each instance. The best objective value obtained across all the 30 runs is used to compute the best percentage deviation ($\%\, Dev$) with respect to the optimal solution value or the best LB bound obtained (*i.e.*, $\%\, Dev = (best\ solution\ GRASP - LB)/(best\ solution\ GRASP) \times 100\,\%$). The robustness is measured by using the average percent deviation ($\%\, Avg\ Dev$) using the best solutions obtained in each of the 30 runs. The CPU time for all the runs of the GRASP is also reported in seconds.

The results in Table 3 show that by using formulation P and a commercial solver (CPLEX), we were able to solve 31 instances to optimality and the final gaps on the remaining instances range from 0.60 to 10.10 %. On the other hand, the branch-and-cut algorithm succeeds in solving 35 out of the 42 instances to optimality within the time limit. For the remaining 7 instances, the final gaps range from 1.50 to 5.50 % . The branch-and-cut algorithm is faster than CPLEX on 30 out of 31 instances that were solved to optimality using both the algorithms. Moreover, our branch-and-cut algorithm is able to solve 4 instances that CPLEX is unable to solve within the time limit. For the instances that could not be solved to optimality, the branch-and-cut always provides smaller final percent gaps than CPLEX.

Table 3 also shows that the GRASP algorithm is very effective in finding high quality solutions for the problem. In particular, it succeeds in finding the optimal solution (or the best known solution) for 41 out of the 42 instances, while using only a fraction of CPU time compared to that of the branch-and-cut algorithms. In only one instance ($n = 50$, $p = 6$, and $\alpha = 0.5$), the branch-and-cut algorithm was able to improve the best solution obtained with GRASP by 0.07 %. The percent average deviations over 30 runs ranges from 0.00 to 5.55 %, thereby depicting the robustness of the GRASP algorithm. For 37 instances, GRASP yields the same solution in each run whereas the average deviation for the other 5 instances range from 0.02 to 0.55 %.

In order to further analyze the efficiency and robustness of proposed solution algorithms over large-scale instances, we have run a last set of computational experiments on instances ranging from 60 to 100 nodes. The results are summarized in Table 4. The first column lists the problem parameters. The second column reports the improved percent LP gap ($\%LP$) obtained by CPLEX when its cut generation strategy is activated.

Table 3 Computational results for the branch-and-cut and GRASP algorithms for small/medium size instances

Instance	*CPLEX*				*Branch-and-Cut*				*GRASP*		
$\|N\|$-p-α	% LP	% Gap	CPU	Nodes	% LP_{cut}	% Gap	CPU	Nodes	% Dev	% Avg	CPU
10-4-0.2	3.37	0.00	2	25	0.00	0.00	<1	0	0.00	0.00	27
10-4-0.5	5.34	0.00	<1	43	0.80	0.00	<1	8	0.00	0.00	27
10-4-0.8	6.93	0.00	<1	16	1.24	0.00	<1	15	0.00	0.00	27
10-6-0.2	5.62	0.00	1	73	3.93	0.00	<1	64	0.00	0.00	36
10-6-0.5	8.58	0.00	1	261	4.10	0.00	1	204	0.00	0.00	36
10-6-0.8	10.62	0.00	2	877	5.66	0.00	2	668	0.00	0.00	36
20-4-0.2	1.70	0.00	4	36	0.04	0.00	<1	3	0.00	0.00	189
20-4-0.5	4.33	0.00	18	485	1.36	0.00	5	54	0.00	0.00	189
20-4-0.8	5.11	0.00	29	1024	0.91	0.00	3	43	0.00	0.00	225
20-6-0.2	5.60	0.00	30	753	0.83	0.00	8	85	0.00	0.00	288
20-6-0.5	8.26	0.00	351	9693	2.08	0.00	24	706	0.00	0.00	315
20-6-0.8	9.68	0.00	1128	32,563	3.09	0.00	159	5422	0.00	0.05	351
20-8-0.2	7.35	0.00	181	4837	2.18	0.00	39	1177	0.00	0.00	405
20-8-0.5	12.84	0.00	2737	58,631	4.93	0.00	1570	20,306	0.00	0.00	468
20-8-0.8	12.66	0.00	8516	245,771	4.36	0.00	2120	24,271	0.00	0.03	459
25-4-0.2	1.79	0.00	14	66	0.26	0.00	3	19	0.00	0.00	378
25-4-0.5	3.15	0.00	47	248	0.28	0.00	6	16	0.00	0.00	441
25-4-0.8	4.50	0.00	119	928	0.81	0.00	13	50	0.00	0.00	450
25-6-0.2	3.46	0.00	32	312	0.20	0.00	6	22	0.00	0.00	594
25-6-0.5	6.35	0.00	328	3435	1.20	0.00	20	99	0.00	0.00	711
25-6-0.8	8.87	0.00	4065	37,955	2.82	0.00	240	2229	0.00	0.00	792
25-8-0.2	7.51	0.00	3170	22,421	3.01	0.00	340	2533	0.00	0.00	873
25-8-0.5	10.12	0.00	7711	68,475	3.37	0.00	1127	6327	0.00	0.00	999
25-8-0.8	11.09	0.00	18,230	152,835	4.04	0.00	5855	14,780	0.00	0.55	999
40-4-0.2	1.65	0.00	71	127	0.33	0.00	27	32	0.00	0.00	1944
40-4-0.5	3.40	0.00	3445	1816	0.82	0.00	112	69	0.00	0.00	2151
40-4-0.8	5.29	0.00	24,354	16,182	1.66	0.00	419	343	0.00	0.00	2358
40-6-0.2	4.10	0.00	4119	3332	1.87	0.00	369	1200	0.00	0.00	3303
40-6-0.5	7.45	2.10	Time	41,396	3.68	0.00	12,862	4225	0.00	0.00	3483
40-6-0.8	8.34	1.40	Time	46,600	3.56	0.00	48,756	14,547	0.00	0.00	3312
40-8-0.2	6.54	0.70	Time	52,491	3.77	0.00	17,623	9436	0.00	0.00	4905
40-8-0.5	10.80	8.70	Time	42,625	5.47	3.43	Time	6708	3.43	3.43	4788
40-8-0.8	10.21	8.00	Time	43,053	4.43	2.27	Time	8807	2.27	2.27	4347
50-4-0.2	1.15	0.00	234	190	0.13	0.00	47	28	0.00	0.00	169
50-4-0.5	2.57	0.00	6729	4332	0.47	0.00	262	76	0.00	0.00	177
50-4-0.8	5.02	1.80	Time	12,659	1.80	0.00	1570	407	0.00	0.00	162
50-6-0.2	3.56	0.00	25,218	7791	1.41	0.00	919	638	0.00	0.00	263
50-6-0.5	7.92	6.60	Time	13,173	4.57	2.32	Time	3554	2.39	2.39	270
50-6-0.8	8.20	7.70	Time	8609	4.35	2.64	Time	3771	2.64	2.66	253

Table 3 continued

Instance	*CPLEX*				*Branch-and-Cut*				*GRASP*		
$\|N\|$-p-α	% LP	% Gap	CPU	Nodes	% LP_{cut}	% Gap	CPU	Nodes	% Dev	% Avg	CPU
50-8-0.2	6.41	6.60	Time	11128	3.92	1.75	Time	3646	1.75	1.76	376
50-8-0.5	10.76	9.60	Time	10109	6.70	5.35	Time	2961	5.35	5.35	371
50-8-0.8	10.60	10.10	Time	14139	6.37	5.48	Time	2485	5.48	5.55	316

Table 4 Computational results for branch-and-cut and GRASP for large-scale instances

Instance	*CPLEX*	*Branch-and-Cut*				*GRASP*		
$\|N\|$-p-α	% LP	% LP_{cut}	% Gap	CPU	Nodes	% Dev	% Avg	CPU
60-4-0.2	1.69	0.96	0.00	243	224	0.00	0.00	256
60-4-0.5	2.99	1.09	0.00	761	188	0.00	0.00	302
60-4-0.8	5.41	2.10	0.00	8850	939	0.00	0.00	344
60-6-0.2	3.84	2.42	0.00	16,838	3927	0.00	0.02	422
60-6-0.5	7.54	4.86	3.45	Time	1801	3.45	3.58	584
70-4-0.2	1.57	0.83	0.00	729	379	0.00	0.00	406
70-4-0.5	3.33	1.43	0.00	4315	373	0.00	0.00	457
70-4-0.8	5.59	2.58	0.00	45,543	1871	0.11	0.22	551
70-6-0.2	3.88	2.05	0.00	17,063	2221	0.00	0.00	737
70-6-0.5	7.83	5.43	4.26	Time	802	4.26	4.26	836
75-4-0.2	1.52	1.06	0.00	1320	678	0.00	0.00	475
75-4-0.5	3.40	1.56	0.00	8595	501	0.00	0.00	629
75-4-0.8	5.64	2.54	0.22	Time	1727	0.32	0.32	685
75-6-0.2	3.94	2.58	0.25	Time	2666	0.25	0.25	868
75-6-0.5	7.61	5.42	4.62	Time	766	4.62	4.62	1047
90-4-0.2	1.45	1.35	0.00	9478	2529	0.00	0.00	912
90-4-0.5	3.14	1.93	0.00	65,524	1369	0.00	0.00	1042
90-4-0.8	5.40	3.28	3.00	Time	165	3.00	3.12	1158
90-6-0.2	3.91	3.34	2.67	Time	641	2.67	2.69	1432
90-6-0.5	7.63	5.98	5.96	Time	267	5.96	5.96	1788
100-4-0.2	1.50	1.39	0.00	22,353	2810	0.00	0.00	1187
100-4-0.5	3.24	2.26	0.74	Time	706	0.74	0.74	1448
100-4-0.8	5.52	3.41	3.39	Time	29	3.39	3.39	1488
100-6-0.2	4.12	3.73	3.19	Time	605	3.19	3.19	1930

It is worth mentioning that CPLEX fails to solve any of these instances within the time limit due to the size and complexity of problem. However, the exact branch-and-cut algorithm succeeds in solving 13 out the 24 instances to optimality and for the remaining instances, the final percent gap is within 6%. The GRASP is able to obtain the optimal solution for 12 out of the 13 instances that were solved to optimality using the exact algorithm. For the remaining one instance, the branch-and-cut was able to improve the best GRASP solution by 0.10%.

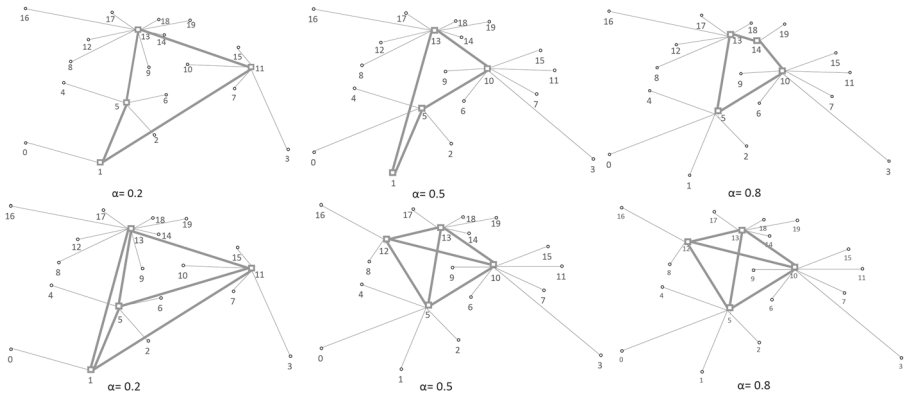

Fig. 1 Optimal networks for CHLP and pHMP with $p = 4$ and different discount factors

Finally, we compare the structure of solution networks obtained with the CHLP and the classical p-hub median problem (pHMP). Figure 1 illustrates the optimal solutions for the CHLP and pHMP of the AP instance with $n = 20$, $p = 4$ and different values of α. As can be seen in Fig. 1, in both models the location of hub facilities tend to become closer to each other as the discounting of costs decreases (i.e., α increases). When considering a small discount factor ($\alpha = 0.2$) the optimal set of hubs is the same for both models. However, with larger discount factors, the optimal set of hubs changes. In the case of $\alpha = 0.5$, given that node 1 has 23 % more incoming and outgoing flow than node 12, it seems better to open a hub at node 1 and to connect it to hubs 5 and 13 to reduce the flow cost. As similar situation happens in the case of $\alpha = 0.8$, where node 14 has 66 % more incoming and outgoing flow than node 12. Note that for the pHMP, the optimal network is the same for $\alpha = 0.5$ and $\alpha = 0.8$. However, it is not the case for the CHLP. This can be partially explained by the fact that in incomplete hub networks, a smaller discount factor encourages even more the dispersion of hub facilities over the plane, as compared to a larger discount factor where hubs tend to be closer together.

6 Conclusions

In this paper, we studied the cycle hub location problem. We presented two solution approaches: a branch-and-cut based exact approach and a GRASP based metaheuristic approach. Two families of valid inequalities based on mixed-dicut inequalities were presented and extensive computational experiments were conducted to evaluate their impact on the quality of LP bounds. One of these families of valid inequalities is new and generalizes the other one. These inequalities were embedded into a branch-and-cut framework to improve the lower bound at some nodes of the enumeration tree. A GRASP metaheuristic was also presented to efficiently obtain high quality solutions for large-scale instances. Computational results on benchmark instances with up to 100 nodes confirm the efficiency and robustness of the proposed algorithms.

Acknowledgements This research was partly founded by the Canadian Natural Sciences and Engineering Research Council under grants 418609-2012 and 386501-2010. This support is gratefully acknowledged. Thanks are due to two referees for their valuable comments.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

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

Alumur, S., Nickel, S., Saldana da Gama, F., & Secerdin, Y. (2015). Multi-period hub network design problems with modular capacities. *Annals of Operations Research*,. doi:10.1007/s10479-015-1805-9.

Alumur, S. A., Kara, B. Y., & Karasan, O. E. (2009). The design of single allocation incomplete hub networks. *Transportation Research Part B: Methodological*, *43*(10), 936–951.

Baldacci, R., Dell'Amico, M., & González, J. S. (2007). The capacitated m-ring-star problem. *Operations Research*, *55*(6), 1147–1162.

Brimberg, J., & Mladenovic, N. (1996). A variable neighbourhood algorithm for solving the continuous location-allocation problem. *Studies in Locational Analysis*, *10*, 1–12.

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*(12), 3088–3096.

Campbell, J. F., & O'Kelly, M. E. (2012). 25 years of hub location research. *Transportation Science*, *46*(2), 153–169.

Campbell, J. F., Ernst, A., & Krishnamoorthy, M. (2005a). Hub arc location problems: Part I: Introduction and results. *Management Science*, *51*(10), 1540–1555.

Campbell, J. F., Ernst, A., & Krishnamoorthy, M. (2005b). Hub arc location problems: Part II: Formulations and optimal algorithms. *Management Science*, *51*(10), 1556–1571.

Cetiner, S., Sepil, C., & Sural, H. (2010). Hubbing and routing in postal delivery systems. *Annals of Operations Research*, *181*, 109–124.

Contreras, I. (2015). Hub location problems. In G. Laporte, F. Saldanha da Gama, & S. Nickel (Eds.), *Location science* (pp. 311–344). New York: Springer.

Contreras, I., & Fernández, E. (2012). General network design: A unified view of combined location and network design problems. *European Journal of Operational Research*, *219*(3), 680–697.

Contreras, I., & Fernández, E. (2014). Hub location as the minimization of a supermodular set function. *Operations Research*, *62*, 557–570.

Contreras, I., Fernández, E., & Marín, A. (2009). Tight bounds from a path based formulation for the tree of hub location problem. *Computers & Operations Research*, *36*(12), 3117–3127.

Contreras, I., Fernández, E., & Marín, A. (2010). The tree of hubs location problem. *European Journal of Operational Research*, *202*(2), 390–400.

Contreras, I., Cordeau, J. F., & Laporte, G. (2011). Benders decomposition for large-scale uncapacitated hub location. *Operations Research*, *59*(6), 1477–1490.

Cook, W., Cunningham, W., Pulleybank, W., & Schrijver, A. (1998). *Combinatorial Optimization*. Hoboken: Wiley.

Current, J. R., & Schilling, D. A. (1994). The median tour and maximal covering tour problems: Formulations and heuristics. *European Journal of Operational Research*, *73*(1), 114–126.

Ernst, A. T., & Krishnamoorthy, M. (1998a). Exact and heuristic algorithms for the uncapacitated multiple allocation p-hub median problem. *European Journal of Operational Research*, *104*(1), 100–112.

Ernst, A. T., & Krishnamoorthy, M. (1998b). An exact solution approach based on shortest-paths for p-hub median problems. *INFORMS Journal on Computing*, *10*(2), 149–162.

Festa, P., & Resende, M. (2011). GRASP: Basic components and enhancements. *Telecommunication Systems*, *46*(3), 253–271.

Gendreau, M., Laporte, G., & Semet, F. (1997). The covering tour problem. *Operations Research*, *45*(4), 568–576.

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

Iyigun, C. (2013). The planar hub location problem: A probabilistic clustering approach. *Annals of Operations Research*, *211*, 193–207.
Kim, J. G., & Tcha, D. W. (1992). Optimal design of a two-level hierarchical network with tree-star configuration. *Computers & Industrial Engineering*, *22*(3), 273–281.
Klincewicz, J. (1998). Hub location in backbone/tributary network design: A review. *Location Science*, *6*(1), 307–335.
Labbé, M., & Yaman, H. (2004). Projecting the flow variables for hub location problems. *Networks*, *44*(2), 84–93.
Labbé, M., & Yaman, H. (2008). Solving the hub location problem in a star-star network. *Networks*, *51*(1), 19–33.
Labbé, M., Laporte, G., Rodríguez Martín, I., & Salazar-González, J. J. (2004). The ring star problem: Polyhedral analysis and exact algorithm. *Networks*, *43*(3), 177–189.
Labbé, M., Laporte, G., Rodríguez Martín, I., & Salazar-González, J. J. (2005a). Locating median cycles in networks. *European Journal of Operational Research*, *160*(2), 457–470.
Labbé, M., Yaman, H., & Gourdin, E. (2005b). A branch and cut algorithm for hub location problems with single assignment. *Mathematical programming*, *102*(2), 371–405.
Lee, Y., Lu, L., Qiu, Y., & Glover, F. (1993). Strong formulations and cutting planes for designing digital data service networks. *Telecommunication Systems*, *2*(1), 261–274.
Lee, Y., Lim, B. H., & Park, J. S. (1996). A hub location problem in designing digital data service networks: Lagrangian relaxation approach. *Location Science*, *4*(3), 185–194.
Lee, Y., Chiu, S., & Sanchez, J. (1998). A branch and cut algorithm for the steiner ring star problem. *International Journal of Management Science*, *4*(1), 21–34.
Liefooghe, A., Jourdan, L., & Talbi, E. G. (2010). Metaheuristics and cooperative approaches for the bi-objective ring star problem. *Computers & Operations Research*, *37*(6), 1033–1044.
O'Kelly, M. E. (1986). The location of interacting hub facilities. *Transportation Science*, *20*(2), 92–106.
O'Kelly, M. E., & Miller, H. J. (1994). The hub network design problem: A review and synthesis. *Journal of Transport Geography*, *2*(1), 31–40.
Ortega, F., & Wolsey, L. A. (2003). A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. *Networks*, *41*(3), 143–158.
Ortiz-Astorquiza, C., Contreras, I., & Laporte, G. (2015). The minimum flow cost Hamiltonian cycle problem: A comparison of formulations. *Discrete Applied Mathematics*, *187*, 140–154.
Martins de Sá, E., de Camargo, R., & de Miranda, G. (2013). An improved Benders decomposition algorithm for the tree of hubs location problem. *European Journal of Operational Research*, *226*, 185–202.
Martins de Sá, E., Contreras, I., & Cordeau, J. F. (2015a). Exact and heuristic algorithms for the design of hub networks with multiple lines. *European Journal of Operational Research*, *246*(1), 186–198.
Martins de Sá, E., Contreras, I., Cordeau, J. F., de Camargo, R. S., & de Miranda, G. (2015b). The hub line location problem. *Transportation Science*, *49*(3), 500–518.
Xu, J., Chiu, S. Y., & Glover, F. (1999). Optimizing a ring-based private line telecommunication network using tabu search. *Management Science*, *45*(3), 330–345.
Yaman, H. (2008). Star p-hub median problem with modular arc capacities. *Computers & Operations Research*, *35*(9), 3009–3019.
Yaman, H. (2009). The hierarchical hub median problem with single assignment. *Transportation Research Part B: Methodological*, *43*(6), 643–658.
Yaman, H., & Elloumi, S. (2012). Star p-hub center problem and star p-hub median problem with bounded path lengths. *Computers & Operations Research*, *39*(11), 2725–2732.
Zanjirani Farahani, R., Hekmatfar, M., Arabani, A. B., & Nikbakhsh, E. (2013). Hub location problems: A review of models, classification, solution techniques, and applications. *Computers & Industrial Engineering*, *64*(4), 1096–1109.