A Branch and Cut Algorithm for the Cycle Hub Location Problem

Ivan Contreras · Moayad Tanash · Navneet Vidyarthi

Received: date / Accepted: date

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 telecommunication systems, where large set-up 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 greedy randomized adaptive search algorithm that is used 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

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 transport, rapid transit systems, trucking industries, postal operations, and telecommunication 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 cus-
tomers 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 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 of 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), Farahani et al (2013), and Contreras (2015). For the case of continuous HLPs, we refer 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. Secondly, it is possible to connect hubs with more effective pathways that allow a constant discount factor to be applied to the flow cost between hub nodes. The third assumption is that hub arcs have no setup cost and thus, hub facilities can be connected at no extra 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; Contreras et al 2011; Martins de Sá et al 2015) 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 2005b,a) 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 topo-
logical structure of their 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 (2014, 2015) study the problem of designing a hub network in which hubs are connected by means of a set of lines and the aim 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 topology. 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 focuses on the selection of the set of nodes to locate facilities, whereas the network design decisions deals with the design of the cycle-star network, by selecting the access and hub arcs as well as the routing of flows through the network. The CHLP 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 telecommunication network, 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 must be located. It is a bicriterion 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. See Labbé et al (1998) and Laporte and Martín (2007) for additional models related to the location of cycle structures on a network.

Potential applications of cycle topologies arise in the design of telecommunication networks, pipelines and high speed train networks, design of emergency routes, newspaper delivery routes, and subway lines (Labbé et al 1998). In the former case, terminal nodes are usually connected to concentrators (or facilities) by point-to-point links, resulting in a star structure, and concentrators are interconnected by a ring. See Xu et al (1999) for an example in digital data service design. In the latter case, the goal is to select a set of facilities, which are served by a single vehicle route (a cycle), and the assignment of demand nodes to their closest facility. Laporte and Martín (2007) provide additional applications considering cycle star structures. Cycle star networks are preferred when the setup costs of the arcs of the network are very high and their reliability is an issue. When minimizing the setup cost, tree-star topologies are particularly attractive as they minimize the number of links on the network, but they contain exactly one path between pair of nodes. However, in the design of reliable networks, cycle topologies may be preferred to tree topologies as they offer an alternative path between any pair of nodes when a link connecting two nodes fails for some reason. Hence, a cycle topology guarantees connectivity of the remaining network while minimizing the setup cost in such situations.

The main contribution of this paper is to propose exact and heuristic algorithms 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 efficiently find violated inequalities. Moreover, we introduce a metaheuristic based on a greedy randomized adaptive search procedure (GRASP) that is used to obtain initial upper bounds for the BC and to obtain feasible solu-
tions 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 eight hubs.

The remainder of the paper is organized as follows. Section 2 provides a formal definition of the problem and presents the flow-based formulation. The BC algorithm and the metaheuristic are presented in Sections 3 and 4, respectively. The computational results and the analysis are presented in Section 5. Conclusions follow in Section 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 O/D nodes as well as the potential sites for locating hubs. For each ordered pair $i, j \in N \times N$, 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 with destination 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: i) there exist $p$ hub arcs; ii) every hub node is connected with exactly two other hub nodes; iii) the graph induced by the hubs does not contain subtours, and iv) 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 the 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 commonly used in the hub location literature (see,
for instance Ernst and Krishnamoorthy 1998b; Contreras et al 2010; 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 $(k, m)$. For each $i, k \in N; i \neq k$ we also define binary location/allocation variables $z_{ik}$ equal to 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 $y_{km}$ equal to one if and only if hub arc $(k, m)$ is selected. Following Contreras and Fernández (2012), the CHLP can be formulated as the following mixed integer program, denoted as FB:

\[
\begin{align*}
\text{minimize} & \sum_{i \in N} \sum_{k \in N} (c_{ik} O_i + c_{ki} D_i) z_{ik} + \sum_{i \in N} \sum_{k \in N} \sum_{m \in N, m \neq k} \alpha c_{km} x_{ikm} \\
\text{subject to} & \sum_{k \in N} z_{ik} = 1 & i \in N \\
& \sum_{k \in N} z_{kk} = p \\
& \sum_{k \in N} \sum_{m \in N} y_{km} = p \\
& \sum_{k < m} y_{km} + \sum_{k > m} y_{mk} = 2z_{kk} & k \in N \\
& O_i z_{ik} + \sum_{m \in N, m \neq k} x_{imk} = \sum_{m \in N, m \neq k} x_{ikm} + \sum_{m \in N} W_{im} z_{mk} & i, k \in N; k \neq i \\
& z_{km} + y_{km} \leq z_{mm} & k, m \in N; m > k \\
& z_{mk} + y_{km} \leq z_{kk} & k, m \in N; m > k \\
& x_{ikm} + x_{imk} \leq O_i y_{km} & i, k, m \in N; m > k \\
& x_{ikm} \geq 0 & i, k, m \in N \\
& z_{km}, y_{km} \in \{0, 1\} & k, m \in N
\end{align*}
\]
assumption that the graph of flows $G_F$ contains a single connected component, together with constraints (2)–(11), eliminates the need for subtour elimination constraints.

3 An Exact Algorithm

In this section we present an exact branch-and-cut algorithm that uses the linear programming (LP) relaxation of formulation FB 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)–(11). 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

$$
\sum_{k \in N \setminus (F \cup \{m\})} x_{ikm} + Q \left( \sum_{k \in F \cap \{m\}} y_{km} + \sum_{k \in F \cap \{m\}} y_{mk} \right) \geq \sum_{j \in J \cup \{m\}} W_{ij} (z_{jm} - z_{im}) \quad (12)
$$

is valid for $Z$.

Constraints (12) state that if the O/D nodes in set $J$ are assigned to hub $m$ and node $i$ is not assigned to $m$, then the amount of flow entering on $m$ via the hub arcs incident to $m$ has to be greater or equal to the sum of the flows with origin in $i$ and with destination in $J \cup \{m\}$, i.e. $\sum_{j \in J \cup \{m\}} W_{ij}$. Given that in any feasible solution to the CHLP the flow originated at $i$ and with destination $m$ and any nodes allocated to $m$ will be routed using the shortest path between these nodes, the flow will enter $m$ via another hub $\hat{k}$ (possibly node $i$) using hub arc $(\hat{k}, m)$. Thus, depending on the set $F$, the flow will be counted either using the flow variables of the first term of the left-hand-side of (12) or using the design variables of the second term.

We can generalize the mixed-dicut inequalities (12) by considering now 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 \( Q_0 = \sum_{j \in (J \cup M)} W_{ij}, \) then the generalized mixed-dicut inequality

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

(13)
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 right-hand-side \( \sum_{j \in (J \cup M)} W_{ij} \left( \sum_{m \in M} z_{jm} - \sum_{m \in M} z_{im} \right), \) 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 right-hand-side of (13) is thus a lower bound on the total flow arriving to the set of hub nodes \( M \) from \( i. \) Note that this right-hand-side 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 right-hand-side would be less than or equal to zero. In the case of the left-hand-side, we note that in any feasible solution that 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 left-hand-side 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 left-hand-side 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. \( \square \)

3.2 Separating Mixed-dicut Inequalities

Given a fractional solution \((\bar{x}, \bar{y}, \bar{z})\) of the LP relaxation of formulation (1)-(11), the separation problem of inequalities (12) and (13) must be solved to determine whether there exist a violated inequality at \((\bar{x}, \bar{y}, \bar{z}).\)

In the case of (12), for each pair of nodes \( i, m \in N, \) we want to find sets \( F \) and \( J \) such that

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

Contreras et al (2010) present an exact algorithm for solving the separation problem of constraints (12) for the tree of hub 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 (12).

Note that the set \(J \subseteq N \setminus \{i,m\}\) affects both the left-and right-hand-side of the inequality, whereas the set \(F \subseteq N \setminus \{m\}\) affects only the left-hand-side. 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

\[
L(Q) = \min_{F \subseteq N \setminus \{m\}} \left( \sum_{k \in N \setminus (F \cup \{m\})} x_{ikm} + Q \left( \sum_{k \in F, k < m} y_{km} + \sum_{k \in F, k > m} y_{mk} \right) \right),
\]

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 \(F \subseteq N \setminus \{m\}\) that minimizes the value of \(L(Q)\) is given by \(F = F_< \cup F_>\), where

\[
F_< = \{ k \in N : k < m \text{ and } \frac{x_{ikm}}{y_{km}} \geq Q \},
\]

and

\[
F_> = \{ k \in N : k > m \text{ and } \frac{x_{ikm}}{y_{mk}} \geq Q \}.
\]

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 \(J_0\) by considering all \(j \in N\) such that \((\bar{z}_{jm} - \bar{z}_{im}) > 0\). Then, it modifies the set \(J_0\) by removing elements from it (one at a time) and evaluating the magnitude of the (possible) violation of the inequality. Let \(\delta\) denote the smallest difference between the left-hand-side and right-hand-side of the constraint. If the output of the algorithm gives \(\delta < 0\), it means that a violated inequality has been found. The steps of the algorithm are outlined in Algorithm 1.

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

\[
\sum_{(k,m) \in \delta^-(M) \setminus F} x_{ikm} + Q_0 \left( \sum_{(k,m) \in F, k > m} y_{mk} + \sum_{(k,m) \in F, k < m} y_{km} \right)
\]

\[
- \sum_{j \in (J \cup M)} W_{ij} \left( \sum_{m \in M} z_{jm} - \sum_{m \in M} z_{im} \right) < 0.
\]

Observe that, sets \(M \subseteq N \setminus \{i\}\) and \(J \subseteq N \setminus M \cup \{i\}\) affect both the left and right-hand-sides, whereas set \(F \subseteq \delta^-(M)\) only affects the left-hand-side.
Algorithm 1: Separation of inequalities (12) for \((i,m)\)

\[
\begin{align*}
    J_0 &\leftarrow \emptyset \\
    \text{for } (j \in N) \text{ do} & \\
    & \quad \text{if } (\bar{z}_{jm} - \bar{z}_{im} > 0) \text{ then} \\
    & \quad \quad J_0 \leftarrow J_0 \cup \{j\} \\
    & \quad \text{end if} \\
    \text{end for} & \\
    \delta &\leftarrow L(Q) - \sum_{j \in J_0 \cup \{m\}} W_{ij}(\bar{z}_{jm} - \bar{z}_{im}) \\
    J &\leftarrow J_0 \\
    \text{for } (l \in J_0) \text{ do} & \\
    & \quad J \leftarrow J \setminus \{l\} \\
    & \quad \text{if } \left( \delta > L(Q) - \sum_{j \in J_0 \cup \{m\}} W_{ij}(\bar{z}_{jm} - \bar{z}_{im}) \right) \text{ then} \\
    & \quad \quad \delta \leftarrow L(Q) - \sum_{j \in J_0 \cup \{m\}} W_{ij}(\bar{z}_{jm} - \bar{z}_{im}) \\
    & \quad \text{else} \\
    & \quad \quad J \leftarrow J \cup \{l\} \\
    & \quad \text{end if} \\
    \text{end for} & \\
\end{align*}
\]

Therefore, for given sets \(M\) and \(J\), we can efficiently select the set \(F\) that minimizes the value of

\[
R(Q_0) = \min_{F \subseteq \delta^-(M) \setminus \delta^-(M) \setminus F} \sum_{(k,m) \in \delta^-(M) \setminus F} x_{ikm} + Q_0 \left( \sum_{(k,m) \in F} \frac{y_{nm}}{y_{km}} \right)
\]

using a similar approach as in the case of constraints (12).

**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 \(F \subseteq \delta^-(M)\) that minimizes the value of \(R(Q_0)\) is given by \(F = F_\leq \cup F_\geq\), where

\[
F_\leq = \{(k,m) \in \delta^-(M) : k < m \text{ and } \frac{x_{ikm}}{y_{km}} \geq Q_0\},
\]

and

\[
F_\geq = \{(m,k) \in \delta^-(M) : k > m \text{ and } \frac{x_{ikm}}{y_{km}} \geq Q_0\}.
\]

The proposed heuristic uses an iterative procedure to construct different subsets of \(M \subseteq N \setminus \{i\}\) and \(J \subseteq N \setminus \{i,m\}\) and computes the associated \(Q(Q_0)\). We first order the variables \(\bar{z}_{ik}\) by decreasing values and we denote \(k_r\) the \(r\)-th element according to that ordering. That is, \(\bar{z}_{k_1 k_1} \geq \bar{z}_{k_2 k_2} \geq \cdots \geq \bar{z}_{k_n k_n}\). 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 \(J_0\) 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 than 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 \(\bar{z}_{k_r k_r} > 0\) are considered, the algorithm tries to modify \(J_0\) by removing elements from it one at a time.
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 $FB$ with a cutting-plane algorithm by initially including only constraints (2)-(8), (5)-(4) at the root node and iteratively adding constraints (9), (12), and (13) 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),(12) and (13) 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 (12) is carried out using Algorithm 1 at the root node and at every other nodes for which the depth is multiple of 25. The separation problem of inequalities (13) is carried out using Algorithm 2 and only at the root node of the enumeration tree. For constraints (12) and (13), we limit the number of generated cuts.
at every iteration of the separation phase by using a threshold value $\epsilon$ 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 $(z_{ik})$.

4 A Heuristic Algorithm

We present a heuristic algorithm based on GRASP. The GRASP is a multi-start metaheuristic used for solving combinatorial optimization problems (Festa and Resende 2011). Each iteration consists of two phases: a constructive phase and a local search phase. In the constructive phase, we obtain a feasible solution using a three-step procedure. 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. A local search phase is later 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 $H \subseteq N$ 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 $m \in H$.

4.0.1 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 structure. A restricted candidate list (RCL) is built using a greedy function, where, at each iteration $t$ a sub-region $N_t^i(r) = \{j \in N^t : d_{ij} \leq r\}$ of candidate nodes $N^t$ around a node $i$ with a radius of size $r$ is considered. We define the greedy function as

$$
\psi_t^1 = \sum_{j \in N_t^i(r)} (W_{ij} + W_{ji}),
$$

and

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

for $t > 1$, where $i(k)$ denotes the node selected as a hub at iteration $k$. The first term of $\psi_t^1$ represents the flow originated at node $i$ with destination $N_t^i(r)$,
and the total flow going into node \( i \) coming from nodes in \( N^t_i(r) \). That is, node \( i \) is considered as a potential hub to serve nodes \( j \in N^t_i(r) \). The second term of \( \psi^t_i \) represents the amount of flow that needs to be routed between nodes inside the sub-region \( N^t_i(r) \) of a candidate hub node \( i \) and the nodes inside the sub-regions \( N^k_{i(k)}(r) \) 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 \( \psi^t_{\min} = \min\{\psi^t_i : i \in N^t\} \) and \( \psi^t_{\max} = \max\{\psi^t_i : i \in N^t\} \), then

\[
RCL = \{i : \psi^t_i \leq \psi^t_{\min} + \alpha (\psi^t_{\max} - \psi^t_{\min})\},
\]

where \( 0 \leq \alpha \leq 1 \). Once a hub is located at a candidate node \( i \), we remove all nodes in \( N^t_i(r) \) 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**

Let \( H \) be the subset of selected hub nodes

Let \( N^t \) be the set of candidate nodes at iteration \( t \)

\[ H \leftarrow \emptyset, t \leftarrow 1, N^0 \leftarrow N \]

**while** \(| H | \neq p \) **do**

Evaluate \( \psi^t_i \) for all \( i \in N^t \)

\[ RCL = \{i : \psi^t_i \leq \psi^t_{\min} + \alpha (\psi^t_{\max} - \psi^t_{\min})\} \]

Select randomly \( i^* \in RCL \)

\[ H \leftarrow H \cup i^* \]

\[ N^t \leftarrow N^{t-1} \setminus \{i^* \cup N^t_i\} \]

\[ t \leftarrow t + 1 \]

**end while**

Assign each node \( j \in N^t \) to its closest hub in \( H \).

Connect hubs using the Nearest Neighbor Algorithm.

### 4.0.2 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. The VND was initially proposed by Hansen and Mladenović (2001) 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 latter one tries to improve the current solution by changing the assignment of one node, whereas the former one considers all solutions that differ from the current one by swapping the assignment of two nodes. Let $S = (H, E, a)$ be the current solution, then

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

and

\[ \mathcal{N}_2(s) = \{ s' = (H, E, a') : \exists ! (j_1, j_2), j'_1 = a(j_2), j'_2 = a(j_1), \forall j \neq j_1, j_2 \}. \]

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 $i \in N \setminus H$ be the nodes which are candidate to replace the open hubs located at site $m \in H$, then

\[ \mathcal{N}_3(s) = \{ S' = (H', E', a') : S' = H' \setminus \{m\} \cup \{i\}, m \in S', i \in N \setminus H \}. \]

To explore $\mathcal{N}_3(s)$ all nodes $m \in N \setminus H$ 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

\[ \mathcal{N}_4(s) = \{ S = (H, E', a') : E' = E \setminus \{(i_1, j_1), (i_2, j_2)\} \cup \{(i_1, i_2), (j_1, j_2)\} \}. \]

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

5 Computational Results

We conduct computational experiments to analyze and compare the performance of the flow-based formulation presented in Section 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 processor 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
A Branch and Cut Algorithm for the Cycle Hub Location Problem

library (http://mscmga.ms.ic.ac.uk/jeb/orlib/phubinfo.html). These instances comprise of postal flow and Euclidean distances between 200 cites in Australia. In our experiments, we have selected instances with $|N| = 10, 20, 25, 40, 50, 60, 70, 75, 90, \text{ and } 100$ nodes. The number of hubs to be opened was set to $p = 4, 6 \text{ and } 8$, and the value of discount factor was varied from $\alpha = 0.2, 0.5 \text{ to } 0.8$.

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 (12) and (13) introduced in Section 3 for the formulation FB. In particular, we compare the results of the following experiments:

1. We solve the LP relaxation of formulation FB and we do not allow CPLEX to add cuts.
2. We solve the LP relaxation of formulation FB and we allow CPLEX to add cuts to improve the initial LP bounds. All the cuts parameters are set to their default settings.
3. We solve the LP relaxation of formulation FB and we dynamically add the mixed-dicut inequalities (12) using the separation heuristic presented in Section 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 FB and we dynamically add the generalized mixed-dicut inequalities (13) using the separation heuristic presented in Section 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 $\alpha$ 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 $\frac{(UB - LP)}{UB} \times 100\%$, 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 when adding the mixed-dicut inequalities (12) to FB. The results include the LP gap ($\%LP$) after adding inequalities (12), 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 (13), 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 FB 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, when adding the mixed-dicut inequalities (12) 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 (13), the average LP gap is 1.94% with a range from 0.00% to 5.86%. In fact, constraints (13) are able to close the optimality gap, and obtaining 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 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, number of hub facilities and considered discount factor. For instance, the LP gap is worse as the size of N and the value of $\alpha$ increases. It is interesting to observe that the number of generated cuts also depend on these parameters.

### Table 1: Comparison between CPLEX cuts and mixed-dicut inequalities

| Instance | $|N|\cdot p\cdot \alpha$ | % LP | % LP$_{cuts}$ | Cuts | CPU | % LP | Cuts | CPU | % LP | Cuts | CPU |
|----------|--------------------------|------|-------------|------|-----|------|------|-----|------|------|-----|
| 10-4-0.2 | 3.37 | 1.96 | 0.03 | 0.92 | 182 | 0.32 | 0.60 | 245 | 0.12 |
| 10-4-0.5 | 5.34 | 2.18 | 0.13 | 0.68 | 178 | 0.42 | 0.07 | 232 | 0.09 |
| 10-4-1.0 | 6.93 | 3.05 | 0.10 | 0.64 | 225 | 0.42 | 0.00 | 241 | 0.08 |
| 10-6-0.2 | 5.62 | 2.64 | 0.08 | 0.91 | 195 | 0.33 | 0.00 | 265 | 0.08 |
| 10-6-0.5 | 5.62 | 4.12 | 0.12 | 1.54 | 299 | 0.46 | 0.00 | 255 | 0.11 |
| 10-6-1.0 | 10.62 | 7.21 | 0.11 | 2.78 | 294 | 0.47 | 0.09 | 467 | 0.24 |

- Average: 0.94 | 5.96 | 411.95 | 20.46 | 2.82 | 241.31 | 83.27 | 1.94 | 4170.76 | 2719.96 |
We next compare the impact of separating both families of inequalities (12) and (13) and adding them to formulation FB at the same time. In particular, we compare the results of the following experiments:

5. We solve the LP relaxation of formulation FB and we first dynamically add constraints (12) using the separation heuristic. When no more inequalities of this type can be found, constraints (13) are then added.

6. We solve the LP relaxation of formulation FB and we first dynamically add constraints (13) using the separation heuristic. When no more inequalities of this type can be found, constraints (12) are then added.

The results of these experiments are given 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 0.00% to 5.56%, whereas in the case of $GMDI + MDI$, the average LP gap is slightly reduced to 1.68% and ranges from 0.00% to 5.43%. In 34 out of 42 instances, $GMDI + MDI$ results in lower LP gap as compared to $MDI + GMDI$. Moreover, by adding the two families of inequalities to formulation FB, we are able to obtain an integral solution from the LP relaxation in 5 instances. In general, combining both families of valid inequalities provide remarkable results in terms of the quality of the LP bounds. Although the $GMDI + MDI$ configuration provides on the average the best LP bounds, the required CPU time is considerably larger than the other configurations of the experiments. However, the configuration 5 in which constraints (12) are added and once no more violated cuts can be found, constraints (13) are added, provides the best overall results in terms of providing a good tradeoff between the quality of the obtained LP bounds and the required CPU time. Therefore, in the reminder of the experiments we will only consider this configuration.

In the second part of the computational experiments, we analyze the performance of our proposed exact and heuristic solution algorithms. In particular, we compare the quality of the obtained solutions using both constraints (12) and (13) within a branch-and-cut framework and the GRASP algorithm introduced in Section 4. These experiments are performed on the same set of instances as before (ranging from 10 to 50 nodes). Throughout this experiment, we set the maximum time limit to 86,400 seconds of CPU time. Instances that could not be solved to optimality within this time limit are marked with the label "time". Moreover, 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 detailed results are reported in Table 3. The second set of columns reports the LP gap ($\%LP$), the percent deviation between final upper and lower bound ($\%Gap$), the CPU time ($CPU$) in seconds, 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 heading Branch-and-Cut reports: $\%LP_{cuts}$ the LP bound at the root node after adding valid inequalities (12) and (13), ($\%Gap$) the final percent deviation at termination, the CPU time (CPU) in seconds, and the number of explored nodes in the branching tree (Nodes). The third 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 robust-

<table>
<thead>
<tr>
<th>Instance</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>N</td>
<td>P-\alpha$</td>
<td>MDI + GMDI</td>
<td>GMDI + MDI</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>% LP</td>
<td>#Cut1</td>
<td>% Cut2</td>
<td>CPU</td>
<td>% LP</td>
<td>#Cut1</td>
</tr>
<tr>
<td>10-4-0.2</td>
<td>0.59</td>
<td>197</td>
<td>77</td>
<td>0.46</td>
<td>0.48</td>
<td>60</td>
</tr>
<tr>
<td>10-4-0.5</td>
<td>0.00</td>
<td>178</td>
<td>78</td>
<td>0.18</td>
<td>0.00</td>
<td>6</td>
</tr>
<tr>
<td>10-4-0.6</td>
<td>0.00</td>
<td>225</td>
<td>60</td>
<td>0.19</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>10-4-0.2</td>
<td>0.00</td>
<td>195</td>
<td>54</td>
<td>0.14</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>10-4-0.2</td>
<td>0.27</td>
<td>302</td>
<td>203</td>
<td>0.65</td>
<td>0.65</td>
<td>35</td>
</tr>
<tr>
<td>20-4-0.5</td>
<td>0.99</td>
<td>1210</td>
<td>484</td>
<td>7.38</td>
<td>0.97</td>
<td>384</td>
</tr>
<tr>
<td>20-4-0.5</td>
<td>0.77</td>
<td>934</td>
<td>481</td>
<td>4.99</td>
<td>0.62</td>
<td>225</td>
</tr>
<tr>
<td>20-4-0.5</td>
<td>0.49</td>
<td>1383</td>
<td>624</td>
<td>16.11</td>
<td>0.45</td>
<td>446</td>
</tr>
<tr>
<td>20-4-0.5</td>
<td>1.54</td>
<td>1821</td>
<td>763</td>
<td>14.85</td>
<td>1.38</td>
<td>472</td>
</tr>
<tr>
<td>20-4-0.5</td>
<td>2.77</td>
<td>1583</td>
<td>1032</td>
<td>13.61</td>
<td>2.67</td>
<td>368</td>
</tr>
<tr>
<td>20-8-0.5</td>
<td>1.99</td>
<td>1582</td>
<td>816</td>
<td>28.15</td>
<td>1.75</td>
<td>443</td>
</tr>
<tr>
<td>20-8-0.5</td>
<td>4.43</td>
<td>2061</td>
<td>1356</td>
<td>26.86</td>
<td>4.27</td>
<td>418</td>
</tr>
<tr>
<td>20-8-0.5</td>
<td>3.86</td>
<td>1834</td>
<td>1598</td>
<td>19.88</td>
<td>3.73</td>
<td>308</td>
</tr>
<tr>
<td>25-4-0.5</td>
<td>0.04</td>
<td>1210</td>
<td>130</td>
<td>3.90</td>
<td>0.02</td>
<td>349</td>
</tr>
<tr>
<td>25-4-0.5</td>
<td>0.05</td>
<td>1393</td>
<td>325</td>
<td>20.38</td>
<td>0.05</td>
<td>296</td>
</tr>
<tr>
<td>25-4-0.5</td>
<td>0.54</td>
<td>1376</td>
<td>576</td>
<td>15.80</td>
<td>0.55</td>
<td>298</td>
</tr>
<tr>
<td>25-4-0.5</td>
<td>0.04</td>
<td>1539</td>
<td>136</td>
<td>7.51</td>
<td>0.01</td>
<td>156</td>
</tr>
<tr>
<td>25-4-0.5</td>
<td>0.66</td>
<td>2209</td>
<td>1042</td>
<td>26.42</td>
<td>0.69</td>
<td>529</td>
</tr>
<tr>
<td>25-4-0.5</td>
<td>2.55</td>
<td>1680</td>
<td>1212</td>
<td>32.84</td>
<td>2.45</td>
<td>460</td>
</tr>
<tr>
<td>25-8-0.5</td>
<td>2.69</td>
<td>2804</td>
<td>1324</td>
<td>37.66</td>
<td>2.50</td>
<td>938</td>
</tr>
<tr>
<td>25-8-0.5</td>
<td>2.98</td>
<td>2490</td>
<td>1846</td>
<td>55.97</td>
<td>2.88</td>
<td>540</td>
</tr>
<tr>
<td>25-8-0.5</td>
<td>3.66</td>
<td>2092</td>
<td>1579</td>
<td>45.96</td>
<td>3.51</td>
<td>382</td>
</tr>
<tr>
<td>40-4-0.5</td>
<td>0.17</td>
<td>2661</td>
<td>129</td>
<td>31.20</td>
<td>0.04</td>
<td>985</td>
</tr>
<tr>
<td>40-4-0.5</td>
<td>0.55</td>
<td>3131</td>
<td>778</td>
<td>155.61</td>
<td>0.46</td>
<td>1025</td>
</tr>
<tr>
<td>40-4-0.5</td>
<td>1.26</td>
<td>3609</td>
<td>1728</td>
<td>370.22</td>
<td>1.27</td>
<td>1061</td>
</tr>
<tr>
<td>40-4-0.5</td>
<td>1.33</td>
<td>5104</td>
<td>184</td>
<td>112.62</td>
<td>1.01</td>
<td>1815</td>
</tr>
<tr>
<td>40-4-0.5</td>
<td>2.97</td>
<td>6209</td>
<td>2228</td>
<td>692.53</td>
<td>2.69</td>
<td>1719</td>
</tr>
<tr>
<td>40-4-0.5</td>
<td>2.77</td>
<td>4478</td>
<td>4310</td>
<td>1365.29</td>
<td>2.78</td>
<td>1190</td>
</tr>
<tr>
<td>40-8-0.5</td>
<td>2.57</td>
<td>6868</td>
<td>2912</td>
<td>704.81</td>
<td>2.52</td>
<td>1654</td>
</tr>
<tr>
<td>40-8-0.5</td>
<td>4.75</td>
<td>5516</td>
<td>4258</td>
<td>1590.11</td>
<td>4.53</td>
<td>1464</td>
</tr>
<tr>
<td>40-8-0.5</td>
<td>3.52</td>
<td>4675</td>
<td>5210</td>
<td>1945.96</td>
<td>3.55</td>
<td>1122</td>
</tr>
<tr>
<td>50-4-0.5</td>
<td>0.03</td>
<td>2315</td>
<td>179</td>
<td>61.12</td>
<td>0.04</td>
<td>533</td>
</tr>
<tr>
<td>50-4-0.5</td>
<td>0.19</td>
<td>3864</td>
<td>889</td>
<td>505.81</td>
<td>0.22</td>
<td>1071</td>
</tr>
<tr>
<td>50-4-0.5</td>
<td>1.34</td>
<td>4430</td>
<td>1874</td>
<td>1391.38</td>
<td>1.36</td>
<td>1142</td>
</tr>
<tr>
<td>50-4-0.5</td>
<td>0.80</td>
<td>5878</td>
<td>1403</td>
<td>822.11</td>
<td>0.73</td>
<td>2110</td>
</tr>
<tr>
<td>50-4-0.5</td>
<td>3.75</td>
<td>6649</td>
<td>4312</td>
<td>4571.24</td>
<td>3.72</td>
<td>2092</td>
</tr>
<tr>
<td>50-4-0.5</td>
<td>3.71</td>
<td>5784</td>
<td>6134</td>
<td>7365.06</td>
<td>3.30</td>
<td>1481</td>
</tr>
<tr>
<td>50-8-0.5</td>
<td>3.05</td>
<td>8113</td>
<td>3807</td>
<td>3225.68</td>
<td>3.05</td>
<td>2635</td>
</tr>
<tr>
<td>50-8-0.5</td>
<td>5.56</td>
<td>7730</td>
<td>6523</td>
<td>8127.63</td>
<td>5.43</td>
<td>2134</td>
</tr>
<tr>
<td>50-8-0.5</td>
<td>4.88</td>
<td>6182</td>
<td>8712</td>
<td>12347.64</td>
<td>5.00</td>
<td>1331</td>
</tr>
<tr>
<td>Average</td>
<td>1.26</td>
<td>2958.38</td>
<td>1793.14</td>
<td>1089.69</td>
<td>1.68</td>
<td>800.60</td>
</tr>
</tbody>
</table>
ness is measured by using the average percent deviation (%Avg Dev) using the best solutions obtained in each of the 30 runs. The average CPU time in seconds across all the runs of the GRASP is also reported.

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

<table>
<thead>
<tr>
<th>Instance</th>
<th>% LP</th>
<th>% Gap</th>
<th>CPU</th>
<th>%LPcut</th>
<th>%Gap</th>
<th>CPU</th>
<th>% Dev</th>
<th>% Avg</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-P-α</td>
<td>20-0.2</td>
<td>1.70</td>
<td>0.00</td>
<td>3.78</td>
<td>0.04</td>
<td>0.00</td>
<td>0.68</td>
<td>3</td>
<td>0.00</td>
</tr>
<tr>
<td>10-0.5</td>
<td>5.34</td>
<td>0.00</td>
<td>0.67</td>
<td>43</td>
<td>0.80</td>
<td>0.00</td>
<td>0.37</td>
<td>8</td>
<td>0.00</td>
</tr>
<tr>
<td>10-0.8</td>
<td>6.93</td>
<td>0.00</td>
<td>0.97</td>
<td>16</td>
<td>1.24</td>
<td>0.00</td>
<td>0.47</td>
<td>15</td>
<td>0.00</td>
</tr>
<tr>
<td>10-0.2</td>
<td>5.62</td>
<td>0.00</td>
<td>1.14</td>
<td>73</td>
<td>3.83</td>
<td>0.00</td>
<td>0.40</td>
<td>64</td>
<td>0.00</td>
</tr>
<tr>
<td>10-0.5</td>
<td>8.58</td>
<td>0.00</td>
<td>1.26</td>
<td>261</td>
<td>4.10</td>
<td>0.00</td>
<td>1.08</td>
<td>294</td>
<td>0.00</td>
</tr>
<tr>
<td>10-0.8</td>
<td>10.62</td>
<td>0.00</td>
<td>1.54</td>
<td>877</td>
<td>5.66</td>
<td>0.00</td>
<td>2.01</td>
<td>668</td>
<td>0.00</td>
</tr>
<tr>
<td>20-0.2</td>
<td>1.70</td>
<td>0.00</td>
<td>3.78</td>
<td>36</td>
<td>0.04</td>
<td>0.00</td>
<td>0.68</td>
<td>3</td>
<td>0.00</td>
</tr>
<tr>
<td>20-0.5</td>
<td>4.33</td>
<td>0.00</td>
<td>17.69</td>
<td>485</td>
<td>1.36</td>
<td>0.00</td>
<td>5.16</td>
<td>54</td>
<td>0.00</td>
</tr>
<tr>
<td>20-0.8</td>
<td>5.11</td>
<td>0.00</td>
<td>29.16</td>
<td>1024</td>
<td>0.91</td>
<td>0.00</td>
<td>3.46</td>
<td>43</td>
<td>0.00</td>
</tr>
<tr>
<td>20-0.2</td>
<td>5.60</td>
<td>0.00</td>
<td>30.48</td>
<td>753</td>
<td>0.83</td>
<td>0.00</td>
<td>8.30</td>
<td>85</td>
<td>0.00</td>
</tr>
<tr>
<td>20-0.5</td>
<td>8.26</td>
<td>0.00</td>
<td>350.89</td>
<td>9093</td>
<td>2.08</td>
<td>0.00</td>
<td>24.15</td>
<td>706</td>
<td>0.00</td>
</tr>
<tr>
<td>20-0.8</td>
<td>9.68</td>
<td>0.00</td>
<td>1127.92</td>
<td>32563</td>
<td>3.09</td>
<td>0.00</td>
<td>159.39</td>
<td>5422</td>
<td>0.00</td>
</tr>
<tr>
<td>20-0.2</td>
<td>7.35</td>
<td>0.00</td>
<td>180.73</td>
<td>4837</td>
<td>2.18</td>
<td>0.00</td>
<td>38.65</td>
<td>1177</td>
<td>0.00</td>
</tr>
<tr>
<td>20-0.5</td>
<td>12.84</td>
<td>0.00</td>
<td>2738.87</td>
<td>58631</td>
<td>4.93</td>
<td>0.00</td>
<td>1569.60</td>
<td>20306</td>
<td>0.00</td>
</tr>
<tr>
<td>20-0.8</td>
<td>12.66</td>
<td>0.00</td>
<td>8516.44</td>
<td>245771</td>
<td>4.36</td>
<td>0.00</td>
<td>2119.75</td>
<td>24271</td>
<td>0.00</td>
</tr>
<tr>
<td>25-0.2</td>
<td>1.79</td>
<td>0.00</td>
<td>13.65</td>
<td>66</td>
<td>0.26</td>
<td>0.00</td>
<td>3.00</td>
<td>19</td>
<td>0.00</td>
</tr>
<tr>
<td>25-0.5</td>
<td>3.15</td>
<td>0.00</td>
<td>46.69</td>
<td>248</td>
<td>0.28</td>
<td>0.00</td>
<td>5.67</td>
<td>16</td>
<td>0.00</td>
</tr>
<tr>
<td>25-0.8</td>
<td>4.50</td>
<td>0.00</td>
<td>118.65</td>
<td>928</td>
<td>0.81</td>
<td>0.00</td>
<td>12.91</td>
<td>50</td>
<td>0.00</td>
</tr>
<tr>
<td>25-0.2</td>
<td>3.46</td>
<td>0.00</td>
<td>32.26</td>
<td>312</td>
<td>0.20</td>
<td>0.00</td>
<td>5.86</td>
<td>22</td>
<td>0.00</td>
</tr>
<tr>
<td>25-0.5</td>
<td>6.35</td>
<td>0.00</td>
<td>327.76</td>
<td>3435</td>
<td>1.20</td>
<td>0.00</td>
<td>20.35</td>
<td>99</td>
<td>0.00</td>
</tr>
<tr>
<td>25-0.8</td>
<td>8.87</td>
<td>0.00</td>
<td>4065.04</td>
<td>37955</td>
<td>2.62</td>
<td>0.00</td>
<td>239.53</td>
<td>2229</td>
<td>0.00</td>
</tr>
<tr>
<td>25-0.2</td>
<td>7.51</td>
<td>0.00</td>
<td>3169.96</td>
<td>22421</td>
<td>3.01</td>
<td>0.00</td>
<td>340.35</td>
<td>2553</td>
<td>0.00</td>
</tr>
<tr>
<td>25-0.5</td>
<td>10.12</td>
<td>0.00</td>
<td>7711.25</td>
<td>68475</td>
<td>3.37</td>
<td>0.00</td>
<td>1127.22</td>
<td>6327</td>
<td>0.00</td>
</tr>
<tr>
<td>25-0.8</td>
<td>11.09</td>
<td>0.00</td>
<td>18229.57</td>
<td>152835</td>
<td>4.04</td>
<td>0.00</td>
<td>5854.66</td>
<td>14789</td>
<td>0.00</td>
</tr>
</tbody>
</table>

The results in Table 3 show that by using formulation FB 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.
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 \((n = 50, p = 6, \alpha = 0.5)\), 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.

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.

<table>
<thead>
<tr>
<th>Instance ((N))</th>
<th>CPLEX</th>
<th>Branch-and-Cut</th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>60-4-0.2</td>
<td>1.69</td>
<td>0.96</td>
<td>242.93</td>
</tr>
<tr>
<td>60-4-0.5</td>
<td>2.99</td>
<td>1.09</td>
<td>760.68</td>
</tr>
<tr>
<td>60-4-0.8</td>
<td>5.41</td>
<td>2.10</td>
<td>8850.31</td>
</tr>
<tr>
<td>60-6-0.2</td>
<td>3.84</td>
<td>2.42</td>
<td>16838.45</td>
</tr>
<tr>
<td>60-6-0.5</td>
<td>7.54</td>
<td>4.86</td>
<td>3.45</td>
</tr>
<tr>
<td>70-4-0.2</td>
<td>1.57</td>
<td>0.83</td>
<td>729.41</td>
</tr>
<tr>
<td>70-4-0.5</td>
<td>3.33</td>
<td>1.43</td>
<td>4315.38</td>
</tr>
<tr>
<td>70-4-0.8</td>
<td>5.59</td>
<td>2.58</td>
<td>45543.09</td>
</tr>
<tr>
<td>70-6-0.2</td>
<td>3.88</td>
<td>2.05</td>
<td>17063.47</td>
</tr>
<tr>
<td>70-6-0.5</td>
<td>7.83</td>
<td>5.43</td>
<td>862</td>
</tr>
<tr>
<td>75-4-0.2</td>
<td>1.52</td>
<td>1.06</td>
<td>1320.45</td>
</tr>
<tr>
<td>75-4-0.5</td>
<td>3.40</td>
<td>1.56</td>
<td>8595.06</td>
</tr>
<tr>
<td>75-4-0.8</td>
<td>5.64</td>
<td>2.54</td>
<td>1727</td>
</tr>
<tr>
<td>75-6-0.2</td>
<td>3.94</td>
<td>2.58</td>
<td>2666</td>
</tr>
<tr>
<td>75-6-0.5</td>
<td>7.61</td>
<td>5.42</td>
<td>766</td>
</tr>
<tr>
<td>90-4-0.2</td>
<td>1.43</td>
<td>1.35</td>
<td>9477.96</td>
</tr>
<tr>
<td>90-4-0.5</td>
<td>3.14</td>
<td>1.93</td>
<td>65523.91</td>
</tr>
<tr>
<td>90-4-0.8</td>
<td>5.40</td>
<td>3.28</td>
<td>165</td>
</tr>
<tr>
<td>90-6-0.2</td>
<td>3.91</td>
<td>3.34</td>
<td>641</td>
</tr>
<tr>
<td>90-6-0.5</td>
<td>7.63</td>
<td>5.98</td>
<td>267</td>
</tr>
<tr>
<td>100-4-0.2</td>
<td>1.90</td>
<td>1.39</td>
<td>22302.74</td>
</tr>
<tr>
<td>100-4-0.5</td>
<td>3.24</td>
<td>2.26</td>
<td>706</td>
</tr>
<tr>
<td>100-4-0.8</td>
<td>5.52</td>
<td>3.41</td>
<td>29</td>
</tr>
<tr>
<td>100-6-0.2</td>
<td>4.12</td>
<td>3.73</td>
<td>605</td>
</tr>
</tbody>
</table>

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 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%.
6 Conclusions

In this paper, we studied the cycle hub location problem. Potential applications of this model appear in telecommunication and transportation systems, where large set-up costs on the links as well as reliability requirements make cycle topologies a prominent network structure. We presented two solution approaches: a branch-and-cut based exact approach and a heuristic 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. These valid inequalities were embedded into a branch-and-cut framework to improve the lower bound at some nodes of the enumeration tree. A GRASP meta-heuristic was also presented to efficiently obtain high quality solutions. Computational results on benchmark instances with up to 100 nodes confirm the efficiency and robustness of the proposed algorithms.

7 Acknowledgments

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.

References

Labbé M, Laporte G, Rodríguez-Martín I (1998) Path, tree and cycle location. Springer
Martins de Sá E, Contreras I, Cordeau JF (2014) Exact and heuristic algorithms for the design of hub networks with multiple lines. Submitted to European Journal of Operational Research