Math. Program., Ser. A 102: 371–405 (2005)

Digital Object Identifier (DOI) 10.1007/s10107-004-0531-x

Martine Labbé · Hande Yaman · Eric Gourdin

A branch and cut algorithm for hub location problems with single assignment

Received: April 14, 2003 / Accepted: April 30, 2004
Published online: 7 July 2004 – © Springer-Verlag 2004

Abstract. The hub location problem with single assignment is the problem of locating hubs and assigning the terminal nodes to hubs in order to minimize the cost of hub installation and the cost of routing the traffic in the network. There may also be capacity restrictions on the amount of traffic that can transit by hubs. The aim of this paper is to investigate polyhedral properties of these problems and to develop a branch and cut algorithm based on these results.

Key words. Hub location – Polyhedral analysis – Branch and cut

1. Introduction

Hubs (concentrators, routers, multiplexers ...) are special nodes that route the traffic in a communication or transportation network. Given a set of terminals (user nodes) and a traffic matrix, the hub location problem consists in choosing a subset of terminals to locate hubs and assigning the terminals to hub nodes in order to minimize the cost of hub location and traffic routing. These problems have applications in transportation and telecommunication (see the survey by Campbell et al. [7]).

Hub location problems can be classified regarding the way terminals are assigned to hub nodes. In multiple assignment problems, the traffic of a same terminal can be routed through different hubs. On the contrary, in single assignment problems, each terminal is assigned to a single hub. We consider hub location problems with single assignment. We study several versions with different capacity restrictions. The principal problem considered is as follows. We are given a set of terminal nodes and a traffic matrix whose entries represent the amount of traffic to be routed between every pair of terminals. We must determine a subset of the terminal nodes to be the hub locations. If a node receives a hub then it is assigned to itself. Each node that does not receive a hub is assigned to exactly one hub node.

Here we adopt the terminology of telecommunication networks. The network connecting the hubs is called the *backbone network* and it is complete. The networks that connect the terminals to hubs are the *access networks* and they are stars. Figure 1 depicts

M. Labbé: Université Libre de Bruxelles, Service d'Optimisation, Boulevard du Triomphe CP 210/01, 1050 Bruxelles, Belgium. e-mail: mlabbe@smg.ulb.ac.be

H. Yaman: Bilkent University, Department of Industrial Engineering, Bilkent, 06800 Ankara, Turkey. e-mail: hyaman@bilkent.edu.tr

E. Gourdin: France Telecom R&D, DAC/OAT, 38-40, rue du General Leclerc, Issy-les-Moulineaux Cedex 9, 92794, France. e-mail: eric.gourdin@rd.francetelecom.com

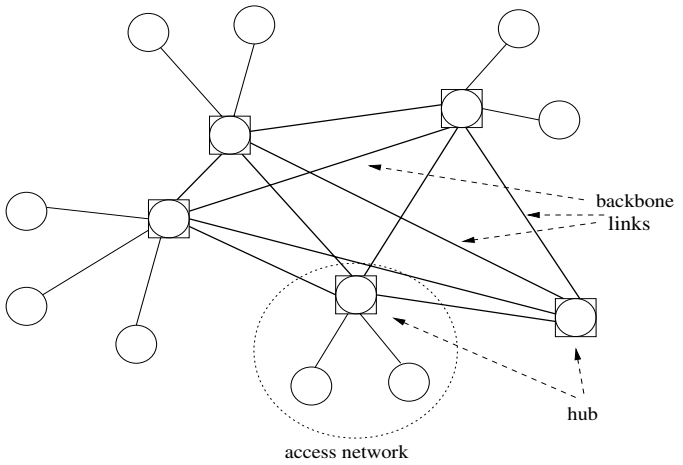

Fig. 1. A network with complete backbone and star access networks

a network with a complete backbone and star access networks. The nodes inside squares are chosen to be the hub nodes.

The traffic between two nodes is routed on the shortest path between these nodes. We assume that the cost of routing the traffic satisfies the triangle inequality. If two nodes are assigned to the same hub, the traffic between them does not enter the backbone network. So the traffic routed on the backbone links is the traffic of nodes that are assigned to different hubs. More specifically, the traffic on a backbone link from node j to node l is the sum of the traffic between all nodes assigned to node j and all nodes assigned to node l.

A hub has a fixed capacity in terms of the traffic that passes through it. This traffic is the sum of the traffic on the links between this hub and the nodes that are assigned to it and the traffic on the backbone links between this hub and the other hubs.

There is a fixed cost for installing a hub at a certain node and a cost per unit of traffic routed between two given nodes. The aim is to locate the hubs and to assign the remaining nodes to the hub nodes to minimize the total cost of installation and routing. We call this problem the *Quadratic Capacitated Hub Location Problem with Single Assignment* and abbreviate it by *QHL*.

We consider two relaxations of QHL based on the capacities. The first relaxation is the *Linear Capacitated Hub Location Problem with Single Assignment (LHL)* where each hub has a fixed capacity in terms of the traffic adjacent at nodes (the sum of the traffic of commodities having this node either as origin or destination) that are assigned to it (and not on the traffic between this hub and the other hubs). The second relaxation is obtained by removing the capacity constraints and it is called the *Uncapacitated Hub Location Problem with Single Assignment (UHL)*.

The aim of this paper is to study the polyhedral properties of hub location problems with single assignment and to develop a branch and cut algorithm to solve such problems. We present our results and the branch and cut algorithm for QHL as it generalizes UHL and LHL.

To our knowledge, there is no previous study of QHL. But location and network design problems arising in telecommunications have long been studied by both oper-

ations researchers and computer scientists (see e.g. [3], [12] and [21]). The interested reader can refer to [11] for a survey of location problems arising in telecommunications and to [24] for an annotated bibliography in network design.

Also, the formulations for LHL can be extended to QHL (see Section 5). For a survey of formulations and solution methods for LHL and UHL see Campbell et al. [7].

The paper is organized as follows: In Section 2, we give a formal definition and a formulation of QHL. In Section 3, we discuss the variants of QHL based on the capacity restrictions. Section 4 is devoted to the polyhedral analysis and valid inequalities. We present our branch and cut algorithm and the computational results in Section 5.

2. Problem definition and formulation for QHL

Let I denote the set of terminal nodes with $|I| = n$ and K the set of commodities. For commodity $k \in K$, $o(k)$ is the origin, $d(k)$ is the destination and t^k is the amount of traffic where $t^k = t_{o(k)d(k)}$. We define $T_{im} = t_{im} + t_{mi}$ for all $(i, m) \in K$. Origins and destinations of commodities are terminal nodes and any pair of terminal nodes defines a commodity. The values t_{ii}'s are defined to be 0 for all $i \in I$.

Each terminal either receives a hub or is connected to another node which receives a hub. Let a_i be the total traffic adjacent at node i, $a_i = \sum_{m \in I} T_{im}$. The results presented throughout the paper remain valid if $a_i \geq \sum_{m \in I} T_{im}$ for all $i \in I$.

If node i is assigned to concentrator node $j \neq i$, then the traffic on the link between nodes i and j is a_i. The cost of routing traffic a_i on the link between node i and node j is denoted by C_{ij}. Any node i that becomes a hub is assigned to itself. The cost of installing a hub at node i is denoted by C_{ii}.

We define the arc set $A = \{(j, l) : j \in I, l \in I, j \neq l\}$. We denote by R_{jl} the cost of routing a traffic unit on arc (j, l) if it becomes a backbone arc, i.e., if both nodes j and l receive hubs. We assume that the cost vector R satisfies the triangle inequality and $R_{jl} \geq 0$ for all $(j, l) \in A$. It is likely that in most real applications, one has $R_{jl} \leq C_{jl}/a_j$. However, note that all the results presented in this paper hold even if this were not the case.

If nodes j and l receive hubs, then the amount of flow on arc (j, l) is given by $\sum_{k \in K_{jl}} t^k$ where K_{jl} is the set of commodities k such that $o(k)$ is assigned to j and $d(k)$ is assigned to l.

If node j becomes a hub node, then the total amount of traffic transiting through node j cannot be larger than its capacity M. This total amount of traffic is equal to the sum of the amounts of traffic adjacent to nodes assigned to hub j (including j) and the amounts of traffic on the backbone arcs (j, l) leaving j and the backbone arcs (l, j) entering j.

Our mixed integer programming formulation of QHL uses two types of variables. The assignment variable $x_{ij} = 1$ if terminal $i \in I$ is assigned to hub $j \in I$ and 0 otherwise. If node i receives a hub then x_{ii} takes value 1 and node i is assigned to itself. Further, we define z_{jl} to be the total traffic on the arc $(j, l) \in A$.

Now, we can present our formulation:

$$\min \sum_{i\in I}\sum_{j\in I} C_{ij}x_{ij} + \sum_{(j,l)\in A} R_{jl}z_{jl} \tag{1}$$

$$\text{s.t.} \sum_{j\in I} x_{ij} = 1 \qquad \forall i \in I \tag{2}$$

$$x_{ij} \le x_{jj} \qquad \forall (i,j) \in A \tag{3}$$

$$z_{jl} \ge \sum_{k\in K'} t^k (x_{o(k)j} + x_{d(k)l} - 1) \qquad \forall (j,l) \in A,\ K' \subseteq K \tag{4}$$

$$\sum_{i\in I} a_i x_{ij} + \sum_{i\in I}\sum_{m\in I} T_{im}x_{ij}(1 - x_{mj}) \le M x_{jj} \qquad \forall j \in I \tag{5}$$

$$z_{jl} \ge 0 \qquad \forall (j,l) \in A \tag{6}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I,\ j \in I. \tag{7}$$

Constraints (2) and (7) ensure that each terminal either becomes a hub location or is assigned to exactly one other node. Constraints (3) state that if a terminal is assigned to a node then this node should be a hub location. Constraints (4) relate the traffic vector z to the assignment vector x. Consider arc $(j,l) \in A$ with $R_{jl} > 0$. Constraints (4) and (7) imply

$$z_{jl} = \max_{K'\subseteq K} \sum_{k\in K'} t^k (x_{o(k)j} + x_{d(k)l} - 1) = \sum_{k\in K_{jl}} t^k (x_{o(k)j} + x_{d(k)l} - 1).$$

Capacity constraints (5) ensure that if we install a hub at node j, then the traffic that passes through node j does not exceed the capacity M.

Note that these constraints are written in the quadratic form. A linear version which will be used in the implementation is the following:

$$\sum_{i\in I} a_i x_{ij} + \sum_{l\in I\setminus\{j\}} (z_{lj} + z_{jl}) \le M x_{jj} \qquad \forall j \in I. \tag{8}$$

Inequalities (4) imply that $\sum_{l\in I\setminus\{j\}}(z_{lj} + z_{jl}) \ge \sum_{i\in I}\sum_{m\in I} T_{im}x_{ij}(1 - x_{mj})$. As $R_{jl} \ge 0$ for all $(j,l) \in A$, for any feasible x there exists a feasible z for which the inequality is tight, implying the quadratic capacity constraints (5).

Let F_{QH} denote the set of points (x, z) that satisfy constraints (2)–(7).

3. Variants of QHL

We first present two relaxations based on the capacity constraints.

Uncapacitated Case. If it is possible to assign all nodes to a single hub, i.e., $\sum_{i\in I} a_i \le M$ then the problem is uncapacitated. Define F_{UH} to be the set of pairs (x, z) that satisfy the constraints (2)–(4), (6) and (7).

Linear Capacitated Case. If the capacity of a hub concerns only the traffic adjacent to nodes assigned to it, then constraints (5) should be replaced by

$$\sum_{i \in I} a_i x_{ij} \leq M x_{jj} \qquad \forall j \in I. \tag{9}$$

Let F_{LH} be the set of (x, z) that satisfy constraints (2)–(4), (6), (7) and (9).

Another type of relaxation corresponds to the situation where the routing cost is negligible.

Concentrator Location Problem. If the routing costs on the backbone network are equal to zero, then we have a pure location problem. We can remove variables z_{jl} and the related constraints from formulation QHL and obtain:

$$\min \sum_{i \in I} \sum_{j \in I} C_{ij} x_{ij}$$

$$\text{s.t. (2), (3), (5) and (7).}$$

Let F_Q be the set of vectors x that satisfy constraints (2), (3), (5) and (7). We also consider the two relaxations based on the capacities.

Table 1 gives the names and abbreviations of the six problems with reference to their capacity structures and routing costs.

4. Polyhedral analysis and valid inequalities

In the sequel, we assume that any two nodes can be assigned together, which implies that $a_i + a_j + \sum_{m \in I \setminus \{i,j\}} (T_{im} + T_{jm}) \leq M$ for all $(i, j) \in A$ and that $a_i \geq \sum_{m \in I} T_{im}$ for all $i \in I$.

Let $e^x_{ij} = (x, z)$ (resp. $e^z_{ij} = (x, z)$) be the unit vector such that $x_{lm} = 0$ for all $(l, m) \in A \setminus \{(i, j)\}$, $x_{ij} = 1$ and $z_{lm} = 0$ for all $(l, m) \in A$ (resp. $x_{lm} = 0$ for all $(l, m) \in A$, $z_{lm} = 0$ for all $(l, m) \in A \setminus \{(i, j)\}$ and $z_{ij} = 1$).

Define $Proj_x(F)$ to be the projection of set F on the x space.

To make the polyhedral analysis easier, we eliminate the x_{jj} variables by substituting $x_{jj} = 1 - \sum_{i \in I \setminus \{j\}} x_{ji}$ for all $j \in I$ (see Avella and Sassano [4]). Then we obtain the following formulation for QHL:

$$\min \sum_{i \in I} \sum_{j \in I \setminus \{i\}} C_{ij} x_{ij} + \sum_{j \in I} C_{jj} \Big(1 - \sum_{i \in I \setminus \{j\}} x_{ji}\Big) + \sum_{(j,l) \in A} R_{jl} z_{jl}$$

s.t.

$$x_{ij} + \sum_{m \in I \setminus \{j\}} x_{jm} \leq 1 \qquad \forall (i, j) \in A \tag{10}$$

$$\sum_{i \in I \setminus \{j\}} (a_i - T_{ij}) x_{ij} + \sum_{i \in I \setminus \{j\}} \sum_{m \in I \setminus \{j\}} T_{im} x_{ij} (1 - x_{mj})$$
$$+ (M - a_j - \sum_{i \in I} T_{ij}) \sum_{m \in I \setminus \{j\}} x_{jm} \leq M - a_j - \sum_{i \in I} T_{ij} \quad \forall j \in I \tag{11}$$

Table 1. Concentrator location and hub location problems and polyhedra

	no routing cost	routing on a complete backbone
no capacity constraints	Uncapacitated Concentrator Location Problem (UCL) $P_U^{'} = conv(\{x \in \{0,1\}^{n(n-1)} : (10)\})$	Uncapacitated Hub Location Problem (UHL) $P_{UH}^{'} = conv(\{(x,z) \in \{0,1\}^{n(n-1)} \times \mathbb{R}_+^{n(n-1)} : (10) \text{ and } (12)\})$
linear capacity constraints	Linear Capacitated Concentrator Location Problem (LCL) $P_L^{'} = conv(\{x \in \{0,1\}^{n(n-1)} : (10) \text{ and } (15)\})$	Linear Capacitated Hub Location Problem (LHL) $P_{LH}^{'} = conv(\{(x,z) \in \{0,1\}^{n(n-1)} \times \mathbb{R}_+^{n(n-1)} : (10), (15) \text{ and } (12)\})$
quadratic capacity constraints	Quadratic Capacitated Concentrator Location Problem (QCL) $P_Q^{'} = conv(\{x \in \{0,1\}^{n(n-1)} : (10) \text{ and } (11)\})$	Quadratic Capacitated Hub Location Problem (QHL) $P_{QH}^{'} = conv(\{(x,z) \in \{0,1\}^{n(n-1)} \times \mathbb{R}_+^{n(n-1)} : (10), (11) \text{ and } (12)\})$

$$z_{jl} \geq \sum_{k \in K', o(k) \neq j,l, d(k) \neq j,l} t^k (x_{o(k)j} + x_{d(k)l} - 1)$$
$$+ \sum_{i \in I \setminus \{j,l\}:(j,i) \in K'} t_{ji}(x_{il} - \sum_{m \in I \setminus \{j\}} x_{jm}) + \sum_{i \in I \setminus \{j,l\}:(i,l) \in K'} t_{il}(x_{ij} - \sum_{m \in I \setminus \{j\}} x_{lm})$$
$$+ t_{jl}(1 - \sum_{m \in I \setminus \{j\}} x_{jm} - \sum_{m \in I \setminus \{j\}} x_{lm}) \qquad \forall K' \subseteq K, (j,l) \in A \qquad (12)$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \qquad (13)$$
$$z_{jl} \geq 0 \qquad \forall (j,l) \in A. \qquad (14)$$

Let $F_{QH}^{'} = \{(x,z) \in \{0,1\}^{n(n-1)} \times \mathbb{R}^{n(n-1)} : (x,z) \text{ satisfies (10)–(14)}\}$ and $P_{QH}^{'} = conv(F_{QH}^{'})$. We do the same projection for all six problems (see Table 1). The linear capacity constraints (9) become:

$$\sum_{i \in I \setminus \{j\}} a_i x_{ij} + (M - a_j) \sum_{i \in I \setminus \{j\}} x_{ji} \leq M - a_j \quad \forall j \in I. \qquad (15)$$

Proposition 1. *$Proj_x(P_{YH}^{'}) = P_Y^{'}$ for $Y \in \{U, L, Q\}$.*

This projection property will be used to relate some of the facets of the polyhedra of hub location problems to the facets of the polytopes of the concentrator location problems.

Proposition 2. *The polytopes $P_U^{'}$, $P_L^{'}$ and $P_Q^{'}$ and the polyhedra $P_{UH}^{'}$, $P_{LH}^{'}$ and $P_{QH}^{'}$ are full dimensional, i.e., $dim(P_U^{'}) = dim(P_L^{'}) = dim(P_Q^{'}) = n(n-1)$ and $dim(P_{UH}^{'}) = dim(P_{LH}^{'}) = dim(P_{QH}^{'}) = 2n(n-1)$.*

Proof. Let $Y \in \{U, L, Q\}$. Assume that all points $(x,z) \in P_{YH}^{'}$ satisfy an equality $\alpha x + \beta z = \gamma$. Choose an arc $(j,l) \in A$ and a point $p = (x,z) \in P_{YH}^{'}$. As $p + e_{jl}^z$ is in $P_{YH}^{'}$, we have $\beta_{jl} = 0$ for all $(j,l) \in A$. Now consider a point $p = \sum_{(j,l) \in A} N e_{jl}^z$ for

large enough N. As p is in P'_{YH}, $\alpha x = 0 = \gamma$. Finally, since for $(i, j) \in A$, the vector $p = e^x_{ij} + \sum_{(l,m)\in A} N e^z_{lm}$ is in P'_{YH}, we obtain $\alpha_{ij} = \gamma = 0$. The proof for P_Y can be done similarly. □

4.1. Facet Defining Inequalities Involving only the Assignment Variables

We give a characterization of facet defining inequalities which involve only the assignment variables.

Theorem 1. *For $Y \in \{U, L, Q\}$, an inequality $\alpha x \leq \alpha_0$ defines a facet of P'_{YH} if and only if it defines a facet of P'_Y.*

Proof. Assume that $\pi x \leq \pi_0$ defines a facet of P'_Y and that all points $(x, z) \in P'_{YH}$ such that $\pi x = \pi_0$ also satisfy $\alpha x + \beta z = \gamma$. Since for any point $p = (x, z) \in P'_{YH}$ we have that $p + e^z_{jl}$ is in P'_{YH}, it follows that $\beta_{jl} = 0$ for all $(j, l) \in A$. As $P'_Y = Proj_x(P'_{YH})$ and as P'_Y and P'_{YH} are full dimensional, $\alpha = \lambda\pi$ and $\gamma = \lambda\pi_0$ for some $\lambda \neq 0$.

Assume that $\pi x \leq \pi_0$ defines a facet of P'_{YH} and that any $x \in P'_Y$ such that $\pi x = \pi_0$ also satisfies $\alpha x = \gamma$. As $P'_Y = Proj_x(P'_{YH})$, any $(x, z) \in P'_{YH}$ such that $\pi x = \pi_0$ also satisfies $\alpha x = \gamma$. This proves that $\alpha = \lambda\pi$ and $\gamma = \lambda\pi_0$ for some $\lambda \neq 0$ since P'_{YH} is full dimensional. Thus inequality $\pi x \leq \pi_0$ defines a facet of P'_Y. □

The polyhedral properties of concentrator location problems are studied in Yaman [23]. We will use the valid inequalities given in [23] in the branch and cut algorithm.

4.2. Facet Defining Inequalities Involving only the Traffic Variables

Now we present some properties of the facet defining inequalities which involve only the traffic variables.

Proposition 3. *Let $\beta z \geq \beta_0$ be a facet defining inequality for P'_{YH} for $Y \in \{U, L, Q\}$. Then $\beta \geq 0$ and $\beta_0 \geq 0$. Let $A^+ = \{(j, l) \in A : \beta_{jl} > 0\}$. If there exists $I^+ \subseteq I$ such that (i) there is $x \in P'_{YH}$ with $\sum_{i\in I^+} \sum_{j\in I\setminus I^+} x_{ij} = |I^+|$ and (ii) for every $(j, l) \in A^+$ either $j \in I^+$ or $l \in I^+$, then $\beta z \geq \beta_0$ is a positive multiple of $z_{jl} \geq 0$ for some $(j, l) \in A^+$.*

Proof. Assume that there exists $(j, l) \in A$ such that $\beta_{jl} < 0$. Let $(x, z) \in P'_{YH}$ be such that $\beta z = \beta_0$. Consider $(x', z') = (x, z) + e^z_{jl}$. Then $(x', z') \in P'_{YH}$ and $\beta z' < \beta_0$ imply that inequality $\beta z \geq \beta_0$ cannot be valid. This shows that $\beta \geq 0$. As $\beta \geq 0$ and $z \geq 0$, we should have $\beta_0 \geq 0$.

If there exists $I^+ \subseteq I$ which satisfies conditions (i) and (ii), then there exists $(x, z) \in P'_{YH}$ where at least one extremity node of each arc $(j, l) \in A^+$ is not a hub in x and so $z_{jl} = 0$ for all $(j, l) \in A^+$. Thus $\beta_0 = 0$. Now, if $\beta z \geq 0$ is facet defining then it must be a positive multiple of $z_{jl} \geq 0$ for some $(j, l) \in A^+$. □

If inequality $z_{jl} \geq \beta_0$ is a valid inequality for P_{YH}, then as set $I^+ = \{j\}$ satisfies conditions (i) and (ii), $\beta_0 = 0$. So variables z_{jl}'s do not admit positive fixed lower bounds.

Proposition 4. *For $(j, l) \in A$, if $t_{jl} = 0$ and $\{j, l, m\}$ can be assigned together for all $m \in I \setminus \{j, l\}$, then $z_{jl} \geq 0$ defines a facet of P'_{YH} for $Y \in \{U, L, Q\}$.*

Proof. For $(j, l) \in A$, assume that $t_{jl} = 0$ and $\{j, l, m\}$ can be assigned together for all $m \in I \setminus \{j, l\}$. Let N denote a very large number. The following $2n(n-1)$ points are in P'_{YH}, satisfy $z_{jl} = 0$ and are clearly affinely independent:

- $\sum_{(t,s)\in A\setminus\{(j,l)\}} Ne^z_{ts}$
- for $(i, m) \in A$ such that $m \in I \setminus \{j, l\}$, $e^x_{im} + \sum_{(t,s)\in A\setminus\{(j,l)\}} Ne^z_{ts}$
- for $i \in I \setminus \{j, l\}$, $e^x_{ij} + e^x_{lj} + \sum_{(t,s)\in A\setminus\{(j,l)\}} Ne^z_{ts}$
- for $i \in I \setminus \{j, l\}$, $e^x_{il} + e^x_{jl} + \sum_{(t,s)\in A\setminus\{(j,l)\}} Ne^z_{ts}$
- $e^x_{lj} + \sum_{(t,s)\in A\setminus\{(j,l)\}} Ne^z_{ts}$
- for $(i, m) \in A$ such that $m \in I \setminus \{j, l\}$, $e^x_{im} + \sum_{(t,s)\in A\setminus\{(j,l),(i,m)\}} Ne^z_{ts}$
- for $i \in I \setminus \{j, l\}$, $e^x_{ij} + e^x_{lj} + \sum_{(t,s)\in A\setminus\{(j,l),(i,j)\}} Ne^z_{ts}$
- for $i \in I \setminus \{j, l\}$, $e^x_{il} + e^x_{jl} + \sum_{(t,s)\in A\setminus\{(j,l),(i,l)\}} Ne^z_{ts}$
- $e^x_{lj} + \sum_{(t,s)\in A\setminus\{(j,l),(l,j)\}} Ne^z_{ts}$
- $e^x_{jl} + \sum_{(t,s)\in A\setminus\{(j,l)\}} Ne^z_{ts}$.

□

4.3. Valid Inequalities Involving both Assignment and Traffic Variables

In this section, we present four families of valid inequalities that involve both assignment and traffic variables. The results and their proofs are given for the quadratic capacitated case as it is the most general one.

In the remaining of the paper, we present the results without removing the variables x_{jj}'s for ease of presentation. So we define $P_{QH} = conv(F_{QH})$.

We first present an extended formulation of the hub location problem (see Skorin-Kapov et al. [20]) and discuss how we can obtain valid inequalities from the projection of this formulation. We define X^k_{jl} to be 1 if the origin of commodity $k \in K$ is assigned to hub $j \in I$ and the destination is assigned to hub $l \in I$ and 0 otherwise. To obtain the new formulation, called *the hub location formulation*, we replace constraints (4) by

$$\sum_{l\in I} X^k_{jl} \geq x_{o(k)j} \qquad \forall j \in I, k \in K \tag{16}$$

$$-\sum_{j\in I} X^k_{jl} \geq -x_{d(k)l} \qquad \forall l \in I, k \in K \tag{17}$$

$$-\sum_{k\in K} t^k X^k_{jl} \geq -z_{jl} \qquad \forall (j, l) \in A \tag{18}$$

$$X^k_{jl} \geq 0 \qquad \forall (j, l) \in A, k \in K \tag{19}$$

in *QHL*. Now we can project out the variables X^k_{jl}'s using Farkas' Lemma.

Proposition 5. *Given* (x, z)*, there exists* X *that satisfies (16)–(19) if and only if*

$$\sum_{a\in A} z_a\beta_a \geq \sum_{k\in K} t^k \sum_{j\in I}(x_{o(k)j}\alpha_j^k - x_{d(k)j}\sigma_j^k) \tag{20}$$

for all $(\alpha, \sigma, \beta) \geq 0$ *such that*

$$\beta_{jl} \geq \alpha_j^k - \sigma_l^k \qquad \forall k \in K, (j,l) \in A \tag{21}$$

$$0 \geq \alpha_j^k - \sigma_j^k \qquad \forall k \in K, j \in I. \tag{22}$$

Any inequality of the form (20) defined by $(\alpha, \sigma, \beta) \geq 0$ which satisfies inequalities (21) and (22) is called a *projection inequality*.

Let C_p be the pointed cone of $(\alpha, \sigma, \beta) \geq 0$ that satisfy inequalities (21) and (22). For $(\alpha, \sigma, \beta) \neq (0, 0, 0)$ in C_p, define sets $B = \{(j,l) \in A : \beta_{jl} > 0\}$, $S_k = \{j \in I : \alpha_j^k > 0\}$ for all $k \in K$, $T_k = \{l \in I : \sigma_l^k > 0\}$ for all $k \in K$ and $K' = \{k \in K : S_k \neq \emptyset\}$.

The extreme rays of C_p lead to nondominated inequalities (20). Extreme rays such that $B = \emptyset$ or $K' = \emptyset$ yield trivial nondominated inequalities which are implied by the constraints of the formulation, such as $z_{jl} \geq 0$ or $x_{d(k)j} \geq 0$. We have a sufficient condition for a special class of the remaining rays to be extreme.

Proposition 6. *A ray* $(\alpha, \sigma, \beta) \in C_p$ *of the form* $\alpha_j^k = 1$ *if* $j \in S_k$, $\sigma_l^k = 1$ *if* $l \in T_k$, $\beta_{jl} = 1$ *if there exists a* $k \in K$ *such that* $j \in S_k$ *and* $l \notin T_k$ *and other entries are* 0 *is an extreme ray of* C_p *if the graph* $G' = (B, E)$ *where* $B = \{(j,l) \in A : \beta_{jl} = 1\}$ *and* $E = \{\{(j,l),(m,n)\} : (j,l) \in B, (m,n) \in B,$ *and there exists a* $k \in K$ *such that* $\beta_{jl} = \alpha_j^k$ *and* $\beta_{mn} = \alpha_m^k\}$ *is connected, the bipartite graphs* $G_k' = (S_k \times T_k, E_k')$ *where* $E_k' = \{\{j,l\} : j \in S_k, l \in T_k, \beta_{jl} = 0$ *or* $j = l\}$ *are connected for all* $k \in K$ *and* $S_k \neq I$ *for all* $k \in K$.

Proof. Assume that the conditions are satisfied and that there exist distinct $(\alpha, \sigma, \beta)^1$ and $(\alpha, \sigma, \beta)^2$ in C_p such that $(\alpha, \sigma, \beta) = 1/2(\alpha, \sigma, \beta)^1 + 1/2(\alpha, \sigma, \beta)^2$.

For $k \in K$, we have $(\alpha_j^k)^1 = (\alpha_j^k)^2 = 0$ if $j \notin S_k$, $(\sigma_j^k)^1 = (\sigma_j^k)^2 = 0$ if $j \notin T_k$ and $\beta_{jl}^1 = \beta_{jl}^2 = 0$ if $(j,l) \notin B$.

If $j \in S_k$ and $l \in T_k$ and $\beta_{jl} = 0$, we have $(\sigma_l^k)^1 \geq (\alpha_j^k)^1$ and $(\sigma_l^k)^2 \geq (\alpha_j^k)^2$. As $(\sigma_l^k)^1 + (\sigma_l^k)^2 = 2$ and $(\alpha_j^k)^1 + (\alpha_j^k)^1 = 2$, we have that $(\sigma_l^k)^1 = (\alpha_j^k)^1$ and $(\sigma_l^k)^2 = (\alpha_j^k)^2$. As the graph G_k' is connected, we can conclude that $(\alpha_j^k)^i = \gamma_k^i$ for all $j \in S_k$ and $(\sigma_j^k)^i = \gamma_k^i$ for all $j \in T_k$ for $i = 1, 2$.

Let $(j,l) \in B$ and $k \in K$ such that $j \in S_k$ and $l \notin T_k$. Then $\beta_{jl}^1 \geq \gamma_k^1$ and $\beta_{jl}^2 \geq \gamma_k^2$. As $\beta_{jl}^1 + \beta_{jl}^2 = 2$ and $\gamma_k^1 + \gamma_k^2 = 2$, we have $\beta_{jl}^1 = \gamma_k^1$ and $\beta_{jl}^2 = \gamma_k^2$ for all $k \in K$ such that $j \in S_k$ and $l \notin T_k$. As the graph G' is connected and as $S_k \neq I$ for all $k \in K$, $\gamma_k^i = \gamma^i$ for all $k \in K$ and $i = 1, 2$. So both $(\alpha, \sigma, \beta)^1$ and $(\alpha, \sigma, \beta)^2$ are multiples of (α, σ, β). □

Corollary 1. *The inequality*

$$\sum_{(j,l)\in A: j\in S, l\in T} z_{jl} \geq \sum_{k\in K'} t^k \Big(\sum_{j\in S} x_{o(k)j} + \sum_{l\in T} x_{d(k)l} - 1\Big) \tag{23}$$

where S and T are nonempty disjoint subsets of I and $K' \subseteq K$ is a valid inequality for P_{QH} and it is not dominated by other projection inequalities.

Proof. Consider (α, β, σ) such that $\alpha_j^k = 1$ if $j \in S$, $\sigma_l^k = 1$ if $l \in I \setminus T$ for $k \in K'$ and $\beta_{jl} = 1$ if $j \in S$ and $l \in T$ and other entries are 0. □

Remark that constraints (4) are special cases of projection inequalities (23) where $S = \{j\}$ and $T = \{l\}$.

Some lower bounds on the traffic which flows on the backbone links can also be computed using the capacity restrictions.

Theorem 2. *For $j \in I$, the traffic bound inequalities*

$$\sum_{l\in I\setminus\{j\}} z_{jl} \geq \sum_{i\in I\setminus\{j\}} T_{ij}^o x_{ij} + T_j^o x_{jj} \tag{24}$$

$$\sum_{l\in I\setminus\{j\}} z_{lj} \geq \sum_{i\in I\setminus\{j\}} T_{ij}^d x_{ij} + T_j^d x_{jj} \tag{25}$$

are valid for P_{QH} where

$$\begin{aligned} T_{ij}^o = \min & \sum_{m\in I\setminus\{i,j\}} t_{im}(1-u_m) \\ s.t. & \sum_{m\in I\setminus\{i,j\}} (a_m - T_{im} - T_{jm})u_m + \sum_{m\in I\setminus\{i,j\}} \sum_{l\in I\setminus\{i,j\}} T_{ml} u_m (1-u_l) \\ & \leq M - a_i - a_j - \sum_{m\in I\setminus\{i,j\}} (T_{im} + T_{jm}) \qquad (26) \\ & u_m \in \{0,1\} \quad \forall m \in I\setminus\{i,j\} \qquad (27) \end{aligned}$$

and

$$\begin{aligned} T_{ij}^d = \min & \sum_{m\in I\setminus\{i,j\}} t_{mi}(1-u_m) \\ & \text{s.t. (26) and (27)} \end{aligned}$$

for $i \in I \setminus \{j\}$,

$$\begin{aligned} T_j^o = \min & \sum_{m\in I\setminus\{j\}} t_{jm}(1-u_m) \\ s.t. & \sum_{m\in I\setminus\{j\}} (a_m - T_{jm})u_m + \sum_{m\in I\setminus\{j\}} \sum_{l\in I\setminus\{j\}} T_{ml} u_m (1-u_l) \\ & \leq M - a_j - \sum_{m\in I\setminus\{j\}} T_{jm} \qquad (28) \\ & u_m \in \{0,1\} \quad \forall m \in I\setminus\{j\} \qquad (29) \end{aligned}$$

and

$$T_j^d = \min \sum_{m \in I \setminus \{j\}} t_{mj}(1 - u_m)$$
$$\text{s.t. (28) and (29).}$$

Proof. Assume that node $i \in I \setminus \{j\}$ is assigned to node $j \in I$. Then the traffic of commodities from node i to nodes not assigned to node j travels on the backbone arcs going out of node j. The value T_{ij}^o is a lower bound on this traffic. □

We now present another family of valid inequalities which give lower bounds on the amount of traffic on the backbone arcs.

Theorem 3. *For $S \subset I$, $T \subset I$ such that $S \cap T = \emptyset$, $I_S \subseteq I$ and $I_i \subseteq I$ for all $i \in I_S$, the strengthened projection inequality*

$$\sum_{j \in S} \sum_{l \in T} z_{jl} \geq \sum_{i \in I_S} [\sum_{m \in I_i} t_{im} \sum_{l \in T} x_{ml} - \tau_i (1 - \sum_{j \in S} x_{ij})] \tag{30}$$

is valid for P_{QH} where

$$\begin{aligned}
\tau_i = \max & \sum_{m \in I_i} t_{im} \sum_{l \in T} u_{ml} \\
s.t. & \sum_{l \in T} u_{ml} \leq 1 && \forall m \in I_i \\
& \sum_{m \in I_i} (a_m + \sum_{s \in I \setminus I_i} T_{ms}) u_{ml} + \sum_{m \in I_i} \sum_{s \in I_i} T_{ms} u_{ml} (1 - u_{sl}) \leq M && \forall l \in T \\
& u_{ml} \in \{0, 1\} && \forall m \in I_i, l \in T
\end{aligned}$$

for all $i \in I_S$.

Proof. If $\sum_{j \in S} x_{ij} = 1$, for all $i \in I_S$, then inequality (30) means that the flow on the S-T cut must be larger than or equal to the sum of traffic between nodes in I_S and nodes assigned to concentrators in T. Hence, inequality (30) is obtained by down-lifting of the terms $(1 - \sum_{j \in S} x_{ij})$ for $i \in I_S$. Specifically, for a node $i \in I$, consider the following structure:

$$\sum_{j \in S \cup T} x_{ij} \leq 1 \tag{31}$$

$$\sum_{l \in T} x_{ml} \leq 1 \qquad \forall m \in I \tag{32}$$

$$\phi_{i,(S,T)} = \sum_{m \in I} t_{im} \sum_{j \in S} x_{ij} \sum_{l \in T} x_{ml} \tag{33}$$

$$\sum_{m \in I} a_m x_{ml} + \sum_{m \in I} \sum_{s \in I} T_{ms} x_{ml} (1 - x_{sl}) \leq M \qquad \forall l \in T \tag{34}$$

$$x_{ij} \in \{0, 1\} \qquad \forall j \in S \cup T \tag{35}$$

$$x_{ml} \in \{0, 1\} \qquad \forall m \in I, l \in T \tag{36}$$

where $\phi_{i,(S,T)}$ represents the amount of traffic that originates at node i and that travels on arcs from set S to set T given the assignment vector x. Let $F_{i,(S,T)}$ be the set of points $(\phi_{i,(S,T)}, x)$ that satisfy constraints (31)–(36) and $P_{i,(S,T)} = conv(F_{i,(S,T)})$. If inequality $\phi_{i,(S,T)} \geq \sum_{j\in S} \pi_{ij}x_{ij} + \sum_{m\in I}\sum_{l\in T} \pi^i_{ml}x_{ml} + \pi^i_0$ is valid for $P_{i,(S,T)}$ for each $i \in I_S$, then inequality $\sum_{j\in S}\sum_{l\in T} z_{jl} \geq \sum_{i\in I_S}(\sum_{j\in S} \pi_{ij}x_{ij} + \sum_{m\in I}\sum_{l\in T} \pi^i_{ml}x_{ml} + \pi^i_0)$ is valid for P_{QH} since $\sum_{j\in S}\sum_{l\in T} z_{jl} \geq \sum_{i\in I} \phi_{i,(S,T)}$ and $\phi_{i,(S,T)} \geq 0$ for $i \in I \setminus I_S$.

If node $i \in I_S$ is assigned to some node j in S, then $\phi_{i,(S,T)} = \sum_{m\in I} t_{im}\sum_{l\in T}$ $x_{ml} \geq \sum_{m\in I_i} t_{im}\sum_{l\in T} x_{ml}$. Otherwise, $\sum_{m\in I_i} t_{im}\sum_{l\in T} x_{ml} - \tau_i \leq 0 = \phi_{i,(S,T)}$. So $\phi_{i,(S,T)} \geq \sum_{m\in I_i} t_{im}\sum_{l\in T} x_{ml} - \tau_i(1 - \sum_{j\in S} x_{ij})$ is valid for $P_{i,(S,T)}$. □

Below we compare these inequalities with projection inequalities (23).

Proposition 7. *For any disjoint subsets S and T of the node set I and $K' \subseteq K$, inequality (23) is dominated by strengthened projection inequality (30) where $I_S = I$ and $I_i = \{m \in I : (i, m) \in K'\}$ for $i \in I$.*

Proof. Inequality (23) is

$$\begin{aligned}\sum_{j\in S}\sum_{l\in T} z_{jl} &\geq \sum_{k\in K'} t^k\Big(\sum_{j\in S} x_{o(k)j} + \sum_{l\in T} x_{d(k)l} - 1\Big)\\ &= \sum_{i\in I}\sum_{m\in I_i} t_{im}\Big(\sum_{j\in S} x_{ij} + \sum_{l\in T} x_{ml} - 1\Big)\\ &= \sum_{i\in I}\Big[\sum_{m\in I_i}\sum_{l\in T} t_{im}x_{ml} - \sum_{m\in I_i} t_{im}\Big(1 - \sum_{j\in S} x_{ij}\Big)\Big].\end{aligned}$$

Clearly $\tau_i \leq \sum_{m\in I_i} t_{im}$ for all $i \in I$. □

As the set T becomes large, the gain over the projection inequalities decreases. If all nodes in I_i can be assigned to the nodes in T, then $\tau_i = \sum_{m\in I_i} t_{im}$. If this is the case for all $i \in I_S$ then inequalities (23) and (30) are equivalent.

In the sequel, we consider inequalities (30) where $S = \{j\}$ and $T = \{l\}$. We use $\phi_{i,(j,l)}$ instead of $\phi_{i,(\{j\},\{l\})}$.

The value of τ_i can be improved if we fix the values of some of the variables. Let I^0 and I^1 be two disjoint subsets of I, define $I_i = I\setminus(I^0 \cup I^1)$ and compute

$$\begin{aligned}\tau_i(I^0, I^1) = \max & \sum_{m\in I_i} t_{im}u_m\\ \text{s.t.} & \sum_{m\in I_i}(a_m + \sum_{s\in I^0} T_{ms} - \sum_{s\in I^1} T_{ms})u_m\\ & + \sum_{m\in I_i}\sum_{s\in I_i} T_{ms}u_m(1 - u_s) \leq M - \sum_{m\in I^1}(a_m + \sum_{s\in I\setminus I^1} T_{ms})\\ & u_m \in \{0, 1\} \qquad \forall m \in I_i.\end{aligned}$$

Then

$$\phi_{i,(j,l)} \geq \sum_{m\in I^1} t_{im}x_{ij} + \sum_{m\in I_i} t_{im}x_{ml} - \tau_i(I^0, I^1)(1 - x_{ij})$$

is valid for $P_{i,(j,l)}$ when $x_{ml} = 1$ for all $m \in I^1$ and $x_{ml} = 0$ for all $m \in I^0$. Since the values of some of the variables are fixed, $\tau_i(I^0, I^1) \leq \tau_i$. By lifting the variables whose values are fixed to 0 or 1, we can obtain a valid inequality of the form

$$\phi_{i,(j,l)} \geq \sum_{m \in I^1} t_{im} x_{ij} + \sum_{m \in I_i} t_{im} x_{ml} - \tau_i(I^0, I^1)(1 - x_{ij}) + \sum_{m \in I^0} \alpha_p x_{ml} - \sum_{m \in I^1} \alpha_p (1 - x_{ml}). \tag{37}$$

The following theorem tells us how to compute the lifting coefficients.

Theorem 4. *Inequality (37) is valid for $P_{i,(j,l)}$ where $\alpha_p = min\{\alpha_p^0, t_{ip}\}$ for all $p \in I^0$ where*

$$\begin{aligned} \alpha_p^0 = \tau_i(I^0, I^1) - \max & \sum_{m \in I_i} t_{im} u_m + \sum_{m \in Y_p} \alpha_m u_m \\ s.t. & \sum_{m \in I_i \cup Y_p} (a_m + \sum_{s \in I_0 \setminus (Y_p \cup \{p\})} T_{ms} - \sum_{s \in I^1 \cup \{p\}} T_{ms}) u_m \\ & + \sum_{m \in I_i \cup Y_p} \sum_{s \in I_i \cup Y_p} T_{ms} u_m (1 - u_s) \\ & \leq M - \sum_{m \in I^1 \cup \{p\}} (a_m + \sum_{s \in I \setminus (I^1 \cup \{p\})} T_{ms}) \\ & u_m \in \{0, 1\} \qquad \forall m \in I_i \cup Y_p \end{aligned}$$

and $Y_p \subset I^0$ is the set of indices of the variables that have been lifted before x_{pl} and $\alpha_p = max\{\alpha_p^1, t_{ip}\}$ for all $p \in I^1$ where

$$\begin{aligned} \alpha_p^1 = \max & \sum_{m \in I_i} t_{im} u_m + \sum_{m \in I^0} \alpha_m u_m + \sum_{m \in Y_p^1} \alpha_m u_m - (\tau_i(I^0, I^1) + \sum_{m \in Y_p^1} \alpha_m) \\ s.t. & \sum_{m \in I_i \cup I^0 \cup Y_p^1} (a_m - \sum_{s \in I^1 \setminus (Y_p^1 \cup \{p\})} T_{ms} + T_{mp}) u_m \\ & + \sum_{m \in I_i \cup I^0 \cup Y_p^1} \sum_{s \in I_i \cup I^0 \cup Y_p^1} T_{ms} u_m (1 - u_s) \\ & \leq M - \sum_{m \in I^1 \setminus (Y_p^1 \cup \{p\})} (a_m + \sum_{s \in I_i \cup I^0 \cup Y_p^1 \cup \{p\}} T_{ms}) \\ & u_m \in \{0, 1\} \qquad \forall m \in I_i \cup I^0 \cup Y_p^1 \end{aligned}$$

and $Y_p^1 \subset I^1$ is the set of indices of the variables that have been lifted before x_{pl}.

Proof. Let $P_{i,(j,l)}(I^0, I^1) = conv(F_{i,(j,l)} \cap \{x : x_{ml} = 0 \ \ \forall m \in I^0, x_{ml} = 1 \ \ \forall m \in I^1\})$. We first lift the variables with indices in I^0 and then the variables with indices in I^1. When lifting x_{pl} with $p \in I^0$, we have an inequality of the form

$$\phi_{i,(j,l)} \geq \sum_{m \in I^1} t_{im} x_{ij} + \sum_{m \in I_i} t_{im} x_{ml} - \tau_i(I^0, I^1)(1 - x_{ij}) + \sum_{m \in Y_p} \alpha_m x_{ml}$$

which is valid for $P_{i,(j,l)}(I^0 \setminus Y_p, I^1)$ and we would like to find an α_p such that the inequality

$$\phi_{i,(j,l)} \geq \sum_{m \in I^1} t_{im} x_{ij} + \sum_{m \in I_i} t_{im} x_{ml} - \tau_i(I^0, I^1)(1 - x_{ij}) + \sum_{m \in Y_p} \alpha_m x_{ml} + \alpha_p x_{pl} \tag{38}$$

is valid for $P_{i,(j,l)}(I^0 \setminus (Y_p \cup \{p\}), I^1)$. For inequality (38) to be valid, we should have:

$$\alpha_p \leq \phi_{i,(j,l)} - \sum_{m \in I^1} t_{im} x_{ij} - \sum_{m \in I_i} t_{im} x_{ml} + \tau_i(I^0, I^1)(1 - x_{ij}) - \sum_{m \in Y_p} \alpha_m x_{ml} \tag{39}$$

for all $x \in P_{i,(j,l)}(I^0 \setminus (Y_p \cup \{p\}), I^1)$. We have two cases to consider:

Case 1: $x_{ij} = 0$. Then $\phi_{i,(j,l)} = 0$ and inequality (39) becomes:

$$\alpha_p \leq \tau_i(I^0, I^1) - \sum_{m \in I_i} t_{im} x_{ml} - \sum_{m \in Y_p} \alpha_m x_{ml}.$$

The value α_p^0 gives the minimum of the right hand side.

Case 2: $x_{ij} = 1$. Then $\phi_{i,(j,l)} = \sum_{m \in I^1} t_{im} + \sum_{m \in I_i \cup Y_p} t_{im} x_{ml} + t_{ip}$. So we have

$$\begin{aligned} \alpha_p &\leq \sum_{m \in I^1} t_{im} + \sum_{m \in I_i \cup Y_p} t_{im} x_{ml} + t_{ip} - \sum_{m \in I^1} t_{im} - \sum_{m \in I_i} t_{im} x_{ml} - \sum_{m \in Y_p} \alpha_m x_{ml} \\ &= \sum_{m \in Y_p} (t_{im} - \alpha_m) x_{ml} + t_{ip}. \end{aligned}$$

It is easy to show by induction that $\alpha_m \leq t_{im}$ for all $m \in Y_p$.

To have the strongest inequality we choose $\alpha_p = min\{\alpha_p^0, t_{ip}\}$.

The lifting for $p \in I^1$ can be done in a similar way. □

A last family of valid inequalities can be obtained by computing lower bounds on the traffic on arc (j, l) using the total number of terminals assigned to hubs j and l.

Theorem 5. *Let $I' \subseteq I$ and $T^\mu(I')$ denote a lower bound on the traffic on arc $(j, l) \in A$ when μ terminals of set I' are assigned to j and l for $\mu = 0, .., |I'|$. Define $dT^{\mu+1}(I') = T^{\mu+1}(I') - T^\mu(I')$ for all $\mu = 0, .., |I'| - 1$. If $dT^{\mu+1}(I') \geq dT^\mu(I')$ for all $\mu = 0, .., |I'| - 1$, then the step inequality*

$$z_{jl} \geq T^\mu(I') + dT^{\mu+1}(I')[\sum_{i \in I'}(x_{ij} + x_{il}) - \mu] \tag{40}$$

is valid inequality for P_{QH} for any $\mu = 0, .., |I'| - 1$.

Proof. Let $\lambda = \sum_{i \in I'}(x_{ij} + x_{il})$. Then by definition $z_{jl} \geq T^{\lambda}(I')$. If $\mu \leq \lambda$ then $T^{\lambda}(I') = T^{\mu}(I') + \sum_{r=\mu}^{\lambda-1} dT^{r+1}(I')$. Since $dT^{r+1}(I') \geq dT^{\mu+1}(I')$ for all $r \geq \mu$, we have $\sum_{r=\mu}^{\lambda-1} dT^{r+1}(I') \geq (\lambda - \mu) dT^{\mu+1}(I')$. So

$$z_{jl} \geq T^{\lambda}(I') = T^{\mu}(I') + \sum_{r=\mu}^{\lambda-1} dT^{r+1}(I') \geq T^{\mu}(I') + (\lambda - \mu) dT^{\mu+1}(I').$$

If $\mu > \lambda$, then we can show that $\sum_{r=\lambda}^{\mu-1} dT^{r+1}(I') \leq (\mu - \lambda) dT^{\mu+1}(I')$. Hence

$$z_{jl} \geq T^{\lambda}(I') = T^{\mu}(I') - \sum_{r=\lambda}^{\mu-1} dT^{r+1}(I') \geq T^{\mu}(I') + (\lambda - \mu) dT^{\mu+1}(I').$$

This proves that inequality (40) is valid for P_{QH} when $x_{ij} + x_{il} = 0$ for all $i \in I \setminus I'$. Further, it is clear that assigning more terminal nodes to nodes j and l cannot decrease the traffic on arc (j, l). □

Now, we present a simple way to compute lower bounds $T^{\mu}(I')$'s. Let $g_m(I')$ be the maximum number of terminals in I' that can be assigned to a given hub m. As a hub node is assigned to itself, $g_j(I')$ and $g_l(I')$ can be different. Let $g(I') = \max\{g_j(I'), g_l(I')\}$. If μ is larger than $g(I')$, then at least $g(I')(\mu - g(I'))$ commodities travel from j to l. Let $t^{(k)}(I')$ denote the kth smallest traffic with origin and destination in I'. We define the lower bounds $T^{\mu}(I')$'s as follows:

$$T^{\mu}(I') = \begin{cases} \sum_{k=1}^{g(I')(\mu - g(I'))} t^{(k)}(I') & \text{if } \mu > g(I') \\ 0 & \text{otherwise.} \end{cases}$$

These bounds $T^{\mu}(I')$'s satisfy $dT^{\mu+1}(I') \geq dT^{\mu}(I')$ for all $\mu = 0, .., |I'| - 1$. So they lead to valid inequalities.

5. A branch and cut algorithm for QHL

In this section, we discuss the branch and cut algorithm to solve QHL. We first compare several formulations to choose the one to use in the algorithm. We strengthen this formulation using projection inequalities. The remaining part of the section is devoted to the discussion of different ingredients of the branch and cut algorithm. We first present a preprocessing and strengthening algorithm. Then we discuss the separation algorithms implemented for each family of cuts. We make some initial tests to decide about the parameters of the branch and cut algorithm. Then we present computational results using some data supplied by France Telecom and the AP data from the OR Library (see Beasley [6]).

5.1. Comparing Formulations

We first present the formulation we use in our branch and cut algorithm. This formulation, called *QHL1*, can be obtained by replacing the quadratic capacity constraints (5) in *QHL* by the linear constraints (8).

Formulation *QHL1* has $O(n^2)$ variables and exponentially many constraints. When we start the branch and cut algorithm, we do not include constraints (4) in the linear programming (LP) relaxation. We add these inequalities whenever we find them to be violated by the optimal solution of the current LP relaxation. For a given arc $(j,l) \in A$ and a solution (x^*, z^*), let $K_{jl} = \{k \in K : x^*_{o(k)j} + x^*_{d(k)l} - 1 > 0\}$. If $z^*_{jl} - \sum_{k \in K_{jl}} t^k (x^*_{o(k)j} + x^*_{d(k)l} - 1) < 0$, the set K_{jl} yields the most violated inequality.

Recall the variables X^k_{jl}'s defined in Section 4.3. Constraints (4) in formulation *QHL1* are obtained by projecting out the variables X^k_{jl}'s from the system:

$$
\begin{aligned}
& X^k_{jl} \geq x_{o(k)j} + x_{d(k)l} - 1 && \forall (j,l) \in A, k \in K \\
& z_{jl} \geq \sum_{k \in K} t^k X^k_{jl} && \forall (j,l) \in A \\
& X^k_{jl} \geq 0 && \forall (j,l) \in A, k \in K
\end{aligned}
$$

which corresponds to the standard linearization of $z_{jl} = \sum_{k \in K} t^k x_{o(k)j} x_{d(k)l}$ (see Dantzig [8]). Inequalities (4) are the only non redundant projection inequalities of this system.

The second formulation we consider is the hub location formulation given by (1)–(3), (6)–(7), (8) and (16)–(19). This formulation, denoted by *QHL2*, has $O(n^4)$ variables and $O(n^3)$ constraints.

We can also obtain a multicommodity flow formulation *QHL2m* using the variables X^k_{jl}'s by replacing (16)–(19) by

$$
\begin{aligned}
& \sum_{l \in I \setminus \{j\}} X^k_{jl} - \sum_{l \in I \setminus \{j\}} X^k_{lj} = x_{o(k)j} - x_{d(k)j} && \forall j \in I, k \in K \\
& z_{jl} \geq \sum_{k \in K} t^k X^k_{jl} && \forall (j,l) \in A \\
& X^k_{jl} \geq 0 && \forall (j,l) \in A, k \in K.
\end{aligned}
$$

This formulation is valid since the costs R_{jl}'s satisfy the triangle inequality. This also implies that the LP relaxations of *QHL2* and *QHL2m* have the same value.

Ernst and Krishnamoorthy [10] suggest a formulation with a smaller number of variables. They aggregate the commodities having the same origin to decrease the number of flow variables. Define f^i_{jl} to be the flow of commodities originating at node $i \in I$ and traveling along arc $(j,l) \in A$. Formulation *QHL3* can be obtained by replacing constraints (4) in *QHL1* by the following set of constraints:

$$\sum_{l\in I\setminus\{j\}} f^i_{jl} - \sum_{l\in I\setminus\{j\}} f^i_{lj} = \sum_{m\in I} t_{im}(x_{ij} - x_{mj}) \qquad \forall i, j \in I \tag{41}$$

$$z_{jl} \geq \sum_{i\in I} f^i_{jl} \qquad \forall (j,l) \in A$$

$$f^i_{jl} \geq 0 \qquad \forall i \in I, (j,l) \in A.$$

Formulation *QHL3* has $O(n^3)$ variables and $O(n^2)$ constraints. Obviously the LP relaxation of *QHL2* has a larger value than the one of *QHL1* and the one of *QHL3*. However, the comparison between *QHL1* and *QHL3* is not so clear as shown by the following example.

Example 1. Suppose we have five nodes and two commodities with one unit of traffic from node 1 to node 2 and from node 1 to node 3. Consider the vector x with nonzero coordinates given by $x_{44} = 1, x_{55} = 1, x_{14} = 0.7, x_{15} = 0.3, x_{24} = 0.5, x_{25} = 0.5, x_{34} = 0.9, x_{35} = 0.1$ (see Figure 2).

As $x_{11} = x_{22} = x_{33} = 0$, there is no flow on the arcs adjacent to nodes 1, 2 and 3. Since the right hand sides of the flow balance constraints (41) for nodes 4 and 5 are equal to zero, the vector (x, z, f) where $f = z = 0$ is feasible for the LP relaxation of *QHL3*.

However (x, z) is not feasible for the LP relaxation of *QHL1* since constraint (4) for arc (4, 5) and $K' = \{(1, 2)\}$ yields $z_{45} \geq 0.7 + 0.5 - 1 = 0.2$.

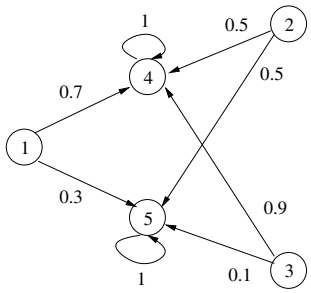

Fig. 2. Assignment with zero traffic in the LP relaxation of QHL3

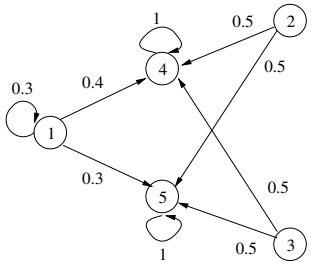

Fig. 3. Assignment with zero traffic in the LP relaxation of QHL1

Now consider the vector x with nonzero coordinates given by $x_{11} = 0.3, x_{44} = 1, x_{55} = 1, x_{14} = 0.4, x_{15} = 0.3, x_{24} = 0.5, x_{25} = 0.5, x_{34} = 0.5, x_{35} = 0.5$ (see Figure 3).

As $x_{1j} + x_{2l} \leq 1$ and $x_{1j} + x_{3l} \leq 1$ for all $(j, l) \in A$, (x, z) where $z = 0$ is feasible for the LP relaxation of *QHL1*. However the flow balance constraint for node 1 is

$$f^1_{12} + f^1_{13} + f^1_{14} + f^1_{15} - f^1_{21} - f^1_{31} - f^1_{41} - f^1_{51} = 2 \times 0.3 = 0.6.$$

So any (x, z) with $z = 0$ is infeasible for the LP relaxation of *QHL3*. □

We now present the results obtained from these four formulations for a small sample of test problems. We have seven problems with 10 nodes, six problems with 12 nodes and five problems with 15 nodes generated from the same cost and traffic data. We keep $M = 10$ and multiply the traffic values by $Q \in \{1, 2/3, 1/2, 1/3, 1/4, 1/5, 1/6\}$ to have different types of problems with respect to the tightness of capacities and the effect of the routing cost. We removed problems with $Q = 1$ for $n = 12$ and $n = 15$ and the problem with $Q = 2/3$ for $n = 15$ as they were infeasible. This set of problems will be used in the remainder of this section to decide about different components of the branch and cut algorithm.

To compare the formulations presented above, we use a branch and cut algorithm which will be described in detail in Section 5.5. We should note that for formulation *QHL1*, the only valid inequalities separated are constraints (4). For the other formulations, we have a pure branch and bound algorithm.

The results are given in Table 2. The columns *gap*, *CPU* and *first* give the duality gap at the root node (i.e., $gap = \frac{opt-db}{opt} * 100$ where opt is the optimal value and db is the dual bound before branching), the CPU time in seconds to solve the problem and the CPU time to solve the first LP relaxation, respectively. We set a time limit of one hour. If the problem is not solved to optimality within one hour of CPU time, then we write *time* in the column CPU. If we run out of memory, then we write *memory*. For formulation *QHL1*, we do not report the CPU time to solve the first LP as it was less than 0.01 seconds for all problems. We do not report it for formulation *QHL2* either, because it was always longer than the one for *QHL2m*. Finally, the gap for *QHL2m* is also omitted since it is equal to that of *QHL2*.

We observe that the duality gap decreases as Q decreases independently of which formulation we choose. This is because QHL approaches the uncapacitated concentrator location problem as Q tends to 0.

Formulations *QHL2* and *QHL2m* do not seem promising to solve big problems as the time to solve the LP relaxations grows very fast. However, the other two formulations do not perform much better as they have larger duality gaps and it takes a lot of time to close this gap. It is clear that if we would like to use one of these two formulations, we should strengthen it.

5.2. Projection Inequalities

Given a current solution (x^*, z^*), to check whether there exists a projection inequality (20) violated amounts to solving a linear programming problem with $O(n^3)$ variables and $O(n^4)$ constraints. Since the size of this LP is about the same as that of the LP

Table 2. Comparison of formulations

		QHL1		QHL2		QHL2m		QHL3		
n	Q	gap	CPU	gap	CPU	CPU	first	gap	CPU	first
10	1	72.22	382.56	31.40	time	time	3.91	43.33	455.82	0.12
	2/3	59.86	1025.53	24.23	time	time	2.92	31.24	memory	0.15
	1/2	47.93	129.13	10.43	841.24	449.97	2.71	16.08	139.49	0.15
	1/3	37.71	27.80	3.38	75.98	40.62	2.23	8.36	11.85	0.12
	1/4	33.01	23.26	2.71	71.36	34.00	1.87	7.53	8.34	0.11
	1/5	28.43	15.66	1.65	38.22	18.03	1.39	5.60	5.02	0.14
	1/6	24.85	10.30	1.14	44.08	20.77	1.4	4.57	5.3	0.13
12	2/3	63.77	memory	23.14	time	time	20.76	31.88	memory	0.7
	1/2	52.16	2313.95	9.43	time	time	18.40	17.22	memory	0.6
	1/3	41.93	793.41	3.99	609.53	329.40	13.55	9.78	367.78	0.7
	1/4	35.85	458.10	2.55	229.69	123.51	11.96	7.55	101.84	0.61
	1/5	31.54	234.08	2.59	154.70	84.6	9.27	6.27	47.53	0.63
	1/6	27.87	112.88	1.98	261.10	137.08	9.03	5.17	31.48	0.46
15	1/2	64.25	time	22.39	time	time	185.50	32.44	memory	3.74
	1/3	51.04	time	9.91	time	time	170.09	18.35	memory	4.85
	1/4	41.48	time	3.61	time	2441.83	143.37	10.41	memory	3.58
	1/5	36.69	time	2.94	2762.05	1330.32	145.00	8.08	2889.87	3.68
	1/6	31.66	time	1.13	768.79	428.04	101.87	4.95	141.91	3.57

relaxation of *QHL2*, we prefer to use heuristics to separate some specific families of projection inequalities. We focus on the projection inequalities (23) for which the separation is easy if we are given the sets S and T. We use Algorithm 1 to separate the projection inequalities.

Algorithm 1 Separate Projection

for all $i \in I$ such that $x^*_{ii} > 0$ **do**
  add inequality (23) if it is violated for $S = \{i\}$, $T = I \backslash \{i\}$ and $K' = \{k \in K : x^*_{o(k)i} - x^*_{d(k)i} > 0\}$
  add inequality (23) if it is violated for $S = I \backslash \{i\}$, $T = \{i\}$ and $K' = \{k \in K : x^*_{d(k)i} - x^*_{o(k)i} > 0\}$
end for
for all $k \in K$ **do**
  add inequality (23) if it is violated for $S = \{i \in I : x^*_{o(k)j} - x^*_{d(k)j} > 0\}$, $T = \{i \in I : x^*_{d(k)j} - x^*_{o(k)j} > 0\}$ and $K' = \{k' \in K : \sum_{j \in S} x^*_{o(k')j} + \sum_{l \in T} x^*_{d(k')l} - 1 > 0\}$
end for

In Table 3, we present the results obtained by using the projection inequalities with formulations *QHL1* and *QHL3*. Clearly, formulation *QHL1* with the projection inequalities outperforms the other formulation.

We can also observe that the problems with large Q values remain much harder to solve. The difficulty of the problem changes a lot with Q. If the number of nodes is not very big, then we can solve the problems with small Q values by an off-the-shelf MIP solver. Problems with bigger number of nodes or larger Q values request the development of a specially tailored branch and cut algorithm.

5.3. Preprocessing and Basic Strengthening

The preprocessing algorithm (Algorithm 2) is based on the capacity restrictions and first checks if the problem is feasible. If so, it also checks for any pair of nodes whether they

Table 3. Improvement due to projection inequalities

		QHL1+pro		QHL3+pro	
n	Q	gap	CPU	gap	CPU
10	1	32.86	376.61	32.80	2138.56
	2/3	24.36	709.90	24.53	3575.78
	1/2	10.86	37.66	10.77	179.39
	1/3	4.68	5.42	4.76	21.75
	1/4	4.68	3.24	4.47	9.40
	1/5	3.58	2.82	3.00	6.43
	1/6	2.56	1.58	2.16	5.76
12	2/3	25.25	time	24.74	time
	1/2	13.17	556.85	12.24	time
	1/3	7.71	38.47	6.47	208.73
	1/4	5.31	12.61	5.23	52.86
	1/5	4.78	10.91	4.14	31.03
	1/6	3.66	9.37	3.24	27.59
15	1/2	26.56	time	24.88	time
	1/3	15.52	time	14.30	time
	1/4	7.91	383.52	7.10	2118.01
	1/5	6.39	184.55	5.38	684.50
	1/6	4.58	29.83	2.85	92.17

can be assigned to the same hub. It removes the nodes that cannot be assigned with any other node. It also adds constraints imposing lower bounds on the traffic and a lower bound on the number of concentrators to be installed.

Algorithm 2 Preprocessing

for all $i \in I$ **do**
  if $a_i + \sum_{l \in I} T_{il} > M$ **then**
    the problem is infeasible
  else if $a_i + a_m + \sum_{l \in I \setminus \{i,m\}} (T_{il} + T_{ml}) > M$ for all $m \in I \setminus \{i\}$ **then**
    fix $x_{ii} = 1$ and $x_{mi} = 0$ for all $m \in I \setminus \{i\}$
    remove node i from the set I by doing the necessary changes in the cost and demand vectors
  else if $a_i + a_m + \sum_{l \in I \setminus \{i,m\}} (T_{il} + T_{ml}) \leq M$ for all $m \in I \setminus \{i\}$ **then**
    for all $j \in I \setminus \{i\}$ **do**
      add inequality $x_{ij} \leq x_{jj}$
    end for
  else
    for all $m \in I \setminus \{i\}$ such that $a_i + a_m + \sum_{l \in I \setminus \{i,m\}} (T_{il} + T_{ml}) > M$ **do**
      for all $j \in I$ **do**
        add inequality $x_{ij} + x_{mj} \leq x_{jj}$
      end for
    end for
  end if
  compute T_i^o and T_i^d as described in Theorem 2
  $d_i \leftarrow a_i + T_i^o + T_i^d$
end for
apply the algorithm of Martello and Toth [18] to compute the L_2 bound (a lower bound on the optimal value of a binpacking problem) to determine a lower bound L on the number of hubs to satisfy all demands d_i's and add inequality $\sum_{j \in I} x_{jj} \geq L$
for all $j \in I$ **do**
  add the traffic bound inequalities $\sum_{l \in I \setminus \{j\}} z_{jl} \geq \sum_{i \in I} T_i^o x_{ij}$ and $\sum_{l \in I \setminus \{j\}} z_{lj} \geq \sum_{i \in I} T_i^d x_{ij}$
end for

The traffic bound inequalities used in the processing are weaker than inequalities (24) and (25). But in this way, we need to solve n quadratic integer programming problems instead of n^2. Similar quadratic integer programming problems also arise in the separation and lifting of valid inequalities for QHL. We use the branch and bound algorithms given in Yaman [23] to solve these problems with small sizes.

It is possible to compute lower bounds for T_i^o's and T_i^d by solving a series of linear knapsack problems. Algorithm 3 computes a lower bound on $T_i^o + T_i^d$ for each $i \in I$ but it can be easily modified to compute lower bounds T_i^o's and T_i^d's separately.

Algorithm 3 Compute Traffic

for all $i \in I$ **do**
  $d_i \leftarrow a_i$
end for
$improved \leftarrow 1$
while $improved$ **do**
  $improved \leftarrow 0$
  for all $i \in I$ **do**
    compute $T_i = \min\{\sum_{m \in I \setminus \{i\}} T_{im}(1 - u_m) : \sum_{m \in I \setminus \{i\}} d_m u_m \leq M - d_i, u \in \{0, 1\}^{n-1}\}$.
    if $a_i + T_i > d_i$ **then**
      $d_i \leftarrow a_i + T_i$ and $improved \leftarrow 1$
    end if
  end for
end while

For problems with $n \geq 20$, we do preprocessing based on lower bounds for T_i^o's and T_i^d's computed using Algorithm 3 rather than their exact values.

In Table 4, we present the results obtained using the preprocessing and basic strengthening algorithm. We compute T_i^o's and T_i^d's by solving quadratic problems. The first and second columns give the results without preprocessing and with preprocessing respectively. The last column gives the percentage improvement in the duality gap and in the CPU time due to preprocessing and strengthening. The sign $\star$ indicates that the improvement in the CPU time cannot be computed as the problem is not solved to optimality in one hour without preprocessing.

Preprocessing and strengthening are quite effective in all kinds of problems, but they are more effective on problems with tight capacities. As long as the computation time is concerned, the preprocessing performs well and leads to an improvement for all problems. The least improvement is around 21%.

The demands d_i's computed during preprocessing can be used to obtain a relaxation of QHL which is a LHL. As $\sum_{i \in I} a_i > M$, we can expect $d_i > a_i$ so that such a relaxation is stronger than a relaxation obtained using demands a_i's. This relaxation will be used often to avoid solving quadratic problems which arise in the separation and lifting of valid inequalities of QHL.

5.4. Other Valid Inequalities and Separation Algorithms

This section discusses the valid inequalities and their separation algorithms used in the branch and cut algorithm. The separation routine is run at every node of the branch and cut tree. Let (x^*, z^*) denote the current solution.

Table 4. Improvement due to preprocessing and basic strengthening

		without prep.		with prep.		% improvement in	
n	Q	gap	CPU	gap	CPU	gap	CPU
10	1	32.86	376.61	0.00	0.07	100.00	99.98
	2/3	24.36	709.90	3.51	1.83	85.59	99.74
	1/2	10.86	37.66	6.26	4.94	42.32	86.88
	1/3	4.68	5.42	4.55	2.42	2.82	55.35
	1/4	4.68	3.24	3.41	1.72	27.14	46.91
	1/5	3.58	2.82	2.26	0.79	36.89	71.99
	1/6	2.56	1.58	1.06	0.62	58.66	60.76
12	2/3	25.25	time	7.29	29.18	71.12	⋆
	1/2	13.17	556.85	7.92	31.58	39.84	94.33
	1/3	7.71	38.47	7.48	18.12	2.93	52.90
	1/4	5.31	12.61	4.74	9.26	10.79	26.57
	1/5	4.78	10.91	3.42	4.97	28.50	54.45
	1/6	3.66	9.37	2.51	3.58	31.29	61.79
15	1/2	26.56	time	7.02	145.42	73.56	⋆
	1/3	15.52	time	11.25	2003.95	27.53	⋆
	1/4	7.91	383.52	6.91	147.81	12.60	61.46
	1/5	6.39	184.55	5.94	122.76	7.02	33.48
	1/6	4.58	29.83	3.00	23.39	34.54	21.59

Lifted Quadratic Cover Inequalities (Yaman [23]) A subset $C \subseteq I$ such that $\sum_{i\in C} a_i + \sum_{i\in C}\sum_{m\in I\setminus C} T_{im} > M$ is called a quadratic cover. This notion is very close to that of cover for knapsack constraints (see Balas [5], Hammer et al. [15] and Wolsey [22]). If $C \subseteq I$ is a quadratic cover, then the quadratic cover inequality $\sum_{i\in C} x_{ij} \leq (|C|-1)x_{jj}$ is valid for P_{QH}.

To separate the quadratic cover inequalities, we use the branch and bound algorithm given in Yaman [23]. For a given $j \in I$, we define $I_1 = \{i \in I : x^*_{ij} = x^*_{jj}\}$ and $I_0 = \{i \in I : x^*_{ij} = 0\}$. We fix $x_{ij} = 1$ for all $i \in I_1$ and $x_{ij} = 0$ for all $i \in I_0$. If we find a cover C, we also fix $x_{ij} = 0$ for $i \in I \setminus (I_1 \cup I_0 \cup C)$. We lift first the variables whose values are fixed to 0 and who have $x^*_{ij} > 0$. If the resulting inequality is violated, then we lift the variables whose values are fixed to 1 except x_{jj}. After, we lift the remaining variables whose values are fixed to 0. The last variable we lift is x_{jj}. This is similar to the order given by Gu et al. [14].

We can also separate cover inequalities on linear knapsack constraints which give relaxations of the quadratic constraint. For $j \in I$ and $K' \subseteq K$, the linear knapsack inequality

$$\sum_{i\in I} a_i x_{ij} + \sum_{(i,m)\in K'} T_{im}(x_{ij} - x_{mj}) \leq M \tag{42}$$

is valid for P_{QH} (see Yaman [23]). So any cover inequality based on this knapsack inequality is also valid for P_{QH}.

For a given $j \in I$, the problem of finding the set K' which defines a knapsack inequality for which there is a violated cover inequality is the same problem as the separation of quadratic cover inequalities. So, we adopt the following heuristic method: We take $K' = \{(i,m) \in K : x^*_{ij} - x^*_{mj} > 0\}$. Then we separate and lift the cover inequality on the knapsack inequality (42) defined by K'.

Another possibility is to consider inequality $\sum_{i\in I} d_i x_{ij} \leq M$ where d_i's are computed during preprocessing. In general these inequalities are different from inequalities (42). However we are not able to compare their strengths. Still, as the inequalities (42) have demands that are based on the actual assignment vector and the associated flow in the backbone network, they are likely to be more useful for problems which do not have very tight capacities.

For problems with $n \geq 20$, we separate cover inequalities based on linear knapsack constraints rather than quadratic cover inequalities. For each node $j \in I$, we first look for a violated lifted cover inequality based on the linear knapsack (42). If there is none, then we try to find one for the knapsack with demands d_i's. We do it sequentially to avoid adding the same inequality twice. The lifting is as explained above.

Lifted Strengthened Projection Inequalities We separate the strengthened projection inequalities (30) where S and T are singletons. For a given arc (j, l), for each node $i \in I$, we take $I^1 = \{m \in I : x^*_{ml} = 1\}$, $I_i = \{m \in I \backslash I^1 : x^*_{ij} + x^*_{ml} > 1\}$, $I_i^+ = \{m \in I \backslash (I^1 \cup I_i) : x^*_{ml} > 0\}$ and $I_i^0 = I \backslash (I^1 \cup I_i \cup I_i^+)$. We fix $x_{ml} = 1$ for all $m \in I^1$ and $x_{ml} = 0$ for all $m \in I_i^+ \cup I_i^0$. We compute $\tau_i(I_i^+ \cup I_i^0, I^1)$ and then lift the variables. For $j \in I$, we use the linear knapsack constraint (42) defined by $K' = \{(i, m) \in K : x^*_{ij} - x^*_{mj} > 0\}$ as the capacity constraint to avoid solving quadratic problems. The lifting order is I_i^+, I^1 and I_i^0.

Step Inequalities We use the following heuristic to separate the step inequalities: For a given arc (j, l), we take $I' = \{i \in I : x^*_{ij} + x^*_{il} > 0\}$. We add the violated inequalities to the formulation.

The valid inequalities we discuss below are valid and facet defining under some conditions for the polytopes of concentrator location problems. For details, see Yaman [23]. Here we give the definition of each inequality and briefly discuss the separation algorithm.

Quadratic Binpacking Inequalities on Three Nodes (Yaman [23]) Let $\{i, j, l\} \subseteq I$ be such that $i \neq j \neq l$. If $\{i, j, l\}$ is a quadratic cover then the quadratic binpacking inequality

$$x_{ij} + x_{il} + x_{ji} + x_{jl} + x_{li} + x_{lj} \leq 1$$

is valid for P_{QH}. We separate these inequalities by enumeration and we add all that are violated.

Binpacking inequalities (Deng and Simchi Levi [9]) are introduced for the polytope of capacitated facility location problem (CFLP) with single assignment. Let $I' \subseteq I$ and $J \subseteq I$. Define $b(I')$ to be the minimum number of concentrators to be installed to assign all nodes in I'. The binpacking inequality

$$\sum_{i\in I'} \sum_{j\in J} x_{ij} - \sum_{j\in J} x_{jj} \leq |I'| - b(I')$$

is valid for P_{QH}.

Residual capacity inequalities (Leung and Magnanti [17]) are valid inequalities for the polytope of CFLP. Let $I^{'} \subseteq I$ and $J \subseteq I$. Define $D(I^{'}) = \sum_{i \in I^{'}} d_i$ and $r = D(I^{'}) - \lfloor \frac{D(I^{'})}{M} \rfloor M$. The residual capacity inequality

$$\sum_{i \in I^{'}} \sum_{j \in J} d_i x_{ij} - r \sum_{j \in J} x_{jj} \leq D(I^{'}) - r \lceil \frac{D(I^{'})}{M} \rceil$$

is valid for P_{QH}.

The separation algorithm for the binpacking and residual capacity inequalities is as follows: We look at all sets of the form $J = I \backslash \{l\}$ for some $l \in I$ such that $x^*_{ll} > 0$. For a given l, the binpacking inequality becomes

$$b(I^{'}) - \sum_{i \in I^{'}} x_{il} \leq \sum_{j \in I \backslash \{l\}} x_{jj}.$$

We take $I^{'} = \{i \in I : \frac{d_i}{M} - x^*_{il} > 0\}$ and $b(I^{'})$ to be the L_2 bound of Martello and Toth [18] for binpacking. We evaluate the binpacking inequality for this $I^{'}$ and J and add it to the formulation if it is violated. Otherwise we check whether the residual capacity inequality defined by the same sets $I^{'}$ and J is violated.

Effective Capacity Inequalities (Aardal et al. [1]) are also valid inequalities for the CFLP polytope. Let $J^{'} \subseteq I$. For each $j \in J^{'}$ choose $I_j \subseteq I$ and define $I^{'} = \cup_{j \in J^{'}} I_j$. Define also $\bar{M}_j = min\{M, \sum_{i \in I_j} d_i\}$ for each $j \in J^{'}$ and $\lambda = \sum_{j \in J^{'}} \bar{M}_j - D(I^{'})$. If $\lambda > 0$ then the effective capacity inequality

$$\sum_{j \in J^{'}} \sum_{i \in I_j} d_i x_{ij} + \sum_{j \in J^{'}} (\bar{M}_j - \lambda)^{+} (1 - x_{jj}) \leq D(I^{'})$$

is valid for P_{QH}. We separate these inequalities as described in Aardal [2], but we do not look for the P-depots structures.

Lifted W-2 Inequalities (Avella and Sassano [4]) The $W - 2$ inequalities are introduced for the p-median polytope and are also valid for the polytope of UCL. Let $W \subseteq I$ with $|W| \geq 4$ and $H \subset A_W = \{(i, j) \in A : i \in W, j \in W\}$ such that for each $j \in W$ there exists exactly one $i \in W$ such that $(i, j) \in H$. Define also $U = \{i \in W :$ there is no j such that $(i, j) \in H\}$. The $W - 2$ inequality

$$\sum_{(i,j) \in A_W} x_{ij} + \sum_{i \in U} \sum_{j \in I \backslash W} x_{ij} \leq |W| - 2$$

is valid for P_{QH} when $x_{lu} = 0$ for all $(l, u) \in A$ such that $l \in I \setminus W$ and $u \in U$. The lifting coefficients are given in Yaman [23].

We separate the W-2 inequalities using a heuristic algorithm. We consider triples $\{i, j, l\}$ such that $x^*_{ij} + x^*_{jl} + x^*_{li} + x^*_{ji} + x^*_{lj} + x^*_{il} > 0.8$. For a given set W, we determine sets H and U and lift the variables x_{ij}'s with $i \in I \setminus W$ and $j \in U$ in decreasing order of x_{ij}'s. If the lifted $W - 2$ inequality is violated, then we add it to the formulation and pass to the next triple. Otherwise, we add a node $s \notin W$ for which $\sum_{t \in W \backslash s_{min}} x^*_{ts} + \sum_{t \in W} x^*_{st} > 0.8$ where $s_{min} = argmin_{k \in W} x^*_{ks}$ and repeat the same steps.

k-triangle Inequalities (Yaman [23]) Consider the graph $H' = (I', A')$ where $I' \subseteq I$ with $|I'| = 2k+1$ for some $k \geq 1$ and number the nodes in I' from 1 to $2k+1$. The arc set A' consists of all arcs $(i, i+1)$ for $i = 1, .., 2k$ and all arcs (j, l) where j and l are both odd and $j > l$. Then the k-triangle inequality $\sum_{(i,j)\in A'} x_{ij} \leq k$ is valid for P_{QH}.

As the separation problem for the k-triangle inequalities is NP-complete (see Yaman [23]), we use a simple greedy heuristic. We consider ordered triples $\{i, j, l\}$ such that $x^*_{ij} > 0$ and $x^*_{ij} + x^*_{jl} + x^*_{li} > 0.8$. If the k-triangle inequality is violated, we add it to the formulation and pass to the next triple. Otherwise, we add nodes t and s not yet considered which have a contribution of at least 0.8 to the left hand side. Then we repeat the same steps.

Lifted k-leaf Inequalities (Yaman [23]) Given an arc $(i, j) \in A$ and a subset $I' \subseteq I \setminus \{i, j\}$ with $|I'| = k$, the k-leaf inequality

$$kx_{ij} + \sum_{t\in I\setminus\{i,j\}} (k-1)x_{it} + \sum_{t\in I'} x_{ti} + \sum_{t\in I'} x_{jt} \leq k$$

is valid for P_{QH} when $x_{li} = 0$ for all $l \in I \setminus (I' \cup \{i\})\})$. The lifting coefficients for $x_{li} = 0$ for all $l \in I \setminus (I' \cup \{i\})\})$ are given in Yaman [23] for certain sequences.

The k-leaf inequalities are separated exactly in $O(n^3)$ time as described in Yaman [23]. Then the variables x_{li} with $l \in I \setminus (I' \cup \{i\})$ are lifted by solving linear knapsack problems. We add the most violated inequality for each $(i, j) \in A$.

2-cycle Inequalities (Yaman [23]) Take a subset $D \subset I$ with $|D| = 3$ and a node $c \in I \setminus D$. Let C_d be a directed cycle on the nodes of D. Renumber the nodes such that $D = \{1, 2, 3\}$, the cycle is 1, 2, 3, 1, the node $c = 4$ and $I \setminus D = \{5, 6, .., n\}$. The 2-cycle inequality

$$2x_{12} + 2x_{23} + 2x_{31} + x_{14} + x_{24} + x_{34} + \sum_{i\in I\setminus\{4\}} x_{4i} \leq 3$$

is valid for P_{QH}. We separate these inequalities by considering triples $\{i, j, l\}$. If $x^*_{ij} + x^*_{jl} + x^*_{li} > 0.8$, then we add all violated 2-cycle inequalities with $1 = i, 2 = j, 3 = l$ and $m \in I \setminus \{i, j, l\}$. Otherwise, if $x^*_{ji} + x^*_{lj} + x^*_{il} > 0.8$, then we add all violated 2-cycle inequalities with $1 = j, 2 = i, 3 = l$ and $m \in I \setminus \{i, j, l\}$.

Odd Hole Inequalities (Padberg [19]) are valid inequalities for the stable set polytope and thus for the UCL polytope and for P_{QH}. These inequalities can be separated in polynomial time by solving a series of shortest path problems on a bipartite graph (see Grötschel et al. [13]).

5.5. Branch and Cut Algorithm and Computational Results

In this section, we present the basic parts of our branch and cut algorithm and discuss the computational results. The branch and cut algorithm is implemented in C++ using ABACUS 2.3 (see Jünger and Thienel [16]) and the LP solver CPLEX 7.0. The runs are taken on an Intel Pentium III, 1 GHz, 1 GB RAM running under Suse 7.2.

We start with the LP relaxation that contains constraints (2), (8) and $x_{ij} \geq 0$ for all $i, j \in I$ and $z_{jl} \geq 0$ for all $(j,l) \in A$. We run the preprocessing and strengthening algorithm. Then we give the formulation to ABACUS.

Primal Heuristic The solution (x^*, z^*) of the current LP relaxation is feasible if x^* is integer and (x^*, z^*) satisfies constraints (4). If x^* is integer but (x^*, z^*) does not satisfy constraints (4), we compute a feasible z' by taking $z'_{jl} = \sum_{k \in K} t^k x^*_{o(k)j} x^*_{d(k)l}$ for all $(j,l) \in A$. If x^* is not integer, we apply a rounding heuristic (Algorithm 4) to obtain an integer x' that satisfies the capacity constraints (5). If we can find such an x', we compute z' as described above. The heuristic is run each time an LP is solved.

Algorithm 4 Rounding Heuristic

for all $i \in I$ **do**
  if $x^*_{ii} \geq 0.5$ **then**
    $x'_{ii} \leftarrow 1$
  else
    find node $j = argmax_{l \in I} x^*_{il}$
    $x'_{ij} \leftarrow 1$
  end if
end for
for all $j \in I$ such that $x'_{jj} = 0$ **do**
  if there exists a node i such that $x'_{ij} = 1$ **then**
    $x'_{jl} \leftarrow 0$ for all $l \in I \setminus \{j\}$ and $x'_{jj} \leftarrow 1$
  end if
end for
$capfeas \leftarrow 0$ and $tour \leftarrow 1$
while not $capfeas$ and $tour \leq 15$ **do**
  $capfeas \leftarrow 1$ and increment $tour$
  for all $j \in I$ **do**
    if $x'_{jj} = 1$ **then**
      $s_j \leftarrow M - \sum_{i \in I_j} a_i - \sum_{i \in I_j} \sum_{m \in I \setminus I_j} T_{im}$ where $I_j = \{i \in I : x'_{ij} = 1\}$
    else
      $s_j \leftarrow 0$
    end if
    if $s_j < 0$ **then**
      $capfeas \leftarrow 0$
    end if
  end for
  if not $capfeas$ **then**
    if $\sum_{j \in I} s_j < 0$ **then**
      $x'_{j^*l} \leftarrow 0$ for all $l \in I \setminus \{j^*\}$ and $x'_{j^*j^*} \leftarrow 1$ where $j^* = argmax_{l \in I : x'_{ll} = 0} \sum_{i \in I} x^*_{il}$
    else
      sort nodes in I such that $s_1 \leq s_2 \leq .. \leq s_n$
      if $-s_1 \leq s_n$ and there exists a node $i \neq 1$ such that $x'_{i1} = 1$ and $s_n > a_i \geq -s_1$
      then
        $x'_{i1} \leftarrow 0$ and $x'_{in} \leftarrow 1$
      else
        $x'_{j^*l} \leftarrow 0$ for all $l \in I \setminus \{j^*\}$ and $x_{j^*j^*} \leftarrow 1$ where $j^* = argmax_{l \in I : x'_{ll} = 0} \sum_{i \in I} x^*_{il}$.
      end if
    end if
  end if
end while

Table 5. Comparison for branching and enumeration strategies

	breadth first			depth first			best first		
	nodes	LP's	CPU	nodes	LP's	CPU	nodes	LP's	CPU
var	225.8	2528.1	376.15	245.6	2633.9	379.28	216.4	2413.6	362.87
sos	751.4	8173.1	1381.12	1543.8	13921.8	2108.9	711.8	7878.8	1315.89

Branching and Enumeration Strategies We consider two branching strategies. The first strategy, denoted by *var*, is to branch on the most fractional x_{jj} variable, i.e., we branch on x_{jj} if $j = argmin_{i \in I} |x^*_{ii} - 0.5|$. If all x^*_{jj}'s are integer then we branch on the most fractional variable x_{ij}. The reason for giving a priority to variables x_{jj}'s is that we can expect to have a more balanced tree.

The second strategy, denoted by *sos*, is to branch on the assignment constraints (2). We find the first node i for which we can find a subset $J \subset I$ such that $\sum_{j \in J} x^*_{ij}$ is close to 0.5. Then in one branch we fix $\sum_{j \in J} x_{ij}$ to 1, and in the other branch we fix $\sum_{j \in I \setminus \{j\}} x_{ij}$ to 1.

Three enumeration strategies are tested: breadth first, depth first and best first.

In Table 5, we present the results for the two branching strategies and three enumeration strategies. We separate the quadratic cover inequalities, the strengthened projection inequalities and the step inequalities. For this analysis, we excluded the problems with 10 nodes, since all these problems were solved in less than 5 seconds after preprocessing. We added problems with 16 nodes with $Q \in \{2/3, 1/3, 1/4, 1/5, 1/6\}$.

We report the average values for the number of nodes in the branch and cut tree (denoted by *nodes*), the number of LP's solved (denoted by *LP's*) and the CPU time in seconds taken over all problems.

For the first branching strategy, there is not a big difference between the results obtained using different enumeration strategies. Still the best first strategy seems to be slightly better. Further, the CPU time for the second strategy is about four times the CPU time for the first strategy. Based on these results, we decided to use the first branching strategy and the best first enumeration strategy in our branch and cut tree algorithm.

Useful Cuts To see which cuts are useful to close the duality gap and decrease the CPU time, we conduct the following test. Initially we separate only the projection inequalities. Then we add sequentially each family of cuts to the existing cuts in the following order:

1. Lifted quadratic cover inequalities (cover)
2. Lifted strengthened projection inequalities (spro)
3. Step inequalities (step)
4. Quadratic binpacking inequalities on three nodes (quadbin)
5. Binpacking and residual capacity inequalities (binres)
6. Effective capacity inequalities (effcap)
7. Lifted $W - 2$ inequalities (w-2)
8. k-triangle inequalities (k-tri)
9. Lifted k-leaf inequalities (k-leaf)
10. 2-cycle inequalities (2-cycle)
11. Odd hole inequalities on the conflict graph (odd)

Table 6. Performance of cuts

ineq.	no. of violated ineq.'s	% imp. in gap	% imp. in LP's	% imp. in CPU
prep		(7.99)	(10334.56)	(1169.73)
cover	337.78	(7.43) 6.48	(5422.67) 21.05	(732.17) 12.04
spro	28120.00	(7.31) 1.65	(2844.33) 39.93	(440.25) 29.87
step	336.78	(7.28) 0.29	(2675.00) 1.24	(397.97) 1.26
quadbin	50.00	(7.28) 0.00	(2880.00) -4.05	(462.14) -7.45
binres	2.22	(7.28) 0.00	(2880.11) -0.01	(465.27) -0.41
effcap	0.00	(7.28) 0.00	(2880.11) 0.00	(467.23) -0.77
w-2	65.11	(7.17) 1.20	(2738.22) 3.07	(447.99) 0.23
k-tri	3.33	(7.17) 0.00	(2735.89) 0.04	(446.88) 0.03
k-leaf	1.22	(7.24) -0.72	(2786.33) -1.16	(455.33) -1.36
2-cycle	5.78	(7.24) 0.00	(2660.78) 2.31	(420.84) 2.56
odd	1.56	(6.10) 0.68	(1501.4) 3.59	(2722.36) -1331.00

In Table 6, we report the average number of all violated inequalities found during the branch and cut tree exploration, average percentage improvement in the duality gap, in the number of LP's solved and in the CPU time for each family of cuts added to the the previous ones of the above list. In parenthesis, we report also the average gap, the average number of LP's solved and the average CPU time (the percentage improvements are the averages of the percentage improvements taken over all problems). For odd hole inequalities, we did not test problems with 16 nodes as it takes too long to solve them.

Cover inequalities and strengthened projection inequalities are the most useful cuts. Step inequalities are also useful for problems with large Q values. Hence, we decided to keep these three families of inequalities in the branch and cut algorithm. Further, even if they seem to be useful to reduce the duality gap and the number of LP's solved, the odd hole inequalities take too long to separate and they are not violated often. So we remove the odd hole inequalities from the branch and cut algorithm.

When we separate all inequalities except the odd cycle inequalities, around 17% of the total CPU time is spent for separation and lifting of inequalities except projection inequalities. Preprocessing takes around 0.02% and the time spent for the LP's is around 39% of the total CPU time. Most of the remaining CPU time is spent for the separation of projection inequalities.

To decide for the rest of the inequalities, we conduct the following test. The branch and cut algorithm contains the first three families of inequalities. Then we add each family separately to see if it improves the results. The results are presented in Table 7. Detailed results are given in Yaman [23].

None of the families of inequalities seem very useful to improve the results. The k-triangle inequalities improve the number of LP's and the CPU time but mainly for problems with small Q values. As the separation algorithm for k-triangle inequalities is an enumerative heuristic, it can become time consuming for large n. Hence, we do not keep these inequalities in the branch and cut algorithm. In conclusion, we keep only cover, strengthened projection and step inequalities.

As the separation and lifting procedures for cover inequalities require the solution of quadratic problems, it can be useful to switch to procedures which approximate the quadratic knapsacks by linear knapsacks, as the problem size grows bigger. We switch for $n \geq 20$. We will discuss this further when we present the computational results.

Table 7. Performance of cuts-2

ineq.	no.of violated ineq.'s	% imp. in gap	% imp. in LP's	% imp. in CPU
cover, spro, step		(7.28)	(2675.00)	(397.97)
quadbin	50.00	(7.28) 0.00	(2880.00) -4.05	(462.14) -7.45
binres	2.22	(7.28) 0.00	(2675.11)-0.01	(401.08) -0.66
effcap	0.00	(7.28) 0.00	(2675.00) 0.00	(400.71) -1.20
w-2	64.44	(7.17) 1.20	(2771.89) -1.78	(447.68) -8.07
k-tri	3.11	(7.29) -0.17	(2679.22) 0.87	(399.77) 0.44
k-leaf	1.22	(7.28) 0.00	(2691.67) -1.53	(401.01) -2.05
2-cycle	6.22	(7.29) -0.17	(2684.56) 0.05	(401.23) -0.41

Cover inequalities are kept in the formulation. Other cuts are removed by ABACUS when they become inactive, i.e. when the corresponding slack variables become basic.

Tailing Off We apply the following rule: If the improvement over the five consecutive LP's is less than 0.05% then we branch.

Test Instances We have two sets of data from France Telecom, the first one has 17 nodes and the second one has 22 nodes. Using the traffic and cost data of these two sets and different values for the hub capacities M and the traffic demand level Q, we obtain different problems. Each problem name has the form ft, n, M, Q where n is the number of nodes, M is the capacity of a hub and Q is the demand level. We only consider feasible problems, since infeasibility is reported during preprocessing. In these problems, the traffic matrix T and the routing cost vector R are symmetric. So we remove half of the commodities and the links. We take the fixed demand of each terminal to be $a_i = \lceil \sum_{m \in I} (t_{im} + t_{mi}) \rceil$.

In the problems with 17 nodes, all commodities have positive traffic demand. This is not the case for the problems with 22 nodes. As a result, problems with 22 nodes are easier than the problems with 17 nodes in general.

We also generated several problems using the AP data set for hub location problems from the OR Library (see Beasley [6]). These problems have 10, 20, 25, 40 and 50 nodes. There are four problems of each kind, differing in the fixed cost for installing hubs and the capacity of hubs. We modified the data to fit to our algorithm (we take M to be the average of the hub capacities, we made the traffic and cost data symmetric and removed the commodities from a node to itself) and we created different problems by taking $Q \in \{1, 1/2, 1/3, 1/4, 1/5\}$. Each one of these problems is called nCK, Q where C is the capacity type and K is the cost type. Both C and K are either L or T. For C, the value T corresponds to tight capacities. For K, the value T corresponds to harder instances for the LHL.

Computational Results In Tables 8–12, we present the results for the test instances described above. The time limit is four hours. If the problem is not solved to optimality, we report the final gap if a feasible solution is found. For each problem we give the number of hubs installed in the optimal solution to evaluate the tightness of the instance capacities. We also report the number of violated inequalities of each family.

Table 8. The results for problems with 17 nodes

	BRANCH AND CUT				CPLEX		
problem	gap	nodes	LP's	CPU	gap	nodes	CPU
ft,17,10,1/2	0.61	7	50	3.02	4.62	1010	57.09
ft,17,10,1/3	12.79	1673	19689	5037.55	20.51	28331	time 20.27 %
ft,17,10,1/4	9.90	707	7595	1681.66	11.99	25723	11252.57
ft,17,10,1/5	7.58	199	2113	471.49	7.59	2797	1110.87
ft,17,10,1/6	6.26	183	2002	433.60	5.50	1327	504.10
ft,17,15,2/3	10.40	671	6858	1324.15	25.57	26400	time 10.90%
ft,17,15,1/2	17.33			memory 0.032%	30.83	18240	time 25.57%
ft,17,15,1/3	12.12	1137	11060	2648.59	17.16	16118	time 13.36%
ft,17,15,1/4	9.14	283	3214	898.29	9.58	6641	3757.29
ft,17,15,1/5	7.00	197	2868	744.82	6.20	2192	882.98
ft,17,15,1/6	5.34	91	1125	251.25	4.05	381	238.99
ft,17,20,1	5.69	99	988	103.41	18.55	100547	12902.41
ft,17,20,2/3	19.56			memory 1.59 %	37.37	15201	time 34.27%
ft,17,20,1/2	13.82	1185	11990	2522.11	27.02	12049	time 19.76%
ft,17,20,1/3	10.78	551	5169	1508.09	14.31	17724	11693.77
ft,17,20,1/4	7.62	263	2748	807.11	7.94	3217	1816.67
ft,17,20,1/5	7.07	139	2036	515.66	6.22	1351	441.31
ft,17,20,1/6	5.64	79	1018	211.27	4.15	406	276.61

For problems with 17 nodes, we compare the performance of our branch and cut algorithm with the one of CPLEX 7.0 MIP solver. The formulation given to CPLEX is the aggregated multicommodity flow formulation *QHL3* with the additional constraints $x_{ij} \leq x_{jj}$ for all $(i, j) \in A$. The variables whose values are fixed to 0 or 1 during the preprocessing are removed from this formulation. We report the duality gap, the number of nodes in the branch and cut tree and the CPU time in Table 8. The duality gap is computed using the best upper bound found by any of the two methods. When the program stops because of the time limit or lack of memory then we report the final duality gap computed using the best upper bound of the corresponding method.

For problems with large Q values, our branch and cut algorithm is faster than CPLEX 7.0. The biggest difference occurs for problem $ft, 17, 20, 1$. For this problem CPLEX takes around 125 times more CPU time. The branch and cut algorithm is not able solve problems $ft, 17, 15, 1/2$ and $ft, 17, 20, 2/3$ to optimality and stops with a gap of 0.032% and 1.59% respectively. The duality gaps at the root node are 17.33% and 19.56% which are very big. However for CPLEX, the gaps were 30.83% and 37.37% which shows that these were hard problems. The other problems that are not solved to optimality by CPLEX in four hours are solved by the branch and cut algorithm. The longest one $ft, 17, 10, 1/3$, takes less than one and a half hour.

As Q decreases, the duality gap we obtain using the aggregated flow formulation with CPLEX is smaller than the duality gap we obtain using the branch and cut algorithm. For these problems, the two solution methods do not differ much for the CPU time. There are two problems where CPLEX is faster than the branch and cut algorithm but the biggest difference is around one minute.

It is possible to use the preprocessing algorithm to strengthen the formulation before giving it to the CPLEX MIP solver. To see if this can improve the results, we add all inequalities that are generated during preprocessing to the formulation. Moreover, we also add knapsack inequalities of the form $\sum_{i \in I} d_i x_{ij} \leq M x_{jj}$ for each $j \in I$. In

Table 9. CPLEX on *QHL3* with and without strengthening

	without strengthening			with strengthening		
problem	gap	nodes	CPU	gap	nodes	CPU
ft,17,10,1/2	4.62	1010	57.09	1.36	75	9.25
ft,17,10,1/3	20.51	28331	time 20.27 %	9.62	18506	time 3.17 %
ft,17,10,1/4	11.99	25723	11252.57	6.91	8673	8558.75
ft,17,10,1/5	7.59	2797	1110.87	5.78	2062	1258.97
ft,17,10,1/6	5.50	1327	504.10	4.84	1139	562.28

Table 10. The number of violated inequalities for problems with 17 nodes

problem	no. of conc.	cover	spro	step
ft,17,10,1/2	7+5	7	448	26
ft,17,10,1/3	8	813	266331	6145
ft,17,10,1/4	6	484	88057	520
ft,17,10,1/5	5	212	17673	0
ft,17,10,1/6	5	205	14321	0
ft,17,15,2/3	9+1	123	95847	3041
ft,17,15,1/3	5	1163	113225	675
ft,17,15,1/4	5	522	22247	10
ft,17,15,1/5	5	196	15659	0
ft,17,15,1/6	5	8	5969	0
ft,17,20,1	8+3	30	11386	774
ft,17,20,1/2	5	1606	126807	1287
ft,17,20,1/3	3	880	35368	20
ft,17,20,1/4	4	216	15063	0
ft,17,20,1/5	3	2	1157	0
ft,17,20,1/6	5	0	5674	0

Table 9, we compare the results with and without these inequalities for the first five problems with 17 nodes.

We see that strengthening the formulation decreases the duality gap and the number of nodes in the tree for all problems. However, the CPU time decreases for problems with large Q values and increases for problems with small Q values. We can conclude that for the hard problems, the strengthening improves the results considerably. Still, the branch and cut algorithm performs better than CPLEX on each of these problems.

Results of Tables 8 and 9 allow to conclude that preprocessing is useful when using formulation *QHL3* and confirm that formulation *QHL3* performs worse than QHL1 even with preprocessing. Similar tests were also performed for formulation QHL2m and yielded to the same conclusion as for QHL3.

In the second column of Table 10, we report the number of hubs installed for each problem. When some of the nodes are assigned to themselves and removed from the problem during preprocessing, we add the number of such nodes to the number of hubs installed.

We also report the number of violated inequalities of each family in Table 10. We observe that the step inequalities are violated when Q is large. After a certain value of Q there is no more violated step inequalities. The number of violated quadratic cover and strengthened projection inequalities also decreases as Q decreases since the capacities become loose.

Table 11. The results for problems with 22 nodes

problem	gap	nodes	LP's	CPU	no. of conc	cover	spro	step
ft,22,10,1/2	2.42	131	597	155.54	5	729	3852	0
ft,22,10,1/3	4.72	1305	5040	870.84	4	4336	20291	1
ft,22,10,1/4	2.76	229	857	165.47	4	783	2704	0
ft,22,10,1/5	0.13	3	14	3.94	3	34	18	0
ft,22,10,1/6	0.00	1	4	1.82	3	11	11	0
ft,22,15,2/3	5.90	1057	4596	894.94	4	3018	15577	0
ft,22,15,1/2	2.78	81	360	87.18	4	457	1380	0
ft,22,15,1/3	0.00	1	7	3.65	3	24	7	0
ft,22,15,1/4	0.34	9	28	5.96	2	24	11	0
ft,22,15,1/5	0.00	1	7	2.87	2	4	3	0
ft,22,15,1/6	0.00	1	6	2.40	2	3	4	0
ft,22,20,1	6.87			memory 1.38%				
ft,22,20,2/3	3.15	41	215	70.34	3	351	627	0
ft,22,20,1/2	1.01	7	53	22.49	3	61	37	0
ft,22,20,1/3	0.07	3	20	8.36	2	14	3	0
ft,22,20,1/4	0.00	1	10	4.20	2	0	4	0
ft,22,20,1/5	0.00	1	10	3.83	2	0	3	0
ft,22,20,1/6	0.00	1	7	2.86	2	0	4	0

Before moving to other test instances, we test the following strategies for the unsolved problems $ft, 17, 15, 1/2$ and $ft, 17, 20, 2/3$:

1. Use Algorithm 3 during the preprocessing and separate the cover inequalities on linear knapsacks (42) to avoid solving quadratic problems.
2. Separate also the inequalities that are removed from the branch and cut algorithm except the odd hole inequalities.
3. Give the objective value of the best solution found as the starting upper bound.
4. Use depth-first search.

We did one change at a time. Problem $ft, 17, 15, 1/2$ was solved to optimality in 12613.79 seconds of CPU time when we solved linear knapsacks rather than quadratic problems. The other changes did not improve the results. We also solved the other problems applying the first strategy. The results did not change significantly.

Based on this result and some preliminary tests, we decided to use Algorithm 3 during the preprocessing and to separate the cover inequalities on linear knapsacks for the remaining test instances as they have bigger sizes than the problems we solved up to now. We also adapted the separation of projection inequalities so that we only consider the commodities with nonzero traffic demands.

In Table 11, we report the results for problems with 22 nodes. There is one problem that is not solved to optimality as we ran out of memory. The remaining instances were solved rather fast except problem $ft, 22, 15, 2/3$. There is only one step inequality violated. As the separation of step inequalities does not take much time, removing them would not change the total CPU time significantly.

In Table 12, we report the results for the hub location problems from the OR library. We omit the results for problems with 10 nodes as all of these problems are solved at the root node in less than one second. We also remove problems of type LL, TL with $Q = 1/4$ and $Q = 1/5$ since in these problems the capacity constraints are redundant.

Table 12. The results for hub location problems

problem	gap	nodes	LP's	CPU	no. of conc	cover	spro	step
20LL,1	2.68	41	199	52.34	4	151	755	1
20TL,1	0.33	3	22	5.73	4	47	73	0
20LL,1/2	0.00	1	3	0.98	2	0	6	0
20TL,1/2	0.00	1	2	0.70	2	0	1	0
20LL,1/3	0.00	1	5	1.61	2	0	13	0
20TL,1/3	0.00	1	4	1.21	2	0	4	0
20LT,1/3	0.87	15	58	14.80	4	64	165	0
20TT,1/3	1.20	13	44	6.40	4	39	133	0
20LT,1/4	0.50	17	48	10.2	3	18	106	0
20TT,1/4	0.92	19	62	12.52	3	46	62	0
20LT,1/5	0.00	1	2	0.66	2	0	4	0
20TT,1/5	0.00	1	2	0.68	2	0	2	0
25LL,1	1.54	23	138	155.41	5	379	829	0
25TL,1	4.98	281	1404	823.46	5	1870	7757	19
25LL,1/2	0.00	1	7	6.88	2	6	16	0
25TL,1/2	0.00	1	7	4.14	2	3	8	0
25LL,1/3	0.00	1	5	4.97	2	0	11	0
25TL,1/3	0.00	1	4	4.18	2	0	4	0
25LT,1/3	2.11	65	262	193.81	5	396	1231	5
25TT,1/3	5.91	471	1728	751.55	5	1473	4743	32
25LT,1/4	0.00	1	6	7.26	4	20	18	0
25TT,1/4	0.00	1	8	4.78	3	15	17	1
25LT,1/5	0.88	17	52	32.48	4	10	115	0
25TT,1/5	1.02	5	14	10.58	3	7	14	0
40LL,1	8.63			memory 7.46 %				
40TL,1	6.88			memory 6.08 %				
40LL,1/2	2.43			memory 0.8 %				
40TL,1/2	0.23	5	23	298.2	3	26	52	0
40LL,1/3	0.00	1	6	86.23	3	0	15	0
40TL,1/3	0.00	1	7	89.11	2	2	8	0
40LT,1/3	4.14			memory 3.34 %				
40TT,1/3	8.17			memory 6.82 %				
40LT,1/4	0.66	25	92	1006.84	4	198	358	0
40TT,1/4	1.09	21	103	324.09	4	102	336	0
40LT,1/5	2.68			memory 1.09 %				
40TT,1/5	0.07	3	14	163.55	3	16	32	0
50LL,1	3.87			memory 2.34 %				
50TL,1				memory				
50LL,1/2	6.10			memory 5.21 %				
50TL,1/2	21.74			memory 21.18%				
50LL,1/3	0.00	1	6	326.78	3	0	10	0
50TL,1/3	0.29	3	12	393.7	2	0	15	0
50LT,1/3	7.21			memory 7.02 %				
50TT,1/3	6.60			memory 6.30 %				
50LT,1/4				memory				
50TT,1/4	20.87			memory 20.49 %				
50LT,1/5	0.53	19	66	1406.92	4	102	187	0
50TT,1/5	0.55	7	43	1037.19	3	69	99	0

The problems of type TT, LT are infeasible for $Q = 1$ and $Q = 1/2$ and are not considered. So we report the results for 48 problems.

Among these 48 problems, 34 of them are solved to optimality. For the 14 problems not solved, we ran out of memory. All problems with 20 and 25 nodes are solved to optimality. We could solve the problems with 40 and 50 nodes only for small values

of Q. Once again, we see that the difficulty of the problem depends a lot on the demand and capacity level of the problem.

For two problems, the program stopped before finding a feasible solution. In fact, a feasible solution can be found by assigning each node to itself. In most cases, such a solution has a very big cost compared to the one of the optimal solution (the number of hubs installed is not more than five for the problems solved), which will probably not improve the performance of the algorithm.

There are also two problems where the algorithm stopped with a gap of more than 20%. Looking at the gaps of the other problems, we can say that this may be due to the bad quality of the upper bound rather than the lower bound. In fact, for bigger size problems, we believe that a better heuristic can improve the performance of the branch and cut algorithm.

In conclusion, problems with small values of Q can be solved by the CPLEX MIP solver for reasonable sizes. The hard problems are the ones where Q is large, so that the traffic demand is high and capacities are tight. The branch and cut algorithm is able to solve such problems of reasonable size.

Acknowledgements. The research of the first author was partially supported by the Banque Nationale de Belgique. The research of the second author was supported by France Telecom R&D under contract no. 99 1B 774. Their support is gratefully acknowledged.

References

1. Aardal, K., Pochet, Y., Wolsey, L.A.: Capacitated Facility Location: Valid Inequalities and Facets. Math. Oper. Res. **20**, 562–582 (1995)
2. Aardal, K.: Capacitated Facility Location: Separation Algorithms and Computational Experience. Math. Program. **81**, 149–175 (1998)
3. Andrews, M., Zhang, L.: Approximation Algorithms for Access Network Design. Algorithmica **34**, 197–215 (2002)
4. Avella, P., Sassano, A.: On the p-Median Polytope. Math. Program. **89**, 395–411 (2001)
5. Balas, E.: Facets of the Knapsack Polytope. Math. Program. **8**, 146–164 (1975)
6. Beasley, J.E.: OR-Library: distributing test problems by electronic mail. J. Oper. Res. Soc. **41**, 1069–1072 (1990)
7. Campbell, J.F., Ernst, A.T., Krishnamoorthy, M.: Hub Location Problems. In: Facility Location: Applications and Theory, Z. Drezner, H.W. Hamacher (eds.), Springer, 2002, pp. 373–407
8. Dantzig, G.B.: On the Significance of Solving Linear Programming Problems with Some Integer Variables. The Rand Corporation, document, 1958, p. 1486
9. Deng, Q., Simchi-Levi, D.: Valid Inequalities, Facets and Computational Results for the Capacitated Concentrator Location Problem. Department of Industrial Engineering and Operations Research, Columbia University, New York, NY 10027-6699, 1992
10. Ernst, A.T., Krishnamoorthy, M.: Solution Algorithms for the Capacitated Single Allocation Hub Location Problem. Ann. Oper. Res. **86**, 141–159 (1999)
11. Gourdin, E., Labbé, M., Yaman, H.: Telecommunication and Location. In: Facility Location: Applications and Theory, Z. Drezner, H.W. Hamacher (eds.), Springer, 2003, pp. 275–305
12. Goemans, M.X., Williamson, D.P.: The primal-dual method for approximation algorithms and its application to network design problems. In: Approximation Algorithms for NP-Hard Problems, D.S. Hochbaum (ed.), PWS Publishing Company, 1997, pp. 144–191
13. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer-Verlag, Berlin, 1988
14. Gu, Z., Nemhauser, G.L., Savelsbergh, M.W.P.: Cover Inequalities for 0-1 Linear Programs: Computation. INFORMS J. Comput. **10**, 427–437 (1998)
15. Hammer, P.L., Johnson, E.L., Peled, U.N.: Facets of Regular 0-1 Polytopes. Math. Program. **8**, 179–206 (1975)

16. Jünger, M., Thienel, S.: The ABACUS system for branch-and-cut-and-price algorithms in integer programming and combinatorial optimization. Softw. Pract. Experience **30**, 1325–1352 (2000)
17. Leung, J.M.Y., Magnanti, T.L.: Valid Inequalities and Facets of the Capacitated Plant Location Problem. Math. Program. **44**, 271–291 (1989)
18. Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations, Wiley, New York, 1990
19. Padberg, M.W.: On the Facial Structure of Set Packing Polyhedra. Math. Program. **5**, 199–215 (1973)
20. Skorin-Kapov, D., Skorin-Kapov, J., O'Kelly, M.: Tight linear programming relaxations of uncapacitated p-hub median problem. Eur. J. Oper. Res. **94**, 582–593 (1996)
21. Swamy, C., Kumar, A.: Primal-dual Algorithms for Connected Facility Location Problems. In: Approximation algorithms for combinatorial optimization, K. Jansen, S. Leonardi, V. Vazirani (eds.), 5th international workshop, APPROX 2002, Proceedings. Lect. Notes Comput. Sci. 2462, Springer, Berlin, 2002 pp. 256–269
22. Wolsey, L.: Faces for a Linear Inequality in 0-1 Variables. Math. Program. **8**, 165–178 (1975)
23. Yaman, H.: Concentrator Location in Telecommunication Networks, Ph.D. Thesis, Université Libre de Bruxelles, 2002. Available at http://smg.ulb.ac.be/
24. Yuan, D.: An Annotated Bibliography in Communication Network Design and Routing. In: Optimization Models and Methods for Communication Network Design and Routing. Ph.D. Thesis, Department of Mathematics, Linköping University, Sweden, 2001