Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Computers & Operations Research 32 (2005) 3227–3245

computers & operations research

www.elsevier.com/locate/cor

# Solving the hub location problem with modular link capacities

Hande Yaman[a,*], Giuliana Carello[b,1]

[a]Department of Industrial Engineering, Bilkent University, Bilkent, 06800 Ankara, Turkey
[b]Dipartimento di Automatica ed Informatica, Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 Torino, Italy

Available online 7 July 2004

**Abstract**

This paper deals with a capacitated hub location problem arising in the design of telecommunications networks. The problem is different from the classical hub location problem in two ways: the cost of using an edge is not linear but stepwise and the capacity of a hub restricts the amount of traffic transiting through the hub rather than the incoming traffic. In this paper both an exact and a heuristic method are presented. They are compared and combined in a heuristic concentration approach to investigate whether it is possible to improve the results within limited computational times.
© 2004 Elsevier Ltd. All rights reserved.

*Keywords:* Hub location; Branch and cut; Tabu search; Heuristic concentration

## 1. Introduction and problem description

Optimization problems related to the design and installation of telecommunication networks gain interest due to the growing importance of telecommunications. Since nowadays installing telecommunication networks is expensive, an optimization phase with the goal of minimizing costs is needed in the design process. This paper deals with an optimization problem arising in the design of a quite common network architecture, the so-called *backbone/tributary network*, in which two kinds of nodes are present. *Terminal nodes* represent origins and destinations of the traffic demands to be routed. Usually, connecting all pairs of terminals nodes by direct links is a very costly solution. So, the traffic originating in different terminal nodes must be collected in nodes called *hubs*, which receive

* Corresponding author.
*E-mail addresses:* hyaman@bilkent.edu.tr (H. Yaman), giuliana.carello@polito.it (G. Carello).
[1] Also for correspondence.

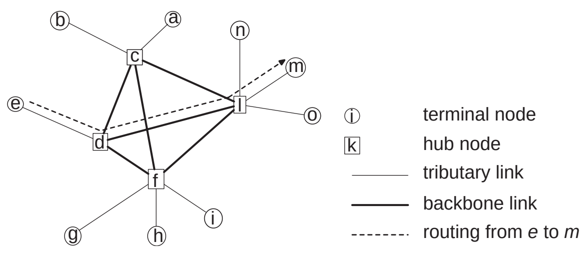

Fig. 1. Backbone/tributary network instance.

traffic from terminal nodes and route it through other hubs towards the destinations. The network connecting the terminal nodes to the hub to which they are assigned is called the *tributary network* and the network connecting the hubs is called the *backbone network*. Since the cost per unit of traffic is usually cheaper in the backbone than in the tributary network, collecting traffic allows to reduce the costs.

Different kinds of optimization problems may arise in backbone/tributary network design. Klincewicz [1] gives a survey on these problems and Yuan [2] gives an annotated bibliography in communication network design and routing problems that presents over 600 references. Problems differ in many aspects, e.g. it is possible to focus on the design of the backbone and/or the tributary networks or on the hub location problem. Furthermore, it is possible to consider different kinds of costs, e.g. to consider fixed costs for installing hubs and/or variable costs for using them as well as fixed costs for installing the needed capacity on the edges or variable costs for using the edges. It is possible to limit the capacities of both edges and hubs. Moreover, in some problems, an a priori structure is specified for the networks. For example, one can look for a network where the backbone is a tree and the tributary networks are stars.

In this paper, we focus on networks with complete backbone and star tributary networks and solve the location and dimensioning problems simultaneously. The locations of the hubs are chosen among the terminal nodes and each terminal node is assigned to exactly one hub. All the outgoing traffic of a terminal node is sent to the hub to which it is assigned, and is routed through the other hubs towards the destinations. All the traffic incoming in the terminal node is routed from its origins through the hubs towards the hub to which the node is assigned. The hubs are fully connected and each terminal is directly connected to the hub to which it is assigned. The traffic between two nodes goes from its origin to the hub of the origin, then on the direct edge to the hub of the destination and finally to its destination. In Fig. 1, an example of the considered network is shown. In the network the hub nodes are $c$, $d$, $f$ and $l$. The edges of the backbone network, which is fully connected, are represented by bold lines. The routing of the traffic from terminal node $e$ to terminal node $m$ is represented by dashed lines. First the traffic is routed to $d$, to which $e$ is assigned. Then it is routed on the direct edge from hub $d$ to hub $l$, to which $m$ is assigned, and finally it reaches $m$ through the star tributary network.

We consider fixed costs of installing hubs and fixed costs of installing the needed capacity on each edge. The capacity needed to route the traffic on an edge is provided by the installation of an integer number of bidirectional *links* of fixed capacity. The link capacity can be different for the backbone and for the tributary edges. For each edge, the cost of establishing a link on this edge is given and this cost depends on the length of the edge. Furthermore, there is a capacity which is the maximum amount of traffic that can be routed through a hub. Given a traffic matrix, which represents the traffic demands between pairs of terminal nodes, the problem is to find the set of nodes which receive hubs and the assignment of each terminal node and to install capacities on links, such that all the traffic is routed respecting the capacity constraints. The aim is to minimize the total cost of the network, which is the sum of hub costs and link costs.

Since the backbone network is fully connected and each terminal is directly connected to its hub, the problem belongs to the family of hub location problems. There are several kinds of hub location problems, which are different for the capacity constraints, for the constraint on the number of hubs or for the assignment requirements. A survey on hub location problems and on most recently proposed algorithms can be found in Campbell et al. [3].

The problem considered here is called the *capacitated single assignment hub location problem with modular link capacities* (*HMLC*). The HMLC is an NP-hard problem (see Yaman [4] for the proof for a special case).

There are many papers dealing with different kinds of hub location problems, but, to the best of our knowledge, none of them considers all the features of HMLC or proposes exact methods for HMLC. Among the papers dealing with problems close to our one, we cite [5–8]. Boland et al. [5] consider the problem where each terminal node can be assigned to several hubs. Ernst and Krishnamoorthy [7] and Labbé et al. [8] consider the capacitated hub location problem with single assignment where there is a cost for routing the traffic but not for installing links. In [7], the capacity of each hub limits only the traffic incoming in the hub. However, in [8], the limit is on the amount of traffic which transits through a hub as it is the case in the HMLC. Finally, Carello et al. [6] consider a generalization of the HMLC, but they do not propose exact methods.

Besides, many other papers have been published dealing with the optimal design of backbone/tributary networks, taking into account technological features (among others, we can cite [9–11]). However, such papers usually consider tributary and backbone network topologies which are different from the ones considered in this paper (e.g. tree or ring backbone topologies), leading to quite different problems especially from the exact solution point of view.

We formulate the HLMC as a quadratic mixed integer programming problem and compare different linearizations. We present two different approaches to solve the HMLC, an exact method and a metaheuristic, based on methods developed for very similar problems (see [8] and [6], respectively). The exact method is a branch and cut algorithm and the metaheuristic is a two-level local search. The branch and cut algorithm is based on a linearization which has an exponential number of constraints.

The aim of the paper is to compare these two methods and to investigate whether it is possible to combine them in order to obtain a better performance in terms of computational time. The metaheuristic is run first to find an initial upper bound to use in the branch and cut algorithm.

Then, the two methods are combined through *heuristic concentration* (see Rosing and ReVelle [12] and Rosing and Hodgson [13]). The metaheuristic is used to limit the number of variables considered by the exact method, i.e. the branch and cut is applied on a subset of the potential locations of the hubs which has been chosen by the heuristic algorithm.

The paper is organized as follows: in the next section a mixed integer programming formulation is derived for the HLMC. A summary of known valid inequalities as well as a new family of valid inequalities are presented and the exact method is described. In Section 3, the metaheuristic is presented. Finally, Section 4 is devoted to the computational results and conclusions.

## 2. Formulation and the exact method

Let $I$ denote the set of terminal nodes with $|I| = n$. Define $K$ to be the set of commodities. There is a commodity per each directed pair of nodes. For nodes $i \in I$ and $m \in I$, $t_{im}$ denotes the traffic from $i$ to $m$. 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 number of links needed to route the traffic adjacent at node $i$. Then, $a_i = \max\{\lceil \sum_{m\in I} t_{im}/Q^a \rceil, \lceil \sum_{m\in I} t_{mi}/Q^a \rceil\}$ where $Q^a$ is the capacity of a tributary link. The cost of installing $a_i$ links 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 $C_{ii}$. If node $i$ becomes a hub node, then the total amount of traffic transiting through node $i$ cannot be larger than its capacity $Q^{\mathrm{h}}$.

Let $E = \{\{j,l\} : j \in I, l \in I\backslash\{j\}\}$ and $A = \{(j,l) : j \in I, l \in I\backslash\{j\}\}$. We denote by $R_{jl}$ the cost of installing a backbone link on edge $\{j,l\} \in E$. If nodes $j$ and $l$ receive hubs, then the amount of traffic on arc $(j,l)$ is the sum of traffic from nodes assigned to $j$ to nodes assigned to $l$. Each backbone link has capacity $Q^{\mathrm{b}}$. The capacity of links installed on edge $\{j,l\}$ cannot be less than the maximum of traffic on arcs $(j,l)$ and $(l,j)$.

We define the following variables: The assignment variable $x_{ij}$ is equal to 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 traffic on arc $(j,l) \in A$ and $w_{jl}$ to be the number of backbone links installed on edge $\{j,l\} \in E$.

The HMLC can be formulated as follows:

$$\min \sum_{i\in I}\sum_{j\in I} C_{ij}x_{ij} + \sum_{\{j,l\}\in E} R_{jl}w_{jl} \tag{1}$$

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

$$x_{ij} \leqslant x_{jj} \quad \forall i \in I, \quad j \in I\backslash\{i\} \tag{3}$$

$$\sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})x_{ij} - \sum_{i\in I}\sum_{m\in I} t_{im}x_{ij}x_{mj} \leqslant Q^{\mathrm{h}}x_{jj} \quad \forall j \in I \tag{4}$$

$$z_{jl} \geqslant \sum_{i\in I}\sum_{m\in I} t_{im}x_{ij}x_{ml} \quad \forall (j,l) \in A \tag{5}$$

$$Q^{\mathrm{b}}w_{jl} \geqslant z_{jl} \quad \forall \{j,l\} \in E \tag{6}$$

$$Q^{\mathrm{b}}w_{jl} \geqslant z_{lj} \quad \forall \{j,l\} \in E \tag{7}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in I, j \in I \tag{8}$$

$$w_{jl} \in \mathbb{Z}_+ \quad \forall \{j,l\} \in E. \tag{9}$$

Constraints (2), (3) and (8) imply that each node should be assigned to exactly one node which is a hub. Capacity constraints (4) state that the capacity of a hub cannot be less than the amount of traffic that transits through this hub. The left-hand side of constraint (4) for $j \in I$ is equal to the sum of the total traffic adjacent at nodes that are assigned to *j* minus the sum of the traffic between pairs of nodes that are both assigned to *j*. This subtraction is done to avoid double counting.

The traffic variables $z_{jl}$'s are computed by means of constraints (5). Constraints (6) and (7) impose that the capacity of links installed on a given edge should be at least the maximum of traffic on the two arcs with the same extremes. Finally, the objective function (1) consists of the cost of locating hubs and installing links on the backbone and tributary networks.

Constraints (4) and (5) are quadratic constraints. We first present a linearization of constraints (4) using some properties of optimal solutions. Then we discuss different ways of linearizing constraints (5).

The capacity constraint (4) for $j \in I$ can be rewritten as

$$\sum_{i\in I}\sum_{m\in I} t_{mi}x_{ij} + \sum_{i\in I}\sum_{m\in I} t_{im}x_{ij}(1-x_{mj}) \leqslant Q^{\mathrm{h}}x_{jj}.$$

Because of constraints (6) and (7) and the fact that $R_{jl} \geqslant 0$ for all $\{j,l\} \in E$, there exists an optimal solution where all constraints (5) are tight. This solution satisfies

$$\begin{aligned}\sum_{l\in I\setminus\{j\}} z_{jl} &= \sum_{l\in I\setminus\{j\}}\sum_{i\in I}\sum_{m\in I} t_{im}x_{ij}x_{ml}\\ &= \sum_{i\in I}\sum_{m\in I} t_{im}x_{ij}\sum_{l\in I\setminus\{j\}} x_{ml}\\ &= \sum_{i\in I}\sum_{m\in I} t_{im}x_{ij}(1-x_{mj}).\end{aligned}$$

So for $j \in I$, we can replace constraint (4) by

$$\sum_{i\in I}\sum_{m\in I} t_{mi}x_{ij} + \sum_{l\in I\setminus\{j\}} z_{jl} \leqslant Q^{\mathrm{h}}x_{jj}. \tag{10}$$

Labbé et al. [8] give a branch and cut algorithm to solve a very similar problem called the *quadratic capacitated hub location problem with single assignment* (*QHL*). In QHL, $w_{jl}$'s are continuous and $w_{jl} = z_{jl} + z_{lj}$ for $\{j,l\} \in E$. Moreover, the capacity constraints have the form

$$\sum_{i\in I}\sum_{m\in I} (t_{im}+t_{mi})x_{ij} + \sum_{i\in I}\sum_{m\in I} (t_{im}+t_{mi})x_{ij}(1-x_{mj}) \leqslant Q^{\mathrm{h}}x_{jj}.$$

They compare several formulations for QHL which are different in the way they linearize constraints (5). The classical hub location linearization (see Skorin-Kapov et al. [14]) uses four-indexed variables: $X_{jl}^{im}$ is 1 if the traffic from *i* to *m* travels from *j* to *l* and 0 otherwise. Then $X_{jl}^{im} = x_{ij}x_{ml}$. We can now linearize constraints (5) using the following set of constraints:

$$\sum_{l\in I} X_{jl}^{im} = x_{ij} \quad \forall i, m, j \in I \tag{11}$$

$$\sum_{l \in I} X_{lj}^{im} = x_{mj} \quad \forall i, m, j \in I \tag{12}$$

$$z_{jl} \geqslant \sum_{i \in I} \sum_{m \in I} t_{im} X_{jl}^{im} \quad \forall (j, l) \in A. \tag{13}$$

Another linearization using variables $X_{jl}^{im}$'s (see Dantzig [15]) can be obtained by replacing constraints (5) by the following set of constraints:

$$X_{jl}^{im} \geqslant x_{ij} + x_{ml} - 1 \quad \forall i, m, j, l \in I$$

$$X_{jl}^{im} \leqslant x_{ij} \quad \forall i, m, j, l \in I$$

$$X_{jl}^{im} \leqslant x_{ml} \quad \forall i, m, j, l \in I$$

$$z_{jl} \geqslant \sum_{i \in I} \sum_{m \in I} t_{im} X_{jl}^{im} \quad \forall (j, l) \in A.$$

As $R_{jl} \geqslant 0$ for all $\{j, l\} \in E$, the second and third sets of constraints are redundant.

The disadvantage of these formulations is their size. The number of variables is $O(n^4)$. In both formulations, there is no cost related to variables $X_{jl}^{im}$'s. So these variables can be projected out. Labbé et al. [8] do the projection for the second linearization and obtain the formulation where constraints (5) are replaced by

$$z_{jl} \geqslant \sum_{(i,m) \in K'} t_{im}(x_{ij} + x_{ml} - 1) \quad \forall K' \subseteq K, (j, l) \in A. \tag{14}$$

This new formulation has $O(n^2)$ variables and exponentially many constraints. Its LP relaxation is very weak compared to the LP relaxation of the formulation obtained using the first linearization. Still it can be improved by adding the so-called *projection inequalities*. These inequalities come from the projection of variables $X_{jl}^{im}$'s in the first linearization. Computationally, this new formulation with the projection inequalities outperforms the formulation that uses the first linearization. So the authors devise a branch and cut algorithm based on this formulation.

Labbé et al. [8] compare these formulations with two other formulations based on multicommodity flows. These multicommodity flow formulations are valid for QHL when the cost of routing the traffic satisfies the triangle inequality. As we take $w_{jl}$'s to be integers, these formulations are not valid in our case.

We modify the branch and cut algorithm presented by Labbé et al. [8] to solve the HMLC. We impose the integrality of variables $w_{jl}$'s. Here we summarize briefly the algorithm. It is implemented in C++ using ABACUS 2.3 (see Jűnger and Thienel [16]) and CPLEX 7.0 as LP solver.

The first step of the algorithm is a preprocessing based on the capacity constraints. The aim of the preprocessing is to compute lower bounds on the amount of traffic which travels on the backbone links. Precisely, for $i \in I$, $j \in I$, we compute a lower bound on the amount of traffic from node *i* to nodes in $I \setminus \{i, j\}$ which are not assigned to *j* when *i* is assigned to *j*. We use the iterative procedure *Compute Traffic* given in [8]. A lower bound on the number of hubs to be installed is also computed. Valid inequalities which impose that two nodes cannot be assigned to the same hub if their demands exceed the capacity are also added. Then the resulting formulation is given to ABACUS.

Labbé et al. [8] identify the following families of cuts as useful to close the duality gap and to reduce the CPU time for instances of QHL: projection, cover, strengthened projection and step inequalities. Labbé and Yaman [17] give new valid inequalities which dominate a class of projection inequalities. In the following, we generalize this class to a larger class which dominates all projection inequalities used by Labbé et al. [8] in their branch and cut algorithm. In our branch and cut algorithm, we use this new family of inequalities. We use also the cover, strengthened projection and step inequalities. In the following, we briefly describe these inequalities.

Let $F$ be the feasible set of the HMLC and $P = conv(F)$.

**Theorem 2.1** (*Labbé et al. [8]*). *The projection inequality*

$$\sum_{j\in S}\sum_{l\in T} z_{jl} \geqslant \sum_{(i,m)\in K'} t_{im}\left(\sum_{j\in S} x_{ij} + \sum_{l\in T} x_{ml} - 1\right) \tag{15}$$

*where S and T are nonempty disjoint subsets of I and $K' \subseteq K$ is a valid inequality for P.*

Labbé and Yaman [17] give a new class of inequalities which dominate the projection inequalities when *S* and *T* are singletons. The theorem below is a generalization of their result.

**Theorem 2.2.** *The improved projection inequality*

$$\begin{aligned}\sum_{j\in S}\sum_{l\in T} z_{jl} \geqslant & \sum_{(i,m)\in K':i\notin S,m\notin T} t_{im}\left(\sum_{j\in S\setminus\{m\}} x_{ij} + \sum_{l\in T\setminus\{i\}} x_{ml} + x_{im} + x_{mi} - 1\right)\\ &+ \sum_{(i,m)\in K':i\in S,m\notin T} t_{im}\left(\sum_{j\in S\setminus\{m\}} x_{ij} + \sum_{l\in T} x_{ml} + x_{im} - 1\right)\\ &+ \sum_{(i,m)\in K':i\notin S,m\in T} t_{im}\left(\sum_{j\in S} x_{ij} + \sum_{l\in T\setminus\{i\}} x_{ml} + x_{mi} - 1\right)\\ &+ \sum_{(i,m)\in K':i\in S,m\in T} t_{im}\left(\sum_{j\in S} x_{ij} + \sum_{l\in T} x_{ml} - 1\right)\end{aligned} \tag{16}$$

*where S and T are nonempty disjoint subsets of I and $K' \subseteq K$ is a valid inequality for P.*

**Proof.** Let $(i, m) \in K'$. Consider the following cases:

- $i \notin S$ and $l \notin T$: If $x_{im} = 1$, then $\sum_{j\in S\setminus\{m\}} x_{ij} = 0$ and $x_{mi} = 0$. Moreover as $m \notin T$, we also have $\sum_{l\in T\setminus\{i\}} x_{ml} = 0$. The case where $x_{mi} = 1$ is similar. If $x_{im} = x_{mi} = 0$, then the validity follows from the validity of inequality (15).

- $i \in S$ and $l \notin T$: If $x_{im} = 1$, then $\sum_{j\in S\setminus\{m\}} x_{ij} = 0$. Also, as $m \notin T$, $\sum_{l\in T} x_{ml} = 0$.
- $i \notin S$ and $l \in T$: If $x_{mi} = 1$, then $\sum_{j\in S} x_{ij} = 0$ since $i \notin S$. Moreover, $\sum_{l\in T\setminus\{i\}} x_{ml} = 0$. □

Now it is easy to prove the following:

**Proposition 2.1.** *For S and T which are nonempty disjoint subsets of I and $K' \subseteq K$, inequality* (16) *dominates inequality* (15).

Because of the result in Proposition 2.1, we can expect inequalities (16) to be more useful than inequalities (15) in a branch and cut framework. The example below shows that for given sets $S$ and $T$ inequality (16) can be violated when inequality (15) is not.

**Example 2.1.** Consider the following instance: $I = \{1, 2, 3, 4, 5\}$, the only nonzero traffic is from node 4 to 5 and the assignment vector is as follows: $x_{11} = x_{22} = x_{33} = 1$, $x_{41} = 0.2$, $x_{42} = 0.3$, $x_{45} = 0.5$, $x_{53} = 0.5$, $x_{55} = 0.5$ and $z_{13} = z_{23} = 0$. Consider $S = \{1, 2\}$ and $T = \{3\}$. Inequality (15) is $z_{13} + z_{23} \geqslant t_{45}(x_{41} + x_{42} + x_{53} - 1) = 0$ whereas inequality (16) is $z_{13} + z_{23} \geqslant t_{45}(x_{41} + x_{42} + x_{53} + x_{45} + x_{54} - 1) = 0.5t_{45}$. The latter cuts off the fractional solution.

The separation of inequalities (16) is as follows: For given $S$ and $T$, and a solution $(x, z, w)$, the right-hand side of inequality (16) is maximized at

$$K' = \left\{(i, m) \in K : i \notin S, m \notin T \text{ and } \sum_{j\in S\setminus\{m\}} x_{ij} + \sum_{l\in T\setminus\{i\}} x_{ml} + x_{im} + x_{mi} > 1\right\}$$
$$\cup \left\{(i, m) \in K : i \in S, m \notin T \text{ and } \sum_{j\in S\setminus\{m\}} x_{ij} + \sum_{l\in T} x_{ml} + x_{im} > 1\right\}$$
$$\cup \left\{(i, m) \in K : i \notin S, m \in T \text{ and } \sum_{j\in S} x_{ij} + \sum_{l\in T\setminus\{i\}} x_{ml} + x_{mi} > 1\right\}$$
$$\cup \left\{(i, m) \in K : i \in S, m \in T \text{ and } \sum_{j\in S} x_{ij} + \sum_{l\in T} x_{ml} > 1\right\}.$$

If inequality (16) for this choice of $K'$ is violated, then it is the most violated inequality for $S$ and $T$. We choose sets $S$ and $T$ as described by Labbé et al. [8].

The cover inequalities are based on capacity constraints. They impose that if the demand of a subset of nodes exceeds the capacity of a hub, then all these nodes cannot be assigned to the hub at the same time. So, the sum of the corresponding assignment variables is less than or equal to the cardinality of the set minus one.

**Theorem 2.3** (*Labbé et al. [8]*). *A subset $C \subseteq I$ such that*

$$\sum_{i\in C}\sum_{m\in I} t_{mi} + \sum_{i\in C}\sum_{m\in I\setminus C} t_{im} > Q^{\mathrm{h}}$$

*is called a quadratic cover. If $C \subseteq I$ is a quadratic cover, then the quadratic cover inequality $\sum_{i\in C} x_{ij} \leqslant (|C|-1)x_{jj}$ is valid for $P$.*

The projection inequalities do not take into account the capacity constraints. So they can be strengthened in the following way:

**Theorem 2.4** (*Labbé et al. [8]*). *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*

$$\sum_{j\in S}\sum_{l\in T} z_{jl} \geqslant \sum_{i\in I_S}\left[\sum_{m\in I_i} t_{im}\sum_{l\in T} x_{ml} - \tau_i\left(1-\sum_{j\in S} x_{ij}\right)\right] \tag{17}$$

*is valid for $P$ where*

$$\begin{aligned}
\tau_i = \max & \sum_{m\in I_i} t_{im}\sum_{l\in T} u_{ml}\\
\text{s.t.} & \sum_{l\in T} u_{ml} \leqslant 1 \quad \forall m \in I_i\\
& \sum_{m\in I_i}\left(\sum_{s\in I} t_{sm} + \sum_{s\in I\setminus I_i} t_{ms}\right)u_{ml} + \sum_{m\in I_i}\sum_{s\in I_i} t_{ms}u_{ml}(1-u_{sl}) \leqslant Q^{\mathrm{h}} \quad \forall l \in T\\
& u_{ml} \in \{0,1\} \quad \forall m \in I_i, l \in T
\end{aligned}$$

*for all $i \in I_S$.*

Finally, step inequalities give lower bounds for the traffic on an arc using the number of nodes assigned to the extremes of the arc.

**Theorem 2.5** (*Labbé et al. [8]*). *Let $I' \subseteq I$ and $T^{\mu}(I')$ denote a lower bound on the traffic on arc $(j,l) \in A$ when $\mu$ terminals of set $I'$ are assigned to $j$ and $l$ for $\mu = 0,\ldots,|I'|$. Define $\mathrm{d}T^{\mu+1}(I') = T^{\mu+1}(I') - T^{\mu}(I')$ for all $\mu = 0,\ldots,|I'|-1$. If $\mathrm{d}T^{\mu+1}(I') \geqslant \mathrm{d}T^{\mu}(I')$ for all $\mu = 0,\ldots,|I'|-1$, then the step inequality*

$$z_{jl} \geqslant T^{\mu}(I') + \mathrm{d}T^{\mu+1}(I')\left[\sum_{i\in I'}(x_{ij}+x_{il}) - \mu\right]$$

*is valid for $P$ for any $\mu = 0,\ldots,|I'|-1$.*

The cover, strengthened projection and step inequalities are separated and lifted as explained in Labbé et al. [8].

Different from the QHL, in HMLC, we have link variables that take integer values. So one should also branch on these variables when they turn out to be fractional. We adapted the following branching strategy: first branch on the assignment variables and then on the link variables. The rationale behind this is that there are two possible values for assignment variables whereas link variables can take any nonnegative integer value. Another reason is that, once the assignment vector is integer, one can easily compute the best values of link variables and find feasible solutions. Having an integer assignment vector also makes the separation of inequalities (16) an easy task. Among assignment variables, we give priority to $x_{jj}$ variables. This is expected to result in a more balanced tree. We explore the nodes using best-first search strategy.

Each time an LP is solved, we use the rounding heuristic given by Labbé et al. [8] to find an integer feasible solution. The initial upper bound is given by the heuristic which is described in the following section.

## 3. The heuristic method

The heuristic method, which is adapted from a heuristic approach proposed for similar problems by Carello et al. [6][2], is a local search approach based on the decomposition of the problem into two subproblems, the location subproblem and the assignment subproblem. Carello et al. [6] develop and compare different local search-based metaheuristics to solve a similar problem, in which different kinds of traffic are considered as well as different assignment requirements. Moreover, additional costs for the equipment of hubs appear in the total cost.

In the heuristic method first the location subproblem is solved by a tabu search that is applied to find the best set of hubs. To complete the solution, represented by a set of hubs, and to compute the objective value, all the terminal nodes must be assigned. In the tabu search step, for each neighbor, the assignments are defined by means of a greedy algorithm. Then a basic local search is applied on the 20 best sets of hubs found by the tabu search to improve the solution of the assignment problem computed by the greedy algorithm.

In addition to the best solution, the algorithm produces also a subset of nodes that represents, in a sense, the best hubs selected by the heuristic. The hubs opened in the best solution belong to this subset as well as two other hubs which appear most often in the set of best heuristic solutions. This set is called the *concentration set* and the problem where hubs can be chosen only among the nodes of this set is called the *concentrated problem*. The concentrated problem is solved using the branch and cut algorithm. For some instances, it is much easier to solve the concentrated problem than the original problem.

The main steps of the proposed heuristic algorithm are the following:

1. An initial feasible solution is found, by means of a greedy algorithm that starts with an empty set of hubs and adds hubs one by one until a feasible solution is found. The hubs are added trying to keep as low as possible the cost of assignments (for more details, see [6]).
2. A tabu search step is applied to the location subproblem.

[2] A preliminary heuristic method has been developed in a research project joint with Telecom Italia (Turin Research & Innovation Laboratories) and a patent application has been filed to cover this issue.

3. A local search on the assignment problem is applied to the 20 best sets of hubs found by the previous step.

The heuristic is implemented in C++.

### 3.1. Tabu search for the location problem

Tabu search is a widely known local search-based metaheuristic (see Glover, [18,19]), which overcomes the limit of basic local search, that may be captured in local minima, by accepting as new current solutions even non improving neighbors. In order to avoid cycles, a memory of the most recently visited solutions is kept by means of a tabu list, in which the moves that produced the last current solutions are stored. Neighbors generated by such tabu moves are discarded. To avoid discarding good solutions generated by tabu moves, all the neighbors which satisfy an aspiration criterion are accepted even if they are generated by tabu moves. The tabu search stops after a given numbers of iterations without improvement of the best solution found.

In the tabu search phase, the solution is represented by the set of chosen hubs. Given the set of chosen hubs, the assignments of terminal nodes are defined by means of a greedy algorithm, based on a greedy algorithm for generalized assignment problem (see [20]). The greedy assigns terminals one by one trying to minimize the disadvantageous assignments, as it is described in [6]: if it does not manage to find a feasible allocation, the neighbor is discarded as unfeasible.

The neighborhood is generated by applying three different moves:

- *adding move*: a new hub is opened;
- *removing move*: a hub is removed from the set of hub nodes;
- *swapping move*: a hub is removed and a new one is opened in another terminal node.

Similar moves have been proposed in [9] as perturbation method. The dimension of such neighborhood is $O(n^2)$.

The main features of the tabu search have been set according to computational experience:

- *stopping criterion*: the tabu search stops after 200 iterations without improvement;
- *tabu list dimension*: in the tabu list the moves are stored that produced the last six neighbors chosen as new current solutions;
- *aspiration criterion*: a solution generated by a tabu move is accepted if it improves upon the best solution found so far.

### 3.2. Local search for the assignment problem

The second step, namely the local search on the assignment subproblem, is applied on each of the 20 best sets of hubs selected by the tabu search step. In the local search the set of hubs is given and the solution is represented by the allocations of all terminals. Starting from the allocations provided by the greedy algorithm, the local search tries to improve the sum of network costs (except hub installation costs) by changing the allocations of one or two terminals. A first improvement strategy is applied, namely the first improving neighbor found replaces the current solution. The local search stops when no improving neighbor is found visiting a complete neighborhood.

Given a current solution, the neighborhood is generated by applying two different moves:

- the assignment of a terminal node is moved from the current hub to another;
- the hubs to which two terminal nodes are assigned are swapped: e.g. if terminal node $i$ is assigned to hub $k$ and terminal node $j$ to hub $m$ in the current solution, a neighbor is generated by assigning $i$ to $m$ and $j$ to $k$.

Both moves are applied only if the new assignments do not violate capacity constraints. The dimension of the neighborhood is $O(n^2)$.

## 4. Computational results

The proposed algorithms are tested on both real life instances and instances derived from the OR Library (see Beasley [21]). There are three instances, with 12, 17 and 49 nodes, derived from real-life instances (see [6]). Capacities are modified to avoid infeasible or uncapacitated instances. Other instances are derived using the AP data set (for hub location problems) from the OR Library.[3] These instances have 10, 20, 25, 40 or 50 nodes. In the OR Library set, there are four instances for each number of nodes, which are different for the kind of capacity and costs considered. There may be tight and loose capacities as well as tight and loose hub costs. These instances are modified to be solved by the branch and cut algorithm as follows: the traffic matrix is made symmetric and $Q^{\mathrm{h}}$ is taken to be the integer nearest to the average value of hub capacities. Further, the capacities are modified to avoid infeasible or uncapacitated instances.

For both sets of instances, other instances are derived from the original ones by scaling capacities and hub costs. Up to five additional instances are derived from each original instance: capacities are reduced, by factors 0.6 and 0.8, and hub costs are multiplied by 0.1, 0.01 and 0.001. These instances are named as the original instance name.cap06, .cap08, .cost01, .cost001 and .cost0001, respectively. Not all the instances with reduced capacities are feasible: computational results are reported only for feasible instances.

For each instance, the heuristic algorithm is run to obtain an upper bound and a concentration set. The branch and cut algorithm is initialized with the upper bound of the heuristic and solves the original and concentrated problems. It is run on an Intel Pentium III, 1 GHz, 1 Gb RAM running under Suse 7.2 with a limit on the maximum resident set size set to 900 Mb and a limit on CPU time set to 4 h. The runs with the heuristic are taken on an AMD XP 2000 at 1.66 GHz, 256 Mb RAM, running under Windows XP.

Let *UBH* denote the upper bound given by the heuristic. Define also *UB*, *LB*, *DB*, *CUB*, *CLB* and *CDB* to be the best upper bound, final lower bound and the dual bound (the lower bound at the root node), respectively, for the original and concentrated problems. In the tables, we report for each problem:

- $Hgap = ((UBH - UB)/UB) * 100$ and $CHgap = ((UBH - CUB)/CUB) * 100$
- $Cgap = ((CUB - UB)/CUB) * 100$

[3] Instances derived from OR Library are available upon request from authors.

Table 1
Results for problems with 10 nodes

| Problem name | Original problem | | | Concentrated problem | | | |
|---|---|---|---|---|---|---|---|
| | Hgap | Dgap | CPU | Cgap | CHgap | CDgap | CCPU |
| A1 | 24.93 | 7.63 | 14.26 | 0.00 | 24.93 | 7.63 | 8.32 |
| A1.cap08 | 6.16 | 1.50 | 1.03 | 0.00 | 6.16 | 1.50 | 0.89 |
| A1.cost01 | 20.23 | 8.27 | 11.47 | 0.00 | 20.23 | 8.27 | 5.94 |
| A1.cost001 | 2.78 | 5.01 | 6.40 | 0.00 | 2.78 | 3.88 | 1.60 |
| A1.cost0001 | 1.43 | 6.07 | 16.92 | 0.00 | 1.43 | 5.47 | 6.27 |
| A2 | 0.00 | 0.28 | 2.60 | 0.00 | 0.00 | 0.28 | 2.56 |
| A2.cost01 | 0.00 | 1.54 | 1.91 | 0.00 | 0.00 | 1.54 | 1.54 |
| A2.cost001 | 0.00 | 4.90 | 12.29 | 0.00 | 0.00 | 4.80 | 5.84 |
| A2.cost0001 | 0.00 | 6.17 | 16.53 | 0.00 | 0.00 | 6.15 | 6.25 |
| A3 | 14.38 | 14.95 | 22.62 | 0.00 | 14.38 | 14.95 | 15.31 |
| A3.cap08 | 3.59 | 0.87 | 0.65 | 0.00 | 3.59 | 0.87 | 0.66 |
| A3.cost01 | 14.87 | 12.19 | 11.96 | 0.00 | 14.87 | 12.19 | 8.15 |
| A3.cost001 | 0.00 | 7.59 | 6.93 | 0.00 | 0.00 | 2.48 | 1.40 |
| A3.cost0001 | 1.41 | 7.52 | 15.57 | 0.00 | 1.41 | 6.09 | 8.93 |
| A4 | 0.00 | 0.11 | 1.64 | 0.00 | 0.00 | 0.11 | 1.48 |
| A4.cost01 | 0.00 | 0.97 | 1.61 | 0.00 | 0.00 | 0.97 | 1.81 |
| A4.cost001 | 0.00 | 4.52 | 15.68 | 0.00 | 0.00 | 4.38 | 7.86 |
| A4.cost0001 | 0.00 | 5.98 | 17.10 | 0.00 | 0.00 | 5.94 | 6.51 |

- $Dgap = ((UB - DB)/UB) * 100$ and $CDgap = ((CUB - CDB)/CUB) * 100$
- $Fgap = ((UB - LB)/UB) * 100$ and $CFgap = ((CUB - CLB)/CUB) * 100$.
- *CPU* (resp. *CCPU*): CPU time to solve the original problem (resp. concentrated problem) (in h:min:s)

We do not report the CPU time for the heuristic as it requires less than 5 min even for the largest instances.

We first report the results for the instances derived from the OR library. In Table 1, the results for instances with 10 nodes are presented. All problems are solved to optimality in less than 1 min. For all instances, the concentrated problem has the same optimal value as the original problem. This suggests that the heuristic gives a good concentration set but it cannot find a good assignment for problems where *Hgap* is positive. Except for two instances, the concentrated problems are solved in shorter times. The difference in CPU times goes up to a factor of 4.

We also solved the four original instances with 10 nodes using CPLEX 8.1.0 MIP Solver. We used the linearization where constraints (5) are replaced by (11)–(13). The results are given in Table 2. We observe that while the branch and cut algorithm took less than a minute to solve these problems, CPLEX MIP solver took more than 19 min for the easiest instance. The branch and cut algorithm is more than 100 times faster for the four instances and for instance A4, it is almost 700 times faster than the MIP solver of CPLEX. This is mostly because the number of variables in the formulation given to CPLEX is $O(n^4)$ and it takes a long time to solve the LPs. The branch and cut algorithm is based on a formulation which has $O(n^2)$ variables and so it takes much shorter to solve the LP’s. As the difference between the performances of the branch and cut algorithm and CPLEX is very large and is not expected to change in favor of CPLEX as $n$ increases, we did not do the test for the remaining instances.

Table 2
Results with CPLEX for problems with 10 nodes

| Problem | CPU (s) |
|---|---|
| A1 | 2776.06 |
| A2 | 1282.16 |
| A3 | 2687.89 |
| A4 | 1141.61 |

Table 3
Results for problems with 20 nodes

| Problem name | Original problem | | | | Concentrated problem | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Hgap | Dgap | Fgap | CPU | Cgap | CHgap | CDgap | CFgap | CCPU |
| B1 | 2.15 | 1.14 | 0.00 | 47:48.69 | 0.00 | 2.15 | 1.14 | 0.00 | 52:37.99 |
| B1.cap06 | 0.58 | 25.07 | 24.77 | | −0.04 | 0.62 | 25.04 | 24.74 | |
| B1.cap08 | 0.36 | 32.55 | 31.85 | | 0.05 | 0.31 | 33.01 | 31.77 | |
| B1.cost01 | 0.00 | 3.92 | 0.00 | 01:42:56.45 | 0.00 | 0.00 | 3.39 | 0.00 | 01:45:48.94 |
| B1.cost001 | 0.00 | 6.63 | 0.00 | 03:36:36.70 | 0.00 | 0.00 | 6.54 | 0.00 | 01:02:51.94 |
| B1.cost0001 | 0.39 | 8.16 | 1.78 | | −1.07 | 1.47 | 6.54 | 0.00 | 01:15:33.29 |
| B2 | 0.00 | 1.18 | 0.00 | 03:48.48 | 0.00 | 0.00 | 1.18 | 0.00 | 03:49.32 |
| B2.cap06 | 3.14 | 0.71 | 0.23 | | 0.00 | 3.14 | 0.71 | 0.24 | |
| B2.cap08 | 0.38 | 0.98 | 0.00 | 47:37.78 | 0.00 | 0.38 | 0.97 | 0.00 | 04:06.46 |
| B2.cost01 | 0.14 | 4.32 | 0.00 | 01:32:34.74 | 0.00 | 0.14 | 3.60 | 0.00 | 01:32:05.66 |
| B2.cost001 | 0.32 | 6.79 | 0.00 | 03:38:14.92 | 0.00 | 0.32 | 7.27 | 0.00 | 01:01:19.29 |
| B2.cost0001 | 0.00 | 8.72 | 3.31 | | 0.00 | 0.00 | 7.09 | 0.00 | 44:40.27 |
| B3 | 0.77 | 0.83 | 0.00 | 01:38:36.71 | 0.00 | 0.77 | 0.83 | 0.00 | 01:38:30.12 |
| B3.cap06 | 0.00 | 25.77 | 25.50 | | 0.00 | 0.00 | 25.77 | 25.50 | |
| B3.cap08 | 0.00 | 33.87 | 32.81 | | −0.29 | 0.29 | 33.68 | 32.38 | |
| B3.cost01 | 0.00 | 3.26 | 0.00 | 01:00:40.17 | 0.00 | 0.00 | 3.24 | 0.00 | 01:21:45.83 |
| B3.cost001 | 0.00 | 6.38 | 0.00 | 55:42.17 | 0.00 | 0.00 | 6.78 | 0.00 | 01:41:12.88 |
| B3.cost0001 | 0.00 | 7.10 | 0.88 | | 0.00 | 0.00 | 7.33 | 0.00 | 03:31:15.11 |
| B4 | 0.03 | 1.08 | 0.00 | 32:09.68 | 0.00 | 0.03 | 1.08 | 0.00 | 32:35.43 |
| B4.cap06 | 3.63 | 0.63 | 0.23 | | 0.00 | 3.63 | 0.63 | 0.24 | |
| B4.cap08 | 2.12 | 0.44 | 0.00 | 01:17:22.64 | 0.00 | 2.12 | 0.44 | 0.00 | 01:17:43.01 |
| B4.cost01 | 0.13 | 3.50 | 0.00 | 01:34:31.07 | 0.00 | 0.13 | 3.51 | 0.00 | 01:12:03.52 |
| B4.cost001 | 0.32 | 6.61 | 0.00 | 02:09:28.38 | 0.00 | 0.32 | 7.00 | 0.00 | 01:02:19.05 |
| B4.cost0001 | 0.00 | 7.94 | 2.16 | | −0.08 | 0.08 | 7.92 | 0.00 | 02:08:46.14 |

In Table 3, we present the results for instances with 20 nodes. The branch and cut algorithm could not solve 10 problems to optimality. The final gap is very large for some of those problems. These are indeed the problems where the capacity is reduced. Problems with tight capacities turn out to be harder as also reported by Labbé et al. [8]. For B1.cap06 and B3.cap08, we found better upper bounds solving the concentrated problems. For B2.cap08, the concentrated problem is solved more than 10 times faster than the original problem. For other instances where the capacity is reduced, restricting the potential hubs to the concentration set does not make the problem easier.

Table 4
Results for problems with 25 nodes

| Problem name | Original problem | | | Concentrated problem | | | |
|---|---|---|---|---|---|---|---|
| | Hgap | Dgap | Fgap | Cgap | CHgap | CDgap | CFgap |
| C1 | 0.00 | 31.91 | 31.68 | 0.00 | 0.00 | 31.91 | 31.64 |
| C1.cap06 | 0.55 | 21.53 | 21.29 | 0.55 | 0.00 | 21.96 | 21.72 |
| C1.cap08 | 0.00 | 0.89 | 0.54 | 0.00 | 0.00 | 0.89 | 0.54 |
| C1.cost01 | 4.56 | 25.95 | 25.48 | 4.36 | 0.00 | 24.98 | 24.53 |
| C1.cost001 | 6.53 | 11.89 | 10.61 | 4.66 | 1.57 | 9.83 | 8.23 |
| C1.cost0001 | 15.73 | 4.97 | 3.62 | 12.93 | 0.77 | 7.23 | 5.49 |
| C2 | 0.00 | 0.91 | 0.25 | 0.00 | 0.00 | 0.91 | 0.29 |
| C2.cap06 | 0.00 | 1.38 | 1.17 | 0.00 | 0.00 | 1.46 | 1.16 |
| C2.cap08 | 2.23 | 1.10 | 0.84 | 1.82 | 0.38 | 1.71 | 1.59 |
| C2.cost01 | 0.74 | 2.79 | 1.54 | 0.28 | 0.45 | 1.94 | 1.24 |
| C2.cost001 | 0.03 | 5.45 | 3.72 | −0.30 | 0.32 | 4.45 | 3.33 |
| C2.cost0001 | 0.00 | 3.97 | 1.83 | 0.00 | 0.00 | 3.72 | 1.89 |
| C3 | 0.00 | 34.68 | 34.53 | 0.00 | 0.00 | 34.68 | 34.53 |
| C3.cap06 | 0.00 | 22.15 | 21.98 | 0.00 | 0.00 | 22.15 | 21.98 |
| C3.cap08 | 0.00 | 0.49 | 0.37 | 0.00 | 0.00 | 0.49 | 0.37 |
| C3.cost01 | 0.00 | 23.34 | 22.57 | 0.00 | 0.00 | 23.25 | 22.47 |
| C3.cost001 | 0.00 | 12.61 | 11.92 | 0.00 | 0.00 | 12.52 | 12.02 |
| C3.cost0001 | 9.34 | 7.62 | 6.21 | 7.98 | 0.61 | 7.76 | 6.26 |
| C4 | 0.00 | 12.83 | 1.13 | 0.00 | 0.00 | 12.83 | 1.12 |
| C4.cost01 | 0.00 | 14.99 | 7.26 | 0.00 | 0.00 | 14.45 | 5.20 |
| C4.cost001 | 5.35 | 11.10 | 4.45 | −0.18 | 5.54 | 8.67 | 2.46 |
| C4.cost0001 | 0.00 | 10.71 | 5.87 | 0.00 | 0.00 | 7.83 | 4.85 |

The other unsolved problems are the ones where the fixed cost of installing a hub is reduced by a factor of 1000. The concentrated versions of these problems are all solved to optimality. As the cost of a hub decreases, we tend to open more hubs. That is probably why restricting the hubs to the concentration set results in easier problems. For B1.cost0001 and B4.cost0001, optimal values of the concentrated versions give the best upper bounds.

For .cost001 instances, except B3.cost001, concentrated problems are solved two to three times faster than the original problems and they have the same optimal value as the original problems.

In total, for four unsolved problems, *Cgap* is negative, implying that the best solution found for the concentrated problem gives the best upper bound.

The results for problems of sizes 25, 40 and 50 are given in Tables 4, 5 and 6, respectively. As these problems are not solved to optimality, we do not present the CPU times. The concentrated version gave the best upper bounds for five instances. The largest improvement is for instance D1.cost01 where the upper bound is improved by around 16%. For other instances, restricting the set of hubs does not seem useful.

In Tables 7–9 we present the results for real-life instances. Problems with 12 nodes are all solved to optimality in less than 1 h. We observe that the concentrated problems are easier to solve for these instances. For three instances, the concentrated problem has the same optimal value as the original problem. For the other two instances, there is a gap between the optimal values of these problems. This

Table 5
Results for problems with 40 nodes

| Poblem name | Original problem | | | Concentrated problem | | | |
|---|---|---|---|---|---|---|---|
| | Hgap | Dgap | Fgap | Cgap | CHgap | CDgap | CFgap |
| D1 | 0.00 | 2.41 | 2.13 | 0.00 | 0.00 | 2.41 | 2.13 |
| D1.cap06 | 0.00 | 10.89 | 10.17 | 0.00 | 0.00 | 10.89 | 9.76 |
| D1.cap08 | 0.00 | 19.82 | 19.77 | 0.00 | 0.00 | 19.82 | 19.76 |
| D1.cost01 | 4.05 | 24.51 | 24.34 | −16.18 | 20.89 | 2.67 | 1.90 |
| D1.cost001 | 8.43 | 10.16 | 8.93 | 7.78 | 0.00 | 10.24 | 9.96 |
| D1.cost0001 | 3.20 | 8.32 | 6.96 | 3.10 | 0.00 | 6.06 | 5.58 |
| D2 | 0.08 | 8.89 | 8.18 | 0.08 | 0.00 | 8.95 | 7.72 |
| D2.cost01 | 0.98 | 15.11 | 13.91 | 0.97 | 0.00 | 14.69 | 9.20 |
| D2.cost001 | 0.00 | 12.42 | 11.49 | −0.10 | 0.10 | 8.45 | 7.04 |
| D2.cost0001 | 0.00 | 11.56 | 10.01 | 0.00 | 0.00 | 7.15 | 6.11 |
| D3 | 0.00 | 11.71 | 11.66 | 0.00 | 0.00 | 11.71 | 11.66 |
| D3.cap06 | 0.00 | 12.71 | 9.51 | 0.00 | 0.00 | 12.71 | 3.71 |
| D3.cap08 | 0.00 | 17.38 | 17.34 | 0.00 | 0.00 | 17.38 | 17.34 |
| D3.cost01 | 24.29 | 2.51 | 0.93 | 19.54 | 0.00 | 21.07 | 20.25 |
| D3.cost001 | 2.16 | 15.46 | 15.10 | 2.11 | 0.00 | 12.77 | 12.22 |
| D3.cost0001 | 0.00 | 9.05 | 7.90 | 0.00 | 0.00 | 7.07 | 6.73 |
| D4 | 0.00 | 10.31 | 6.84 | 0.00 | 0.00 | 10.31 | 0.96 |
| D4.cost01 | 0.00 | 11.72 | 9.80 | 0.00 | 0.00 | 11.50 | 8.25 |
| D4.cost001 | 5.04 | 12.93 | 11.28 | 4.80 | 0.00 | 12.40 | 9.74 |
| D4.cost0001 | 0.00 | 14.82 | 13.50 | −2.26 | 2.26 | 10.65 | 9.39 |

implies that the optimal hubs are not all in the concentration set and so the heuristic could not find good locations for hubs. Still, for these two problems the optimal solutions of the concentrated problems give better upper bounds than the heuristic.

The branch and cut algorithm could not solve the problems with 17 nodes. But the concentrated versions are solved to optimality except one. For this unsolved problem, the final gap for the original version is more than 15%. The final gap for the concentrated version is 6.68%, but the upper bound is not better than the final upper bound of the original problem. There are two other problems for which the optimal value of the concentrated version is worse than the upper bound of the original version. For one instance, the upper bound obtained for the concentrated problem is better than the upper bound of the heuristic.

Neither the original nor the concentrated versions of problems with 49 nodes could be solved. The final upper bounds are the same as the initial bounds for all problems. The maximum final gap is less than 7%.

In summary, the branch and cut algorithm solves instances of small sizes faster than CPLEX MIP solver. But for larger sizes, it cannot prove optimality. Still, for some problems, it improves the upper bound of the heuristic and also gives some idea about the quality of the solution.

Restricting the set of locations for hubs to the concentration set is usually useful for instances where the fixed cost of installing hubs is small, i.e. instances of type .cost0001. For these instances, concentration makes the problems easier to solve. If the optimal hubs are in the set given by the heuristic, then the concentrated version solves the problem and saves computational time. Most problems with tight capacities remain hard even in their concentrated versions. For some real-life instances, the concentrated problems turned out to be easier to solve. For problems with 12 nodes, when the heuristic returned the

Table 6
Results for problems with 50 nodes

| Problem name | Original problem | | | Concentrated problem | | | |
|---|---|---|---|---|---|---|---|
| | Hgap | Dgap | Fgap | Cgap | CHgap | CDgap | CFgap |
| E1 | 0.86 | 25.21 | 24.96 | 0.86 | 0.00 | 25.63 | 25.60 |
| E1.cap06 | 0.00 | 10.79 | 10.43 | 0.00 | 0.00 | 10.80 | 10.11 |
| E1.cap08 | 0.00 | 19.85 | 19.75 | 0.00 | 0.00 | 19.85 | 19.75 |
| E1.cost01 | 0.00 | 25.04 | 24.80 | 0.00 | 0.00 | 19.97 | 18.83 |
| E1.cost001 | 0.00 | 15.48 | 14.82 | 0.00 | 0.00 | 9.95 | 9.68 |
| E1.cost0001 | 0.00 | 9.68 | 9.68 | 0.00 | 0.00 | 5.89 | 5.64 |
| E2 | 0.00 | 13.17 | 12.85 | 0.00 | 0.00 | 12.43 | 11.86 |
| E2.cost01 | 0.00 | 20.94 | 20.53 | 0.00 | 0.00 | 18.70 | 17.68 |
| E2.cost001 | 0.00 | 17.14 | 16.98 | 0.00 | 0.00 | 14.74 | 13.31 |
| E2.cost0001 | 0.00 | 100.00 | 13.62 | 0.00 | 0.00 | 7.26 | 6.89 |
| E3 | 0.00 | 27.99 | 27.97 | 0.00 | 0.00 | 27.99 | 27.97 |
| E3.cap06 | 0.00 | 12.68 | 11.10 | 0.00 | 0.00 | 12.68 | 10.66 |
| E3.cap08 | 0.00 | 24.36 | 24.31 | 0.00 | 0.00 | 24.36 | 24.31 |
| E3.cost01 | 0.00 | 21.69 | 20.27 | 0.00 | 0.00 | 21.68 | 20.25 |
| E3.cost001 | 3.12 | 14.66 | 14.07 | 3.03 | 0.00 | 13.00 | 12.58 |
| E3.cost0001 | 0.00 | 8.05 | 7.24 | 0.00 | 0.00 | 6.26 | 6.14 |
| E4 | 0.00 | 11.31 | 9.84 | 0.00 | 0.00 | 10.32 | 7.21 |
| E4.cost01 | 0.00 | 12.12 | 11.27 | 0.00 | 0.00 | 12.13 | 11.22 |
| E4.cost001 | 0.00 | 23.11 | 22.40 | 0.00 | 0.00 | 17.92 | 16.74 |
| E4.cost0001 | 0.00 | 13.54 | 13.54 | 0.00 | 0.00 | 9.63 | 8.87 |

Table 7
Results for problems with 12 nodes

| Problem name | Original problem | | | Concentrated problem | | | |
|---|---|---|---|---|---|---|---|
| | Hgap | Dgap | CPU | Cgap | CHgap | CDgap | CCPU |
| H12 | 7.73 | 4.66 | 34:36.80 | 3.91 | 3.52 | 5.25 | 04:39.60 |
| H12.cap06 | 0.00 | 11.69 | 57:51.05 | 0.00 | 0.00 | 11.47 | 13:00.18 |
| H12.cap08 | 5.13 | 13.26 | 42:00.54 | 4.17 | 0.75 | 11.99 | 02:51.53 |
| H12.cost01 | 5.47 | 7.14 | 37:29.63 | 0.00 | 5.47 | 6.08 | 10:41.54 |
| H12.cost001 | 2.94 | 7.42 | 47:53.94 | 0.00 | 2.94 | 6.20 | 25:24.46 |

optimal locations for hubs in the concentration set, concentrated problems were solved two times faster in the average. For problem H12.cap06, optimality is proved in more than 57 min for the original version and in 13 min for the concentrated version.

For other instances, concentration did not improve the best upper bound found solving the original problems. Still, for some instances, this analysis helped to see when the heuristic failed to find good upper bounds and in what part (location or assignment) of the problem it actually failed.

Table 8
Results for problems with 17 nodes

| Problem name | Original problem | | | Concentrated problem | | | | |
|---|---|---|---|---|---|---|---|---|
| | Hgap | Dgap | Fgap | Cgap | CHgap | CDgap | CFgap | CCPU |
| H17 | 1.90 | 12.85 | 0.35 | 1.86 | 0.00 | 11.26 | 0.00 | 17:59.50 |
| H17.cap08 | 2.24 | 22.87 | 15.21 | 0.21 | 2.03 | 22.86 | 6.68 | |
| H17.cost01 | 2.37 | 30.61 | 4.06 | 2.32 | 0.00 | 21.56 | 0.00 | 04:02.32 |
| H17.cost001 | 0.00 | 52.03 | 15.64 | 0.00 | 0.00 | 27.91 | 0.00 | 08:34.55 |
| H17.cost0001 | 0.00 | 57.06 | 18.13 | 0.00 | 0.00 | 29.05 | 0.00 | 06:34.22 |

Table 9
Results for problems with 49 nodes

| Problem name | Original problem | | | Concentrated problem | | | |
|---|---|---|---|---|---|---|---|
| | Hgap | Dgap | Fgap | Cgap | CHgap | CDgap | CFgap |
| H49 | 0.00 | 1.96 | 1.87 | 0.00 | 0.00 | 1.62 | 1.60 |
| H49.cap06 | 0.00 | 3.11 | 3.05 | 0.00 | 0.00 | 2.69 | 2.65 |
| H49.cap08 | 0.00 | 2.14 | 2.06 | 0.00 | 0.00 | 1.77 | 1.69 |
| H49.cost01 | 0.00 | 5.13 | 4.93 | 0.00 | 0.00 | 4.19 | 4.09 |
| H49.cost001 | 0.00 | 7.06 | 6.91 | 0.00 | 0.00 | 5.88 | 5.83 |
| H49.cost0001 | 0.00 | 6.40 | 6.16 | 0.00 | 0.00 | 5.26 | 5.15 |

## Acknowledgements

The authors wish to thank Prof. Martine Labbé for her help and suggestions and anonymous referees for their valuable comments.

## References

[1] Klincewicz JG. Hub location in backbone tributary network design: a review. Location Science 1998;6:307–35.

[2] Yuan D. An annotated bibliography in communication network design and routing. In: Optimization models and methods for communication network design and routing. PhD thesis, Department of Mathematics, Linköping University, Sweden, 2001.

[3] Campbell JF, Ernst AT, Krishnamoorthy M. Hub location problems. In: Drezner Z, Hamacher HW., editors. Facility location: applications and theory. Berlin: Springer; 2002. pp. 373–407.

[4] Yaman H. Concentrator location in telecommunication networks. PhD thesis, Université Libre de Bruxelles, 2002. Available at http://smg.ulb.ac.be/.

[5] Boland N, Ebery J, Ernst A, Krishnamoorthy M. The capacitated multiple allocation hub location problem: formulation and algorithms. European Journal of Operational Research 2000;120:614–31.

[6] Carello G, Della Croce F, Ghirardi M, Tadei R. Solving the hub location problem in telecommunication network design: a local search approach. Networks, 2004, to appear.

[7] Ernst A, Krishnamoorthy M. Solution algorithms for the capacitated single allocation hub location problem. Annals of Operations Research 1999;86:141–59.

[8] Labbé M, Yaman H, Gourdin E. A branch and cut algorithm for the hub location problems with single assignment. Mathematical Programming, 2004, to appear.
[9] Chamberland S, Sanso B, Marcotte O. Topological design of two-level telecommunication networks with modular switches. Operations Research 2000;48:745–60.
[10] Chamberland S, Sansò B. On the design problem of multitechnology networks. INFORMS Journal on Computing 2001;13:245–56.
[11] Chamberland S, Sansò B. Topological expansion of multiple-ring metropolitan area networks. Networks 2000;36:210–24.
[12] Rosing KE, ReVelle CS. Heuristic concentration: two stage solution construction. European Journal of Operational Research 1997;97:75–86.
[13] Rosing KE, Hodgson MJ. Heuristic concentration for the *p*-median: an example demonstrating how and why it works. Computers and Operations Research 2002;29:1317–30.
[14] Skorin-Kapov D, Skorin-Kapov J, O'Kelly M. Tight linear programming relaxations of uncapacitated *p*-hub median problem. European Journal of Operational Research 1996;94:582–93.
[15] Dantzig GB. On the significance of solving linear programming problems with some integer variables. Document, The Rand Corporation, 1959. p. 1486.
[16] Jűnger M, Thienel S. The ABACUS system for branch-and-cut-and-price algorithms in integer programming and combinatorial optimization. Software Practice and Experience 2000;30:1325–52.
[17] Labbé M, Yaman H. Projecting the flow variables for hub location problems. Networks, 2004, to appear.
[18] Glover F. Tabu search. Part I. Orsa Journal on Computing 1989;1:190–206.
[19] Glover F. Tabu search. Part II. Orsa Journal on Computing 1990;2:4–32.
[20] Martello S, Toth P. Knapsack problems—algorithms and computer implementations. New York: Wiley; 1990.
[21] Beasley JE. OR-Library: distributing test problems by electronic mail. Journal of the Operational Research Society 1990;41:1069–72 http://mscmga.ms.ic.ac.uk/info.htmlhttp://mscmga.ms.ic.ac.uk/info.html.