A Hub Location Problem with Fully Interconnected Backbone and Access Networks

Tommy Thomadsen
Informatics and Mathematical Modelling
Technical University of Denmark
2800 Kgs. Lyngby
Denmark
tt@imm.dtu.dk

Jesper Larsen[*]
Informatics and Mathematical Modelling
Technical University of Denmark
2800 Kgs. Lyngby
Denmark
jla@imm.dtu.dk

July 21, 2005

Abstract

This paper considers the design of two-layered fully interconnected networks. A two-layered network consists of clusters of nodes, each defining an access network and a backbone network. We consider the integrated problem of determining the access networks and the backbone network simultaneously. A mathematical formulation is presented, but as the linear

[*]Corresponding author

programming relaxation of the mathematical formulation is weak, a formulation based on the set partitioning model and column generation approach is also developed. The column generation subproblems are solved by solving a series of quadratic knapsack problems. We obtain superior bounds using the column generation approach than with the linear programming relaxation. The column generation method is therefore developed into an exact approach using the Branch-and-Price framework. With this approach we are able to solve problems consisting of up to 25 nodes in reasonable time. Given the difficulty of the problem, the results are encouraging.

Keywords: Hierarchical networks, Fully interconnected networks, Hub location, Branch-and-Price.

1 Introduction

Wired communication networks are usually organized in a hierarchal structure based on two or more layers. This structure has proven robust to changing demands and upgrades and is seen as the right compromise between cost and redundancy. We consider networks with two layers, but the analysis is generalizable to more layers.

The two layers in the network are denoted the backbone network and the access networks. The backbone network connects disjoint clusters of nodes, each comprising an access network. The node connecting an access network to the backbone network is called a hub. An example is shown in Figure 1. Here hubs are shown as squares, thin lines are connections in an access network, while thick lines represent connections in the backbone network. The dashed lines mark the clusters, each representing an access network.

When designing such hierarchal or layered networks, a number of interrelated questions have to be resolved: Which nodes should be hubs, how should we define the clusters, and which interconnections should we allow. Since theses problems are interrelated, they should be addressed by an integrated approach in order to ensure an optimal solution.

This paper considers the joint selection of hubs and clustering of nodes of two-layered networks. In each of the layers, we assume the networks to be fully interconnected. This corresponds to Figure 1, where the access networks and the backbone network are fully interconnected, that is, for two nodes in the backbone or the same cluster, there is a link. The objective is to minimize link costs consist-

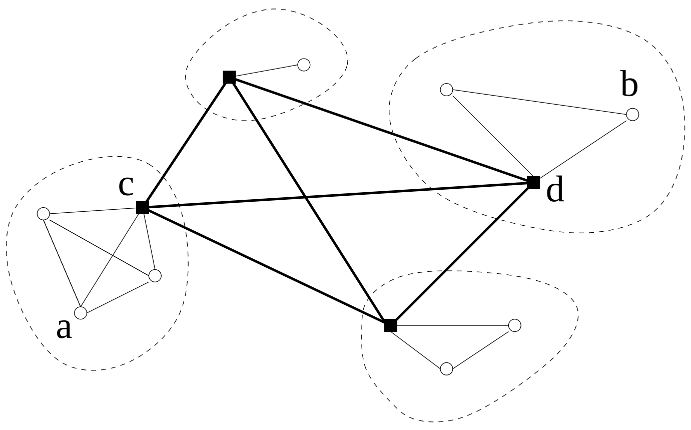

Figure 1: Example of a two-layered network. All access networks and the backbone network are fully interconnected.

ing of a cost for each link that needs to be established. This problem is denoted the Fully Interconnected Network Design Problem (FINDP).

Note that communication between two nodes in the same access network (eg. a and c in Figure 1) is handled within the access network. On the other hand, communication between two nodes not in the same access network has to be routed via the backbone network. So communication from a to b is routed via c and d in the backbone network in Figure 1.

Klincewizc [1] surveys work on two-layered network design. A significant part of the work reviewed focus on fully interconnected backbone networks. In particular [2] and [3] consider the design of fully interconnected backbone network and a star network in the access networks. Using the fully interconnected network on both layers is an open problem according to the survey paper of Klincewizc.

The FINDP is a fixed charge problem in which a cost is incurred for establishing links. Several papers deals with fixed charge network design problems, e.g. [4] and the very recent [5]. Other papers consider two layered network design problems where it is assumed that the clusters are predetermined. Such papers are surveyed in [6]. Thomadsen and Stidsen [7] address such a problem, where the backbone is a fixed charge network design problem.

The FINDP could be used in setting up a computation cluster of computers. Instead of affording a direct connection between all computers, they are divided into two layers, with a backbone network ensuring fast access beyond the access network of a computer.

The paper is organized as follows. Section 2 presents a mathematical formulation for the FINDP problem. As the linear relaxation of the mathematical formulation is weak, we develop a column generation approach in Section 3, and present the Branch-and-Price framework to obtain integral solutions in Section 4. Experimental results are presented in Section 5, and finally the conclusions are given in Section 6.

2 Network Design

First we formulate a mathematical formulation for the FINDP. Consider the graph $G = (V, E)$, where V is the set of nodes and E is the set of undirected links. Let c_{ij} denote the cost of link ij in E.

We consider a problem where there are lower and upper bounds on the number of clusters and the number of nodes in the clusters. Let b_{min} and b_{max} be a lower bound and an upper bound on the number of clusters, respectively. Furthermore v_{min} is the lower bound on the number of nodes in any cluster and v_{max} the corresponding upper bound.

We initially define three sets of variables x_{ij}, y_{ij} for i, j in V, $i < j$, and h_i for i in V. The variable h_i is 1 if node i is a hub for a cluster, and 0 otherwise. The variable x_{ij} is 1 if ij is a link in the access and 0 otherwise, and correspondingly y_{ij} is 1 if ij is a link in the backbone network and 0 otherwise. Initially we get:

$$\begin{aligned}
\min \quad & \sum_{ij \in E} c_{ij} x_{ij} + \sum_{ij \in E} c_{ij} y_{ij} & & (1) \\
\text{s.t.} \quad & y_{ij} + x_{ij} \le 1 & \forall ij \in E \quad & (2) \\
& h_i + h_j + x_{ij} \le 2 & \forall ij \in E \quad & (3) \\
& y_{ij} \le h_k & \forall ij \in E, k \in \{i, j\} \quad & (4) \\
& x_{ik} + x_{jk} \le x_{ij} + 1 & \forall i, j, k \in V, i < j, k \ne i, k \ne j \quad & (5) \\
& y_{ik} + y_{jk} \le y_{ij} + 1 & \forall i, j, k \in V, i < j, k \ne i, k \ne j \quad & (6) \\
& x_{ij}, y_{ij}, h_i \text{ binary} & & (7)
\end{aligned}$$

The objective function (1) is the sum of costs in the access networks and the backbone network. Inequality (2) ensures that a link cannot be used both in the backbone network and the access network at the same time. Next inequality (3) states that if both nodes i and j are hubs then there can be no access network link between these nodes. Now inequality (4) says that if a node k is not a hub, then a backbone network link cannot be incident to k. Finally (5) and (6) ensure that the access network, respectively the backbone network are fully interconnected.

The formulation can be strengthened by adding

$$h_i + h_j \leq y_{ij} + 1 \qquad \forall ij \in E \tag{8}$$

where (2) and (8) dominates (3), and (4) together with (8) dominates (6).

This initial formulation does, however, not ensure that each cluster contains a hub. Therefore, we introduce the variable w_{ij} for each ordered pair (i, j) where $i, j \in V, i \neq j$. If w_{ij} is 1 node i is a hub and node j is connected to i and 0 otherwise, i.e.,

$$w_{ij} = h_i x_{ij}.$$

Described by linear constraints we get:

$$\begin{aligned}
& w_{ij} \leq h_i && \forall i, j \in V, i \neq j && (9)\\
& w_{ij} \leq x_{ij} && \forall i, j \in V, i \neq j && (10)\\
& h_i + x_{ij} \leq 1 + w_{ij} && \forall i, j \in V, i \neq j && (11)\\
& h_j + \sum_{i, i \neq j} w_{ij} = 1 && \forall ij \in E && (12)\\
& w_{ij} \text{ binary} && && (13)
\end{aligned}$$

The constraints (9) and (10) force w_{ij} to 0 if node i is not a hub or if there is no link in the access network between i and j. Inequality (11) set w_{ij} to 1 if node i is a hub *and* there is a link between i and j in the network. Then (12) either forces node i to be a hub *or* to be connected to exactly one hub j, and as a consequence, each cluster contains a hub.

Finally we describe bounds on the number of clusters (14) and nodes in each cluster (15):

$$b_{\min} \leq \sum_i h_i \leq b_{\max} \tag{14}$$

$$v_{\min} - 1 \leq \sum_j x_{ij} \leq v_{\max} - 1 \qquad \forall i \in V \tag{15}$$

When we refer to the formulation FINDP for the two-layered fully interconnected network problem we refer to the formulation defined by (1)-(2), (4)-(5) and (7)-(15). The linear programming relaxation is obtained by replacing (7) and (13) with a non-negativity constraint on all variables. This formulation is denoted LP-FINDP.

3 Decomposition and Column Generation

LP-FINDP generally provides a poor bound on the optimal value of the FINDP. This is largely due to the weak LP relaxation and the inherent symmetry in the formulation.

In order to obtain a better formulation of the FINDP problem let $\mathcal{C}$ be the set of all clusters with a hub selected, and let $\mathcal{B}$ be the set of all backbone networks, where each backbone network is a set of hubs. Let a_i^c be 1 if node i is in cluster c and 0 otherwise. Furthermore let s_i^c be 1 if node i is a hub in cluster c and 0 otherwise. For the backbone, let s_i^b be 1 if node i is in backbone, and 0 otherwise. For a cluster c in $\mathcal{C}$ let c_c be the cost of the cluster, i.e., the sum of all cost on the links in c, that is,

$$c_c = \sum_{i,j,i<j} a_i^c a_j^c c_{ij}$$

and correspondingly let c_b be the cost of the backbone b in $\mathcal{B}$, so:

$$c_b = \sum_{i,j,i<j} s_i^b s_j^b c_{ij}$$

Now we can formulate an alternative formulation of the FINDP problem. Variables are u_c which are 1 if cluster c in $\mathcal{C}$ is selected, 0 otherwise and v_b which are 1 if backbone b in $\mathcal{B}$ is selected, 0 otherwise.

$$\min \quad \sum_{c\in\mathcal{C}} c_c u_c + \sum_{b\in\mathcal{B}} c_b v_b \tag{16}$$

$$\text{s.t.} \quad \sum_{c\in\mathcal{C}} a_i^c u_c = 1 \quad i \in V \tag{17}$$

$$-\sum_{c\in\mathcal{C}} s_i^c u_c + \sum_{b\in\mathcal{B}} s_i^b v_b = 0 \quad i \in V \tag{18}$$

$$u_c, v_b \text{ binary} \tag{19}$$

Here, (16) is the objective function minimizing the accumulated cost of the access networks and the backbone network. The equalities (17) ensure that all nodes belong to exactly one access network, while the constraints (18) ensure that hub nodes are in the backbone network.

Consider a small example of 4 nodes that should be divided into exactly 2 clusters of precisely 2 nodes each. For this small instance, it is possible to enumerate all possibilities. Let us denote the nodes a, b, c and d. Figure 2 shows the coefficient matrix and right-hand sides for this small problem. Each possible cluster consists of 2 nodes, so for each possible cluster there are two hub candidates. Therefore each feasible cluster results in two different columns representing the cluster but with different hubs (represented by the -1 coefficient in the lower half). Then follows a column for each possible backbone solution. If e.g. we choose the two clusters (a, b) and (c, d), then depending on the choice of hub, the matching column in the rightmost part will enforce the right cost of the backbone network.

$$
\begin{array}{llllllllllllllllllll}
a: & 1 & 1 & 1 & 1 & 1 & 1 & & & & & & & & & & & & & =1 \\
b: & 1 & 1 & & & & & 1 & 1 & 1 & 1 & & & & & & & & & =1 \\
c: & & & 1 & 1 & & & 1 & 1 & & & 1 & 1 & & & & & & & =1 \\
d: & & & & & 1 & 1 & & & 1 & 1 & 1 & 1 & & & & & & & =1 \\
\hline
a: & -1 & & -1 & & -1 & & & & & & & & 1 & 1 & 1 & & & & =0 \\
b: & & -1 & & & & & -1 & & -1 & & & & 1 & & & 1 & 1 & & =0 \\
c: & & & & -1 & & & & -1 & & & -1 & & & 1 & & 1 & & 1 & =0 \\
d: & & & & & & -1 & & & & -1 & & -1 & & & 1 & & 1 & 1 & =0
\end{array}
$$

Figure 2: Coefficient matrix for a small example consisting of 4 nodes. Blanks in the matrix represent entries of value 0.

It is possible to enumerate all columns for very small instances. However, for a problem with $K = 20, v_{\min} = b_{\min} = 6$ and $v_{\max} = b_{\max} = 8$ we get approximately 2.1 million columns. Enumerating all these columns will in practice be computational inefficient. In order to avoid generating all clusters and backbones a priori, we use the iterative method of column generation. The clusters and backbone networks are generated as they are needed. For each constraint (17) we associate a dual variable α_i and for each constraint (18) we associate the dual variable β_i.

Preliminary results show significantly better bounds by adding the following strengthening constraint to the formulation:

$$\sum_{b \in \mathcal{B}} v_b = 1 \tag{20}$$

The constraint (20) has an associated dual variable γ.

Most often column generation involves a master problem and a subproblem. The master problem solves an LP relaxation and delivers dual variables to the subproblem where new variables are computed in case a better one exists. The linear programming relaxation of the FINDP (defined by (16) to (20)) problem is denoted CG-FINDP and is obtained by replacing (19) with non-integrality constraints. For the CG-FINDP the master problem is associated with *two* subproblems; one for generating clusters and one for generating backbone configurations, see Figure 3.

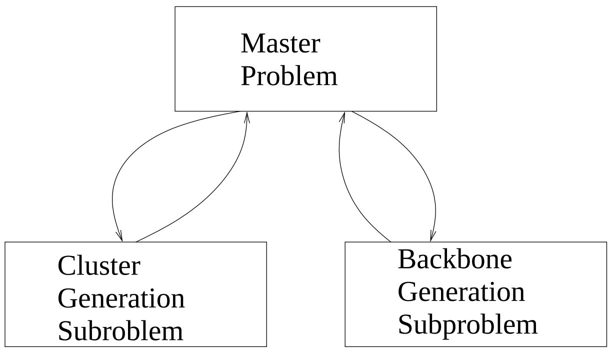

Figure 3: Overview of the column generation algorithm.

3.1 The Backbone generation subproblem

For an optimal solution the reduced cost of a backbone column is:

$$\sum_{ij \in E} c_{ij} y_{ij} - \sum_{i \in V} \beta_i s_i - \gamma \tag{21}$$

where y_{ij} and s_i are binary variables representing whether the link ij respectively the node i is part of the backbone network.

In order to find a new column to enter the basis, we seek the column with the most negative reduced cost. By multiplying the reduced cost with -1 the objective function is:

$$\max \sum_{i \in V} \beta_i s_i + \gamma - \sum_{ij \in E} c_{ij} y_{ij} \tag{22}$$

The constraints for obtaining a feasible backbone network are:

$$y_{ij} \leq s_i \qquad ij \in E \tag{23}$$

$$y_{ij} \leq s_j \qquad ij \in E \tag{24}$$

$$s_i + s_j \leq y_{ij} + 1 \qquad ij \in E \tag{25}$$

$$b_{min} \leq \sum_{i \in V} s_i \leq b_{max} \tag{26}$$

$$s_i, y_{ij} \text{ binary} \tag{27}$$

The link ij can only be selected for the backbone network if both node i and j are part of it. This is ensured by the constraints (23) and (24). Furthermore, inequalities (25) enforces link ij to be part of the backbone network if nodes i and j are selected. Finally, constraint (26) ensures that the bounds on the number of clusters (access networks) are enforced. The pricing problem for the backbone generation subproblem is defined by (22)-(27). A solution approach to this formulation is described in Section 3.3

3.2 The Cluster generation subproblem

The definition of the cluster generation subproblem is parallel to the approach for the backbone network. Let a_i and x_{ij} be 1 if the node i respectively the link ij is in the cluster, and 0 otherwise. Furthermore the variable s_i is 1 if node i is a hub, and 0 otherwise. Now the reduced cost of a cluster is:

$$\sum_{ij \in E} c_{ij} x_{ij} - \sum_{i \in V} \alpha_i a_i + \sum_{i \in V} \beta_i s_i \tag{28}$$

The cluster generation problem seeks the column with the most negative reduced cost, or equivalent, has the objective:

$$\max \sum_{i \in V} \alpha_i a_i - \sum_{i \in V} \beta_i s_i - \sum_{ij \in E} c_{ij} x_{ij} \tag{29}$$

A feasible column (defining an access network) must fulfill:

$$\sum_{i \in V} s_i = 1 \tag{30}$$

$$s_i \leq a_i \qquad i \in V \tag{31}$$

$$x_{ij} \leq a_i \qquad ij \in E \tag{32}$$

$$x_{ij} \leq a_j \qquad ij \in E \tag{33}$$

$$a_i + a_j \leq x_{ij} + 1 \qquad ij \in E \tag{34}$$

$$v_{min} \leq \sum_{i \in V} a_i \leq v_{max} \tag{35}$$

$$a_i, s_i, x_{ij} \text{ binary} \tag{36}$$

Each access network has precisely one hub, which is ensured by equation (30), and the hub has to be part of the access network, which is ensured by (31). Parallel to (23)-(25) for the backbone generation problem, (32)-(34) ensure that a link ij is selected if and only if both nodes i and j are selected. A feasible access network can only have between $v_{\min}$ and $v_{\max}$ nodes, i.e., it has to fulfill (35). Thus the pricing problem for the cluster generation is defined by (29)-(36). A solution approach is discussed in the following section.

3.3 Solving the subproblems

Instead of solving the problems for backbone and cluster generation directly, it can be observed that both of the subproblems can in fact be solved as a series of quadratic knapsack problems (QKP). For at general description of QKP see [8]. The quadratic knapsack problem seeks to maximize a quadratic objective function subject to a single capacity constraint. If we let the binary variable q_i be equal to 1 if item i is selected and 0 otherwise, and let q_{ij} be 1 if both i and j are selected and 0 otherwise. Finally let p_i be the profit of selecting item i, p_{ij} be the profit of selecting both item i **and** j. Then the QKP can be formulated as:

$$\max \quad \sum_i p_i q_i + \sum_i \sum_{j:i<j} p_{ij} q_{ij} \tag{37}$$

$$\text{s.t.} \quad q_{ij} \le q_i \qquad i, j \tag{38}$$

$$q_{ij} \le q_j \qquad i, j \tag{39}$$

$$q_i + q_j \le q_{ij} + 1 \qquad i, j \tag{40}$$

$$\sum_j w_j q_j \le C \tag{41}$$

$$q_{ij}, q_j \text{ binary} \tag{42}$$

where w_j is the weight of the j'th item and C is the capacity of the knapsack. Constraints (38), (39), and (40) ensure consistency of variables and (41) is the knapsack constraint. A solution approach to the QKP is described in [9]. The approach is based on Lagrangian relaxation and seems to be the current state-of-the-art for exact solution of the QKP. Furthermore the source code is available at the homepage of David Pisinger, see `www.diku.dk/~pisinger`. This approach and the available code is used to solve both subproblems.

Let us first consider our subproblem for the backbone network, (22)- (27). The subproblem bears some resemblance with the QKP. The nodes can be considered items with a weight of 1 and the profit equals the cost of the links in the backbone. The deviation is that both a lower and upper bound exist on the contents of the knapsack. To address this, we take the approach of adding a constant to all coefficients in the objective, such that all coefficients are non-negative. With non-negative coefficients in the objective and weights equal to 1 in the knapsack, the optimal solution to the corresponding QKP, will always fill the knapsack to capacity. Hence, the subproblem can be solved by solving a series of QKP's, one for each of the capacities $C = b_{\min}, b_{\min} + 1, \ldots, b_{\max}$. The algorithm is shown in Figure 4.

Similarly to the backbone generation problem, we add a constant to all coefficients in the objective, such that all coefficients are non-negative and solve a problem for each of $C = v_{\min}, v_{\min} + 1, \ldots, v_{\max}$. However, the cluster generation problem poses one additional complication compared to the backbone generation problem. In addition to choosing nodes, one of the nodes has to be designated as the hub node. We handle this by enforcing each of the nodes to be hub, one at a time. This corresponds to fixing each of the s_i variables to 1 in turn. However, fixing a variable to 1 cannot be done directly using the code used to solve the QKP's.

```
for $b_k = b_{\min}$ to $b_{\max}$ do
    Solve QKP for $C = b_k$
    if Solution has a positive value then
        add the cluster column to the master problem
    end if
end for
```

Figure 4: The backbone generation algorithm.

Instead, a sufficiently high value is added to the objective coefficient of the node, thus ensuring that the node is selected. As a consequence, $|V|(v_{\max} - v_{\min} + 1)$ QKP problems have to be solved. The algorithm is shown in Figure 5.

```
for $i \in V$ do
    for $v_k = v_{\min}$ to $v_{\max}$ do
        Solve QKP for $C = v_k$ and $s_i$ forced to $1$
        if Solution has a positive value then
            add the cluster column to the master problem
        end if
    end for
end for
```

Figure 5: The cluster generation algorithm.

The approach described in [9] also contains a greedy heuristic for the QKP. In our generation of subproblems, we run this heuristic on the full series of subproblems before running the exact approach. The exact approach is only used if no columns can be obtained by the heuristic. The QKP is also used as a subproblem in a column generation approach in [10]. Furthermore paper G in [11] uses a similar approach to solve a related two layered network design problem.

3.4 Initialization

To initialize the column generation, a number of “dummy” columns for the clustering and the backbone part are generated. The dummy columns for the clustering part consists of one column per node. Each column contains one node with no designated hub, that is, the column contains a single 1 for the i’th row ($a_i^c = 1$).

For the backbone part we also generate one column per node i. These columns have a 1 corresponding to the i'th node being a hub ($s_i^b = 1$), and all remaining coefficients are 0. In order to satisfy (20), an additional "dummy" column is added. It does not contain any nodes but has a coefficient 1 for the constraint (20). All these dummy columns are added to ensure a feasible LP upon branching and they are assigned a value sufficiently high in order to force them out of the basis in the optimal solution.

4 Branch-and-Price

As the column generation method described above cannot guarantee integral solutions, it has to be embedded in the Branch-and-Bound framework. The combination of column generation and Branch-and-Bound is often denoted Branch-and-Price or IP column generation [12, 13].

In case the solution of a branch node in the Branch-and-Bound tree is not integer and cannot be fathomed, we branch. Here, we implement the Ryan-Foster branching [14]. We branch on whether node i and j are in the same cluster or not. A constraint enforcing this requirement is added to the master problem. This does, however, not guarantee integer optimality. Two clusters with the same nodes but with different hubs may be selected, each with u_c equal to $\frac{1}{2}$. Now branching on whether node i and j are in the same cluster cannot be applied, yet the solution is not integer. This is handled by branching on whether a node is a hub or not, that is, in one branch, node i is forced to be hub, and in the other branch node, i cannot be hub.

In the master problem, the choices taken by the branching strategy results in the addition of constraints. Let $B_1 \subseteq E$ be the set of "nodes are/are not in the same cluster" branches and $B_2 \subseteq V$ be the set of "node i is/is not hub" branches. We define p_b^c for $\{i, j\} = b \in B_1$ to be equal to one if i and j are in the same cluster, otherwise 0. Furthermore, recall that s_i^c is 1 if i is a hub in cluster c. So we get:

$$\sum_{c \in C} p_b^c u_c = 0/1 \quad b \in B_1 \tag{43}$$

$$\sum_{c \in C} s_i^c u_c = 0/1 \quad i \in B_2 \tag{44}$$

where (43) with the right-hand side 0 corresponds to forbid clusters containing i

and j, and a right-hand side of 1 forces a cluster to contain both i and j. Correspondingly a right-hand side of 0 in (44) means that node i is not hub, and 1 forces i to be hub.

Branching is implemented by first determining the u_c column with fractional values closest to $\frac{1}{2}$. Then, the first row covered (i.e., it has a coefficient of 1) by this column is found. Now we search for another column that has a fractional value and covers the same row. Such a column must exist due to the partitioning constraints for the clusters (17). So either:

1. The columns cover exactly the same rows. This implies that the hubs are not identical and we branch on whether the node is a hub or not.

2. The columns cover different rows. Now we determine the first row where they differ and branch on whether nodes are in the same cluster or not.

Referring back to the example in Figure 2, consider the situation where the first two columns are picked with a value of $\frac{1}{2}$ and the two remaining nodes where covered by a single column with value 1. The branching approach will then detect that both columns define the same cluster (they cover the exact same rows) and therefore branching will force the node represented by the first row (node a) to either be or not be a hub. If instead u_c is equal to $\frac{1}{2}$ for the first and the third column, the test reveals that the two columns define different clusters and branching will force a and b to be in the same cluster or not.

In the branching tree, a depth first strategy is applied. This enables use of "warm start" in the LP relaxation of the master problem with the previous solution. Having two candidates for branching on the same depth, we choose the one that fixes the right hand side values to 1 in (43) and (44).

The branch constraints (43) and (44) lead to dual variables which needs to be incorporated into the subproblems. Taking these dual variables into account in the subproblem is sufficient, i.e., it is not necessary to force the subproblem to generate columns feasible with respect to a given set of branches. The dual variables δ_b and ϵ_i corresponding to (43) and (44) are only added to the calculation of the reduced cost for a cluster column. No modification of the backbone columns are necessary.

We now modify (28) to reflect that the reduced cost of the columns should include the dual variables of the branch constraints:

$$\sum_{ij\in E} c_{ij}x_{ij} - \sum_{i\in V} \alpha_i a_i + \sum_{i\in V} \beta_i s_i - \sum_{b\in B_1} \delta_b p_b - \sum_{i\in B_2} \epsilon_i s_i \tag{45}$$

where p_b is 1 for $b = (i, j)$, if i and j are in the same cluster.

Therefore, for the cluster generation problem, the objective of the pricing problem (29) is modified to:

$$\max \sum_{i \in V} \alpha_i a_i - \sum_{i \in V} \beta_i s_i - \sum_{ij \in E} c_{ij} x_{ij} + \sum_{b \in B_1} p_b \delta_b + \sum_{i \in B_2} s_i \epsilon_i \tag{46}$$

By noting that $p_b = p_{\{i,j\}} = a_i a_j = x_{ij}$ and rewriting, we obtain:

$$\max \sum_{i \in V} \alpha_i a_i + \sum_{i \in V} (\epsilon_i - \beta_i) s_i + \sum_{ij \in E} (\delta_{ij} - c_{ij}) x_{ij} \tag{47}$$

Thus including the additional dual variables is only a matter of modifing the constants of the objective, and hence can easily be included.

5 Experimental Results

We have tested the two bounds and the Branch-and-Price approach on generated instances with n nodes for $n = 10, 15, 20,$ and 25. All the graphs are fully connected and two types of instances have been generated. Euclidean instances where the link costs are proportional to the Euclidean distances between the endpoints which have been randomly located in the unit square, and random instances, where the link costs are randomly selected using a uniform distribution. Furthermore $b_{\min}$ and $v_{\min}$ are set to $\lfloor \sqrt{n} \rfloor - B_d$, and $b_{\max}$ and $v_{\max}$ are set to $\lfloor \sqrt{n} \rfloor + B_d$. Here we have tested each instance with B_d equal to $1, 2,$ and 3.

First we have tested the column generation scheme (CG-FINDP) against the LP relaxation of the FINDP (LP-FINDP) to see which approach produces the tightest bounds. The results are shown in Table 1 and Table 2. In the tables, "Gap" is the gap to the known optimal solution, "Iter" denotes the number of iterations, i.e., the number of calls to the subproblems in the column generation algorithm, and "Cols" identifies the number of columns generated.

For both types of graphs, it is evident that the column generation approach produces bounds superior to the LP relaxation. On the Euclidean instances the gaps are between 27% and 76% for the LP relaxation, which will make it difficult to obtain an efficient exact approach based on an LP relaxation. In contrast the bound for CG-FINDP are between 1.88% and 28%, and for the random instances, the bounds are even better. Here the largest deviation from the optimal solution is below 10% for the CG-FINDP. For the LP relaxation, the gaps have increased and

Table 1: The LP relaxation of the FINDP formulation vs. the column generation approach for a lower bound on the Euclidean instances.

Problem		LP-FINDP		CG-FINDP			
n	B_d	Seconds	Gap (%)	Seconds	Gap (%)	Iter	Cols
10	1	0.15	52.6	0.47	14.9	8	159
10	2	0.12	61.1	0.88	3.9	9	219
10	3	0.15	69.9	1.05	13.1	9	237
15	1	1.15	31.3	1.84	8.8	17	388
15	2	1.31	60.0	2.32	1.9	14	408
15	3	0.38	73.3	3.25	17.2	13	450
20	1	1.69	57.0	3.62	27.7	21	545
20	2	1.16	65.6	6.14	11.1	17	718
20	3	1.36	76.0	8.24	13.8	14	747
25	1	6.39	27.1	13.30	14.9	24	846
25	2	4.73	56.5	14.58	20.2	21	1272
25	3	5.07	70.8	19.06	15.6	19	1224

are in the interval between 48% and 89%. The cost of better bounds is a modest increase in running times. Note that both the number of columns and iterations needed is low. We never need to generate more than 1400 columns and run 27 iterations.

Note that gaps are smaller for the Euclidean instances than for the corresponding random instances wrt. the LP-FINDP bound. This is very much in line with the results for comparable problems like the uncapacitated facility location. However, for the column generation approach tighter bounds are obtained on the random instances.

Based on the results above we have only tested an exact approach based on the column generation bound. The results of the tests are presented in Table 3 and Table 4. Here the column "BB" displays the number of Branch-and-Bound nodes needed to find the optimal solution, "Cols" and "Iter" denote the number of columns respectively the number of iterations in the column generation process that is needed. Finally, the remaining 4 columns presents the total running time, and then a breakdown into the Master Problem ("MP"), the exact pricing algorithm ("SP opt") and the heuristic pricing algorithm ("SP heu").

The results clearly show that the randomly generated instances are easier to

Table 2: The LP relaxation of the FINDP formulation vs. the column generation approach for a lower bound on the randomly generated instances.

Problem		LP-FINDP		CG-FINDP			
n	B_d	Seconds	Gap (%)	Seconds	Gap (%)	Iter	Cols
10	1	0.16	66.5	0.41	5.7	9	163
10	2	0.14	74.4	1.26	4.5	9	194
10	3	0.13	78.5	0.82	4.7	8	222
15	1	1.74	49.6	1.15	5.3	12	294
15	2	1.29	76.0	3.64	1.8	13	397
15	3	0.47	80.4	5.34	5.9	14	487
20	1	1.81	56.8	5.49	1.5	21	599
20	2	1.09	80.2	12.15	9.7	16	636
20	3	1.12	88.9	14.46	7.1	19	697
25	1	6.61	48.5	14.97	6.1	25	763
25	2	4.97	76.2	33.24	4.0	27	1220
25	3	4.64	83.7	36.97	1.8	27	1367

solve than the Euclidean instances. Obviously, the tighter gap plays an important role. Except for the three Euclidean instances with 25 nodes, all running times can be seen as reasonable. It is worth noting that a large fraction of the time spent by our algorithm is spent in the exact SP. In the breakdown of the time usage, the time spent in the exact SP algorithm always accounts for the vast majority of the running time. Most of the problems, including all randomly generated instances, are solved generating only a few thousand columns.

Instead of solving the two subproblems by sequentially solving a series of QKPs, the formulations of the two subproblems ((22)-(27) and (29)-(36)) can be solved directly by a MIP solver. Since the heuristic SP produces most columns, the call to the exact SP procedure is often just to check that all columns have been generated. Thus the single call to the MIP solver can replace the exact SP procedure, which involves solving a series of QKPs optimally. Initial computational experiments support that this speeds up the procedure.

Table 3: Results for the Branch-and-Price for the FINDP on the Euclidean instances

Problem		BB	Cols	Iter	Seconds			
n	B_d				Total	MP	SP opt	SP heu
10	1	54	441	213	25	0	17	6
10	2	34	540	152	27	0	18	8
10	3	6	346	41	8	0	4	3
15	1	245	1859	1172	391	9	326	52
15	2	78	1201	431	201	2	166	31
15	3	66	1310	398	256	2	207	42
20	1	1621	7226	6228	4796	302	4091	369
20	2	120	2481	679	696	9	610	68
20	3	1006	6971	4324	7551	221	6655	636
25	1	4568	19137	19375	42619	5626	35279	1463
25	2	45839	55692	115849	671346	248690	402661	14474
25	3	4922	18658	16978	71056	4948	62838	2984

6 Conclusion

The contribution of this paper is the development of two different formulations (a mathematical formulation and one based on column generation) and an exact solution approach for a two-layered network design problem. The problem is defined by using a fully interconnected topology both for the access networks and the backbone network.

Our computational experiments are based on two sets of instances, one randomly generated and one using Euclidean distances. The results show that the bound based on column generation is superior to the LP relaxation of the mathematical formulation. The gaps are often more than a factor 10 worse on the LP relaxation. The bounds on the column generation approach are tight enough – especially on the random instances – to develop an optimal approach, even though this bound is more time consuming to compute than the LP relaxation.

The optimal method is able to solve all randomly generated instances within one hour. The bounds on the Euclidean instances are worse than for the randomly generated instances, which is also reflected in the running times. For the Euclidean

Table 4: Results for the Branch-and-Price for the FINDP on the randomly generated instances

Problem		BB	Cols	Iter	Seconds			
n	B_d				Total	MP	SP opt	SP heu
10	1	4	179	14	1	0	1	0
10	2	2	219	8	2	0	1	0
10	3	2	247	12	3	0	2	1
15	1	15	551	123	40	0	34	5
15	2	54	917	269	177	1	152	19
15	3	120	1547	634	575	4	495	69
20	1	34	1168	215	179	2	169	13
20	2	150	2061	787	1197	10	1093	79
20	3	262	3248	1565	3151	31	2867	231
25	1	45	1697	475	944	6	885	36
25	2	77	2453	510	1720	9	1609	65
25	3	42	2281	367	1565	6	1455	64

instances 5 out of the 12 instances cannot be solved within one hour – one instance takes almost 8 days to solve. It is noteworthy that most of our problems are solved generating only a few thousand columns.

We believe that further improvements can be obtained by proving optimality of the subproblems solving the pricing problems directly in a MIP solver instead of solving a series of QKP's. Furthermore the running times on especially the Euclidean instances suggest research in heuristics. One possible approach is to base them on the existing optimal method. Feasible solutions obtained by such heuristics can be used to speed up the Branch-and-Price algorithm.

References

[1] Klincewicz J. Hub location in backbone/tributary network design: a review. Location Science 1998;6:307–355.

[2] Ernst A, Krishnamoorthy M. Efficient algorithms for the uncapacitated single allocation p-hub median problem. Location Science 1996;4(3):139–154.

[3] Skorin-Kapov D, Skorin-Kapov J, O'Kelly M. Tight linear programming relaxations of uncapacitated p-hub median problems. European Journal of Operational Research 1996;94(3):582–593.

[4] Crainic T, Frangioni A, Gendron B. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. Discrete Applied Mathematics 2001;112(1-3):73–99.

[5] Costa A. A survey on benders decomposition applied to fixed-charge network design problems. Computers and Operations Research 2005; 32(6):1429–1450.

[6] Feresmans C, Labbe M, Laporte G. Generalized network design problems. European Journal of Operational Research 2003;148(1):1–13.

[7] Thomadsen T, Stidsen T. The generalized fixed-charge network design prolem. Computers and Operations Research 2005;To appear.

[8] Kellerer H, Pferschy U, Pisinger D. Knapsack Problems. Springer, 2004.

[9] Caprara A, Pisinger D, Toth P. Exact solution of the quadratic knapsack problem. INFORMS Journal on Computing 1999;11:125–137.

[10] Johnson E, Mehrotra A, Nemhauser G. Min-cut clustering. Mathematical Programming 1993;62:133–151.

[11] Thomadsen T. Hierarchical Network Design. PhD thesis, Informatics and Mathematical Modelling, Technical University of Denmark, 2005.

[12] Barnhart C, Johnson E, Nemhauser G, Savelsbergh M, Vance P. Branch-and-price: column generation for solving huge integer programs. Operations Research 1998;46(3):316–329.

[13] Vanderbeck F, Wolsey L. An exact algorithm for ip column generation. Operations Research Letters 1996;19(4):151–159.

[14] Ryan D, Foster B. An integer programming approach to scheduling. In: Wren A, editor. Computer Scheduling of Public Transport Urban Passenger Vehcile and Crew Scheduling. Amsterdam: North Holland, 1981. p. 269–280.