Hierarchical Network Design Using Simulated Annealing

Tommy Thomadsen* Jens Clausen†

Informatics and Mathematical Modelling
Technical University of Denmark
DK-2800 Kongens Lyngby, Denmark

September 4, 2002

Abstract

The hierarchical network problem is the problem of finding the least cost network, with nodes divided into groups, edges connecting nodes in each groups and groups ordered in a hierarchy. The idea of hierarchical networks comes from telecommunication networks where hierarchies exist. Hierarchical networks are described and a mathematical model is proposed for a two level version of the hierarchical network problem. The problem is to determine which edges should connect nodes, and how demand is routed in the network. The problem is solved heuristically using simulated annealing which as a sub-algorithm uses a construction algorithm to determine edges and route the demand. Performance for different versions of the algorithm are reported in terms of runtime and quality of the solutions. The algorithm is able to find solutions of reasonable quality in approximately 1 hour for networks with 100 nodes.

Keywords: Hierarchical Networks; Network Design.

1 Introduction

Telecommunication networks consist of cables (optical or electrical wires) and switching and multiplexing equipment located at telephone switches connecting subscribers

*Email: tt@imm.dtu.dk
†Email: jc@imm.dtu.dk

and other switches. The networks are described in terms of graphs and extensive research in the topic has taken place. In this paper we focus on the design of backbone networks, specifically on choosing how switches are connected and on the capacities of the cables.

Network design problems can (roughly) be divided into two categories (I) Access network design and (II) Backbone network design [27]. Access network design assumes a centralized traffic demand and may involve hierarchical structures. These problems are closely related to facility location problems and clustering problems. Backbone network design, on the other hand, assumes a distributed (several sources and destinations) traffic demand and allows arbitrary topologies of the network to be chosen.

The two categories require rather different solution methods [27], however many models and solution methods treat both network types or borrow elements from the other category.

In backbone telecommunication networks, switches are arranged in groups and the groups of switches are connected. Not much research have been done on hierarchical backbone networks with a distributed demand pattern. Several reasons are immediate, one being that dividing the network into groups limits the choice of solutions, and hence low cost solutions may be overlooked. Additionally, routing the distributed demands while designing the topology of the network presents substantial difficulties.

The contribution of the current paper is twofold. We present and define a new type of hierarchical networks and define a minimization problem - the hierarchical network problem - based on this. The problem is a backbone network design problem with distributed demand pattern, but borrows many concepts from access network design. Secondly the paper presents a solution algorithm and computational results for this.

The remaining part of this section contains an introduction to telecommunication networks (with special focus on hierarchies) and related work. The hierarchical network problem is defined in section 2 and section 3 presents the solution strategy. The hierarchical network problem is solved in two phases. The first phase consists of a simulated annealing algorithm and is described in section 4, and section 5 describes the greedy algorithm of the second phase. Section 6 contains computational results and finally section 7 gives conclusions and suggestions for further research.

1.1 Telecommunication Networks

In telecommunication networks, different cable capacities are used in order to allow cheap, low capacity connections where sufficient, while allowing higher capacity cables to be used where required. The hierarchies contribute with an organizational element, that is, the hierarchies divide the network into areas which more easily can be handled by staff who maintains and modifies the network. Also, planning hierarchically is easier than optimizing the network as a whole.

The network is usually divided into at least three levels - a national, regional and local level. The national level connects regional areas and the regional levels connect local areas. National, regional and local areas contain a number of local switches, and subscribers are connected directly to a local switch. Regional switches are in addition always local switches.

When a subscriber dials the number of another subscriber, this is located, and a path is set up between the two subscribers. Which route to choose is programmed into the switching equipment, and thus setting up a path merely consists of reserving a fraction of the capacity for the call.

If two subscribers are connected to the same local switch or to local switches, which can connect without using regional switches, such a connection is used, and regional and national switches are not used. If the subscribers are connected to the same regional switch or regional switches, which can connect without using national switches, the call will only occupy connections to the regional switch and to the subscriber (see figure 1).

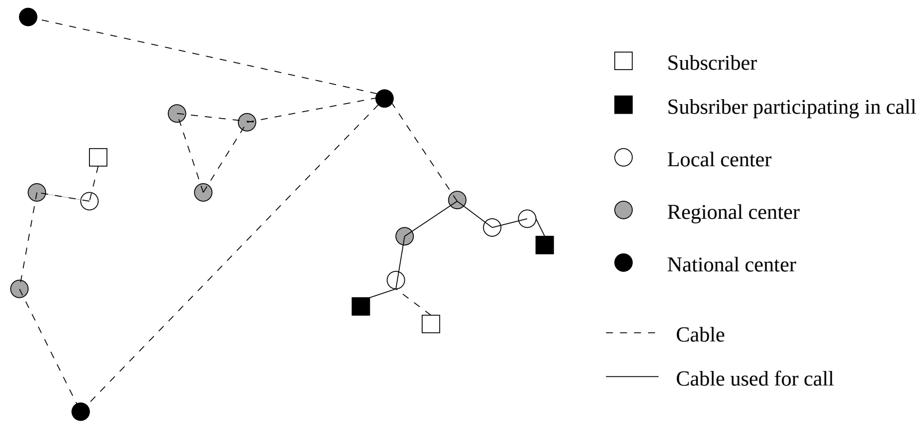

Figure 1: *Example of a regional call - some additional switches and connections are shown*

If subscribers are connected to different national switches a call will have to go through both local, regional and national switches (see figure 2). Much of the traffic in the network is in fact data transmission, but the distinction between local, regional and national transmission is still valid.

Cables and the equipment facilitating communication over the cables have different costs and capacities. The price of establishing a connection depends mainly on the

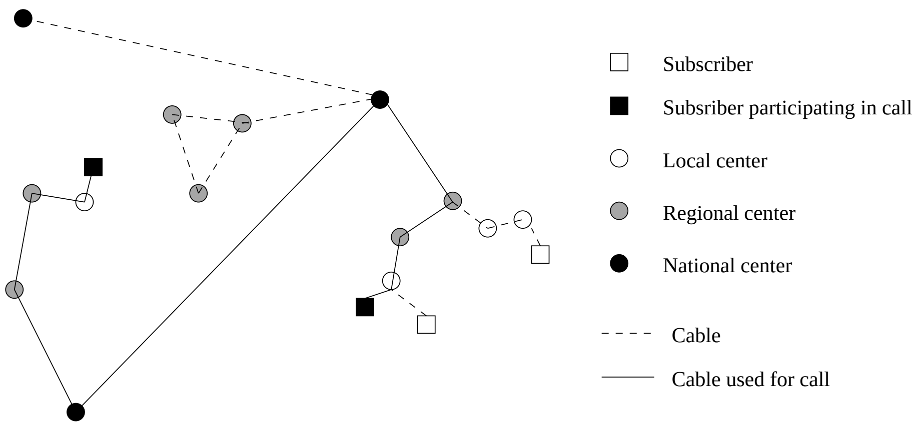

Figure 2: *Example of a national call - some additional switches and connections are shown*

cost of digging down a cable and the price of the equipment facilitating communication. The physical cable used for different capacities are usually the same.

Usually higher-level switches (e.g. national switches) use connections with high capacities, but there is no direct dependency between switch level and connection type used. Thus high capacity connections can be established between local switches, if a customer has a need for a particularly high capacity connection between two locations. The mathematical model we propose does, however, only allow cables of one capacity to be used at a given level.

Location of switches and cables, and capacity limits, are in practice historically determined and has been determined and changed as the network evolved. The network continuously evolves, new cables and switches are added to the network, and replacement of existing equipment with higher capacity equipment is done frequently.

If a network were to be built from scratch, it would probably be very different from the current network. This is so since the need for capacity has changed over time, and thus an optimal location of switches and cables at some point in time may currently not be optimal.

The location of new switches, new cables and upgrade of switches to increase capacity over cables is of major concern to the telecommunication companies. Determining how an optimal solution would look like, if starting from scratch, would contain valuable information. Also protection against failures in the network is of major concern. This is however not treated in this paper. The model described is to be considered as a first step towards the solution of the full problem.

1.2 Related Work

Early descriptions of hierarchical networks are [18] and [19]. The hierarchal networks are described in the context of the Arpanet with the purpose of reducing the overhead traffic required to maintain routing tables.

[20, 36, 23, 31, 32] solve problems on locating concentrators or facilities in a network with centralized demand pattern. Some ([36, 23]) denote the problems as multi-level or hierarchical and in fact all problems involve some sort of hierarchies.

Other papers with a centralized demand structure and hierarchies are [13] which solves an access network design problem and [24] which solves a network design hierarchical star-star problem. The multi-level capacitated minimum spanning tree problem [10] has a centralized demand structure, and a hierarchical structure emanates, because the edges have different levels representing the capacities.

Another type of hierarchical network design problem is defined in [7]. The problem is somewhat different from others: Given two primary nodes and some secondary nodes, the least cost network connecting the primary nodes with a primary path and connecting secondary nodes not on the primary path to a node on the primary path via a secondary path is to be found. The problem is extended with transshipment facilities in [6]. In [30] a new formulation is presented which is used to obtain a Lagrangean relaxation based algorithm. The algorithm is modified to handle transshipment facilities in [5]. In [34] a dynamic programming based algorithm is suggested and in [35] the problem is enhanced to allow multiple paths and the dynamic programming algorithm is modified to handle this. [14] presents a arborescence formulation for the problem, with multiple primary nodes.

[3] solves a multi-level network problem with more destination nodes but only one source node of the highest level. The destination nodes require service of different levels, and potential source nodes of lower levels require service of higher levels to be able to supply the service.

Yet another way to use multi-level (or -layer) network design is to let each layer represent a network protocol layer. This is the approach of [29]. The layers can have different properties; some layers may e.g. be able to reconfigure the routing of traffic to avoid cables which have failed. Thereby, robustness of the network can be addressed.

For a survey of network design see [21, 11] and for a description of the subproblems involved in the overall design of a network see [12, 2].

2 The Hierarchical Network Problem

The Hierarchical Network Problem (HNP) problem is described using a mathematical model in the following subsections, but initially the HNP is described in general terms.

A hierarchical network (HN) consists of nodes and edges between any pair of nodes. Edges and nodes have levels corresponding to the levels in telecommunication networks. In this initial description of the HN's, the edge level is determined by the capacity of the edge. The following terms are needed to describe HN's.

Node Level Highest level of any edge incident to the node.

Level l Groups The set of connected components of the subgraph induced by level l edges and nodes of level l or higher.

Concentrator Node Node which has higher level than the group it is in.

We note, that a group may contain only one node. Figure 3 shows a HN where edges, nodes and groups are indicated, including the levels. All nodes of level 2 are concentrators in a level 3 group and nodes of level 1 are concentrators in both a level 2 group and a level 3 group.

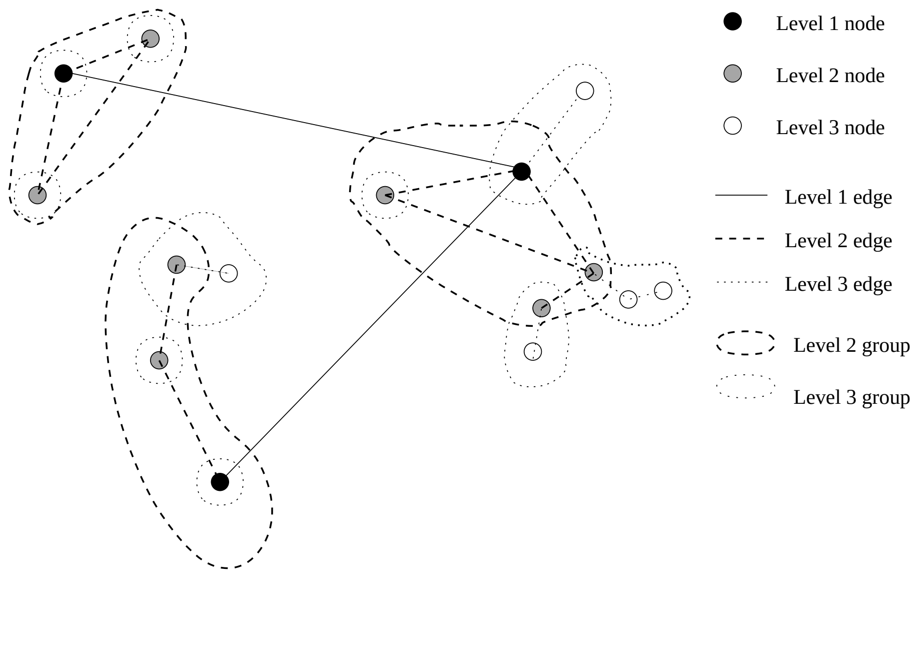

Figure 3: *Example of nodes, groups and concentrator nodes*

Given these concepts, a HN is a subset of edges, such that:

- There is at most one edge between any pair of nodes
- A group has exactly one concentrator node, except the highest level group which has none
- The network is connected

The HNP is to find the least cost HN that satisfies a demand, given by an origin-destination demand matrix for each pair of nodes. The cost of a solution is measured by two terms, the setup-cost which is the sum of costs for establishing the edges chosen, and the flow-cost which depends on how much flow (or traffic) is sent through the set of edges.

An alternative description of HN's is to define these by the way they are constructed given a set of nodes: Partition the network into disjoint sets. In each such set, select a concentrator. The concentrators makes up the highest level group. Each disjoint subset is again partitioned into disjoint subsets and concentrators selected to find the next-highest level groups. This is continued until groups contain single nodes. For each group of level l, nodes in the group are connected using level l edges. This description corresponds closely to the way the HNP is solved (see section 3).

Deciding the number of groups of each level could be part of the problem. We have chosen to request this to be specified beforehand. Doing this reduces the solution-space and gives a way of controlling the solution. It is our experience that allowing for any number of groups, the solution will usually consist of few groups. This may be due to the following: Since edges are not allowed to cross group boundaries, dividing the network into groups limits the possible selection of edges. Hence some solutions may not be feasible if groups are divided.

We will now return to the description of the mathematical model for the problem.

2.1 Index Sets

L	Set of all levels.
V	Set of all nodes.
E	Set of potential edges.
G_l	For l, $l \in L$, the set of groups of level l.

The sets are index-sets, i.e. the elements in a set are denoted by numbers between 1 and the cardinality of the set. For the set of levels, the lowest numbers describe the highest levels (as for previous examples).

There is one group of level 1, i.e. $G_1 = \{1\}$ and $G_{l-1} < G_l$ for $l \in \{2, \ldots, |L|\}$.

2.2 Data

$cs^l_{ij}, i < j$	Cost of setting up a level $l \in L$ edge between $i \in V$ and $j \in V$.
$cf^l_{ij}, i < j$	Cost per unit flow of a level $l \in L$ edge between $i \in V$ and $j \in V$.
$cap^l_{ij}, i < j$	Capacity of level $l \in L$ edge between nodes $i \in V$ and $j \in V$.
$d_{ij}, i < j$	Demand from $i \in V$ to $j \in V$.

cs^l_{ij} is named the setup-cost and cf^l_{ij} the flow-cost. cap^l_{ij} is usually equal for all edges in the same level.

We assume that the data are demand-connected (i.e. the graph with edges corresponding to positive demands is connected). Thus to route the demand, the solution must be connected as well.

Usually (e.g. for telecommunication networks), the setup-costs increase with increasing level, i.e. for $l < l'$, $cs^l_{ij} > cs^{l'}_{ij}$. Also, it seems reasonable that the flow-costs decrease with increasing level, i.e. for $l < l'$, $cf^l_{ij} < cf^{l'}_{ij}$.

2.3 Decision Variables

$x^l_{ij} \in \{0, 1\}$, $i < j$	1 if there is a level l edge between nodes $i \in V$ and $j \in V$, 0 otherwise.
$p^{mn}_i, m < n, i \neq m, i \neq n$	1 if node i is on the path chosen to satisfy the demand from $m \in V$ to $n \in V$, otherwise 0.
$f^{mnl}_{ij} \geq 0$, $i \neq j$, $m < n$	Amount of flow on the edge from $i \in V$ to $j \in V$ originating in demand from $m \in V$ to $n \in V$ on level $l \in L$ edge.
$g^{lh}_i \in \{0, 1\}, l \in L$	1 if node $i \in V$ is in level l group $h \in G_l$, 0 otherwise.
$t^{lh}_i \in \{0, 1\}, l \in \{2, \ldots, \|L\|\}$	1 if node $i \in V$ is concentrator for level l group $h \in G_l$, 0 otherwise.

Note that flows and demands are directed.

2.4 Objective Function

The cost for a given network is the total cost of setting up edges, and the sum of all flow through edges:

$$\min \sum_{i,j \in V, i<j, l \in L} cs^l_{ij} \cdot x^l_{ij} + \sum_{i,j,m,n \in V, m<n, l \in L} cf^l_{ij} \cdot f^{mnl}_{ij} \tag{1}$$

2.5 Flow conservation constraints

For all demands d_{mn}, the net flow out of nodes m and n must be d_{mn} and $-d_{mn}$ respectively.

$$\forall m, n \in V, m < n:$$

$$\sum_{i \in V \setminus \{m\}, l \in L} f^{mnl}_{mi} - f^{mnl}_{im} = d_{mn} \tag{2}$$

$$\sum_{i \in V \setminus \{n\}, l \in L} f^{mnl}_{ni} - f^{mnl}_{in} = -d_{mn} \tag{3}$$

For all demands d_{mn}, the net flow in/out of all nodes $i, i \neq m, i \neq n$ must be 0. I.e. no flow should "pile up" anywhere.

$$\forall m, n \in V, m < n, i \in V \setminus \{m, n\} : \sum_{j \in V \setminus \{i\}, l \in L} f^{mnl}_{ij} - f^{mnl}_{ji} = 0 \tag{4}$$

Using these constraints only will allow flows to split, i.e. a demand may be satisfied using multiple paths. By adding constraints which for all demands d_{mn}, and all nodes $i \in V \setminus \{m, n\}$ require that, either the entire demand passes through i or the demand does not pass through the node, it is ensured that the flow cannot split. This is ensured by the following constraints.

$$\forall m, n \in V, m < n, i \in V \setminus \{m, n\} : \sum_{j \in V \setminus \{i\}, l \in L} f^{mnl}_{ij} + f^{mnl}_{ji} = 2 \cdot d_{mn} \cdot p^{mn}_i \tag{5}$$

Using these constraints, constraints of type (4) are redundant.

2.6 Capacity Constraints

Capacity constraints ensure, that no edge has more flow than its capacity allows.

$$\forall i \in V, j \in V\backslash\{i\}, l \in L : \sum_{m\in V, n\in V, m<n} f_{ij}^{mnl} + f_{ji}^{mnl} \leq cap_{ij}^{l} \cdot x_{ij}^{l} \tag{6}$$

The capacity constraints additionally ensures, that an edge of level l is only used ($f_{ij}^{mnl}>0$) if it is set up ($x_{ij}^{l} = 1$).

Recall that we assume that demands are demand-connected. Since the flow constraints ensure that demands are fulfilled and the capacity constraints ensure that edges are only used if they are set up, connectivity of the resulting network is also ensured.

2.7 Constraints Relating to Hierarchies

Between two nodes there can be one edge only:

$$\forall i, j \in V, i < j : \sum_{l\in L} x_{ij}^{l} \leq 1 \tag{7}$$

A node is in at most one group at each level:

$$\forall i \in V, l \in L : \sum_{h\in G_l} g_i^{lh} \leq 1 \tag{8}$$

Each group has exactly one concentrator node (except the highest level group which has none):

$$\forall l \in \{2, \ldots, |L|\}, h \in G_l : \sum_{i\in V} t_i^{lh} = 1 \tag{9}$$

A node can only be concentrator in one group:

$$\forall i \in V, l \in \{2, \ldots, |L|\} : \sum_{h\in G_l} t_i^{lh} \leq 1 \tag{10}$$

If there is an edge of level $l < |L|$ between nodes i and j ($x_{ij}^{l} = 1$), then node i and j are concentrator nodes for a group of level $l+1$ ($\sum_{h\in G_{l+1}} t_i^{(l+1)h} = 1$ and $\sum_{h\in G_{l+1}} t_j^{(l+1)h} = 1$).

$$\forall i, j \in V, i < j, l \in \{1, \ldots, |L| - 1\} :$$

$$x_{ij}^{l} \leq \sum_{h\in G_{l+1}} t_i^{(l+1)h} \tag{11}$$

$$x_{ij}^{l} \leq \sum_{h\in G_{l+1}} t_j^{(l+1)h} \tag{12}$$

If i is concentrator node of a level $l+1$ group ($\sum_{h \in G_{l+1}} t_i^{(l+1)h} = 1$), then there is an edge of level l incident to i ($\sum_j x_{ij}^l \geq 1$).

$$\forall i \in V, l \in \{1, \ldots, L-1\} : \sum_{h \in G_{l+1}} t_i^{(l+1)h} \leq \sum_{j \in V, i<j} x_{ij}^l + \sum_{j \in V, j<i} x_{ij}^l \tag{13}$$

If a node is concentrator in group h of level l, then it is in the group as well.

$$\forall i \in V, l \in L, h \in G_l : t_i^{lh} \leq g_i^{lh} \tag{14}$$

If a node i can be reached from j via a link of level l then i and j are in the same level l group. This is expressed using the following two sets of constraints:

$$\forall i, j \in V, i<j, l \in L, h \in G_l : x_{ij}^l + g_i^{lh} \leq g_j^{lh} + 1 \tag{15}$$

$$\forall i, j \in V, i<j, l \in L, h \in G_l : x_{ij}^l + g_j^{lh} \leq g_i^{lh} + 1 \tag{16}$$

If a node i is in a group of level l then it must be concentrator in a level $l+1$ group.

$$\forall i \in V, l \in \{1, \ldots, |L|-1\}, h \in G_l : g_i^{lh} \leq \sum_{h' \in G_{l+1}} t_i^{(l+1)h'} \tag{17}$$

If a node i is concentrator for a level $l+1$ group, then it must be in a group of level l.

$$\forall i \in V, l \in \{1, \ldots, |L|-1\}, h \in G_{l+1} : t_i^{(l+1)h} \leq \sum_{h' \in G_l} g_i^{lh'} \tag{18}$$

These constraints correspond to the description of hierarchical networks given previously. The description does not limit the number of levels. In the sequel, however, we will for simplicities work with 2 levels only - a primary and a secondary level. The set of secondary groups is abbreviated by G corresponding to G_2 - note that there is only one primary group.

3 Solution Strategy

A branch-and-bound based solution algorithm has been implemented which is able to find optimal solutions to networks of up to 15 nodes. However, a network consisting of 15 nodes is small compared with real world telecommunication networks, which have hundreds, maybe thousands of nodes. Hence, a simulated annealing algorithm has been designed, which in approximately 1 hour finds reasonable solutions for the HNP with 100 nodes.

The branch and bound algorithm has provided ideas for the simulated annealing algorithm, and the solution strategy used (as described in this section) is the same for both algorithms.

The strategy is based on a division of the solution process into two phases, allowing us to solve the problem using two separate algorithms. The phases are:

1. Divide the network into groups and choose concentrator nodes.
2. Optimize each group in turn wrt. communication.

The first phase provides the division into groups and selects concentrators (thereby creating the primary group). The different group divisions and concentrator selections are traversed using a simulated annealing algorithm (described in section 4). During the second phase each group is optimized in turn, using a construction algorithm (described in section 5).

The phase division is exemplified in figure 4 and figure 5. Both figures shows the same example network. Distances are assumed to be Euclidean, whereas the demand between pairs of nodes and the capacity of edges is unspecified, but assumed to be appropriate, i.e. a solution exists and the suggested selection of edges allows routing the demand in the network.

Figure 4 shows the network divided into groups and for each secondary group a concentrator is indicated as well. This is the output of phase one.

Figure 5 shows a selection of edges, which together with the groups and concentrators make up a HN. The output of phase two is the selection of edges and a specification of how the demands are routed using these edges.

3.1 Optimizing a Group

Dividing the solution process into phases eases the optimization process carried out by phase two. The key point is that during phase two, each group can be optimized in turn without taking into account how other groups are optimized. This is described thoroughly in the following.

For a two level HN, flow between two nodes consists of up to three paths which can be determined independently. To illustrate this consider the example in figure 6, which shows a path for demand ad.

Let $c(a)$ denote the concentrator of the group containing node a. If two nodes a and d are in different groups, the flow will have to go through the primary group (cf. figure 6). In fact, since there is only one concentrator in each group, the flow path

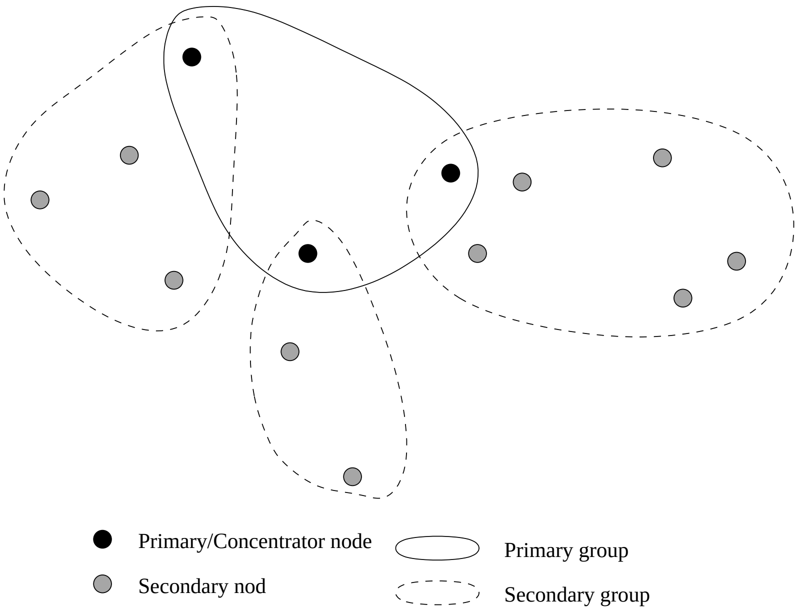

Figure 4: *Example network. Groups defined and concentrators selected at the left*

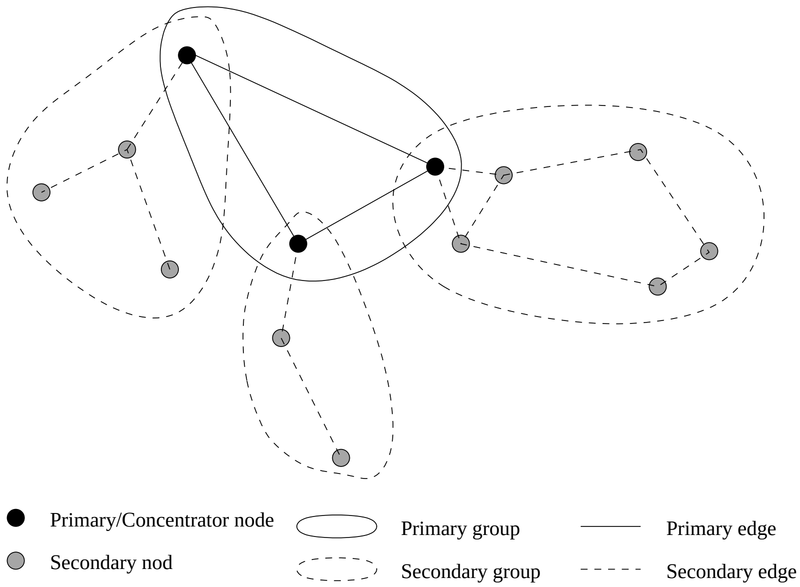

Figure 5: *Example network. Groups defined, concentrators* and *edges selected*

consists of three sub-paths: One from a to $c(a)$, one from $c(a)$ to $c(d)$ and one from $c(d)$ to d. These sub-paths can be determined independently.

For a demand d_{ij}, one or more sub-paths may be empty if either i or j is concentrator

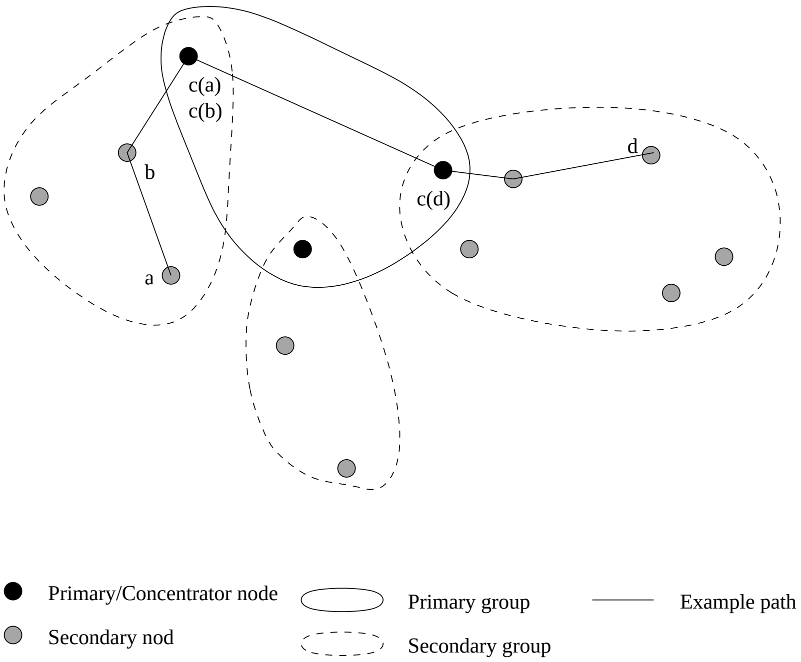

Figure 6: *Example path*

in a group, as e.g. the demand and corresponding path between $c(a)$ and d in figure 6. If i and j are in the same group as e.g. a and b in figure 6, the path is determined within the group.

We note that the sub-paths determined belongs to one group each, i.e. both end-points of all edges in a sub-path are in the same group. Hence, since for all demands, the path can be divided into sub-paths which can be determined independently, each group including the primary group can be optimized independently of the other groups.

Each such sub-path corresponds to a sub-demand. Some sub-demands start and end in the same group. These are bundled, i.e. they are handled as if they were one demand in order to decrease runtime. Bundling demands may as a result reduce the solution space. A group which may have a feasible solution if bundling was not done may be infeasible if bundling is done. Demands are not allowed to split, and bundling demands prevents demands in the bundle from splitting. If this turns out to be a major problem, all demands can simply be routed without bundling.

When solving groups, concentrators participate in two groups - the primary group and the secondary group in which they are the concentrator. The solution to the original problem is determined by summing up costs for the groups and aggregating sub-paths.

4 Phase One: Simulated Annealing Algorithm

Our simulated annealing algorithm is described in the following. The solution space consists of all possible divisions of groups and in conjunction with that, all possible selections of concentrator nodes. The algorithm is outlined in figure 7.

```
Find initial solution
Initialize temperature
do
   Pick neighbour-solution at random
   if neighbour-solution is feasible then
      if neighbour-solution is better than current then
         Update current
      else
         Update current with a probability that depends on
            solution value difference and temperature
   Update temperature
until stopping criteria is met
```

Figure 7: *The Simulated Annealing algorithm*

A current solution is maintained. The solution space is traversed by selecting a neighbour picked at random. If the solution value is better than the current solution value, the current solution is replaced. If on the other hand the solution is worse than the current solution it may be accepted. The idea is that while the temperature is decreased, fewer of the worse solutions are accepted and the current solution converges to a local optimum. In the following subsections the initial solution, neighbourhood, temperature update scheme, accepting criteria and stopping criteria is specified.

4.1 Initial Solution

The selection of a good initial solution has a positive effect on the solution quality and/or runtime compared with selecting a completely random initial solution. The idea is to divide the network into $|G|$ groups of equal size where nodes in the groups are close. The distance is measured as the primary setup-cost, but in the test instances this makes no difference, since costs are all proportional to the Euclidean distances.

At first the two nodes which defines the diameter of the network (i.e. are farthest from each other) are selected. Then the node with the highest average distance to the already selected nodes are selected. This is continued until $|G|$ nodes have been selected. Each selected node defines a group. Remaining nodes are assigned to groups such that each group contains at most $\lceil |V|/|G| \rceil$ nodes. This is done by solving an assignment problem using the distances as the costs.

Finally the concentrators are selected by first selecting the two nodes which are closest but are in different groups. The next concentrator selected is the concentrator which has the lowest average distance to already selected concentrators and is in a group where no concentrator is selected. This is continued until all $|G|$ groups have a concentrator.

The edges and flow is determined as described in the following section. If an infeasible solution is found at first, alternative group divisions are tried. The test instances have been generated such that this problem does not occur. If finding a feasible solution is a problem, the algorithm can be modified to accept infeasible solutions, but with a high penalty on the objective value if a solution is infeasible. Hereby, the simulated annealing algorithm can be used to search for a feasible solution.

4.2 Neighbourhood

The neighbourhood consist of two kinds of neighbour solutions:

- Group-neighbour
- Concentrator-neighbour

A group-neighbour is constructed by moving one non-concentrator node to another group. A concentrator-neighbour is constructed by selecting a new concentrator in one group. This is exemplified in the following figures 8, 9 and 10 which shows an example network, a group-neighbour and a concentrator-neighbour respectively.

In figure 9, node a is moved from one group to another. In figure 10, node b is selected as concentrator instead of node a.

4.2.1 Limiting the Neighbourhood

The number of neighbour-solutions is large. Let $|G|$ be the number of groups in the network and $|V|$ the number of nodes. The number of group-neighbours is $(|V| - |G|) \cdot (|G| - 1)$, since each non-concentrator node can be moved to any of the other groups. The number of concentrator-neighbours is $|V| - |G|$, since this is the number of non-concentrator nodes. The total number of neighbours is hence substantial, and there are many more group-neighbours than there are concentrator-neighbours.

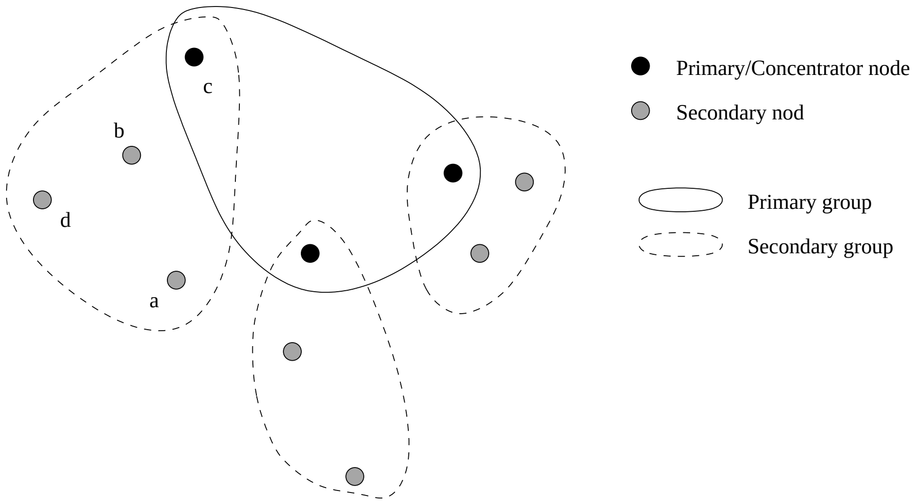

Figure 8: *Example network*

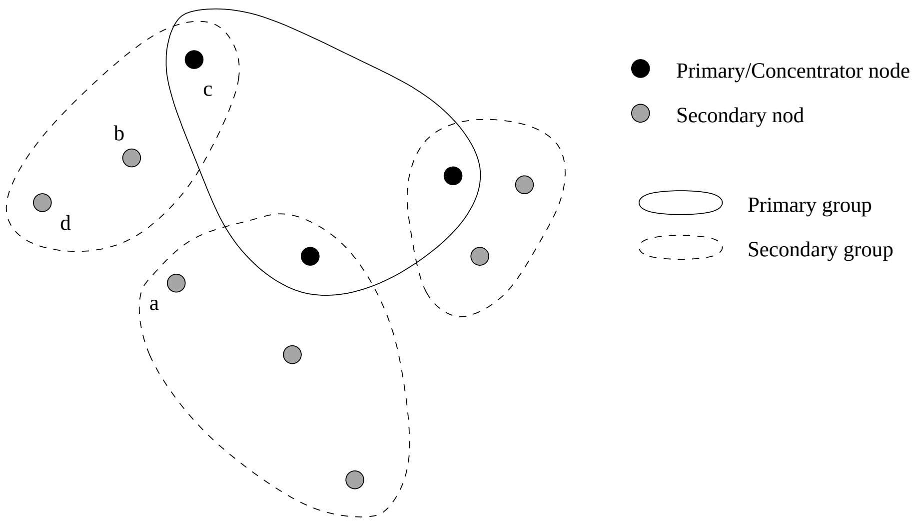

Figure 9: *Example on a group-neighbour*

Problems arising from networks in the real world usually have the property that they consist of points in a plane. In this case many potential group-divisions are not interesting, since usually good solutions do not contain groups with members scattered in the plane.

Thus neighbour-solutions which are constructed by moving a node to a distant group (measured in an appropriate way) from the other nodes in the group are not beneficial. Such an example is shown in figure 11 where node d is moved to a group which is not "close" to the node.

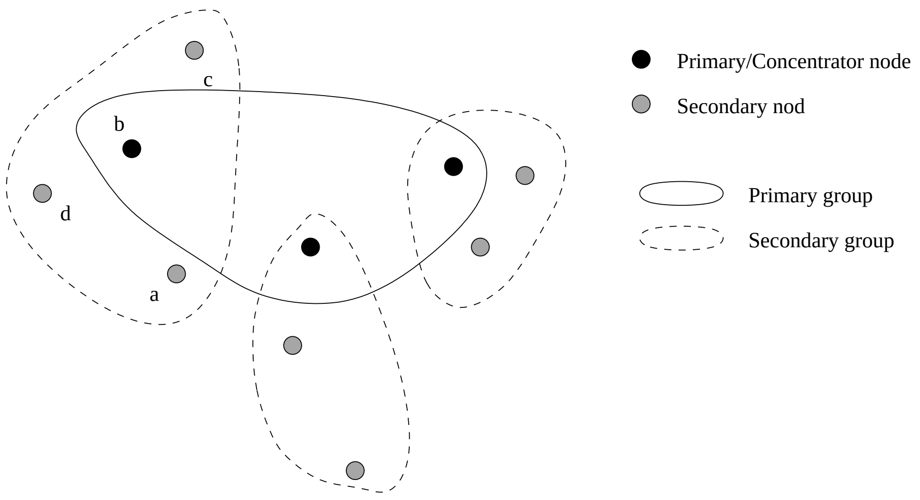

Figure 10: *Example on a concentrator-neighbour*

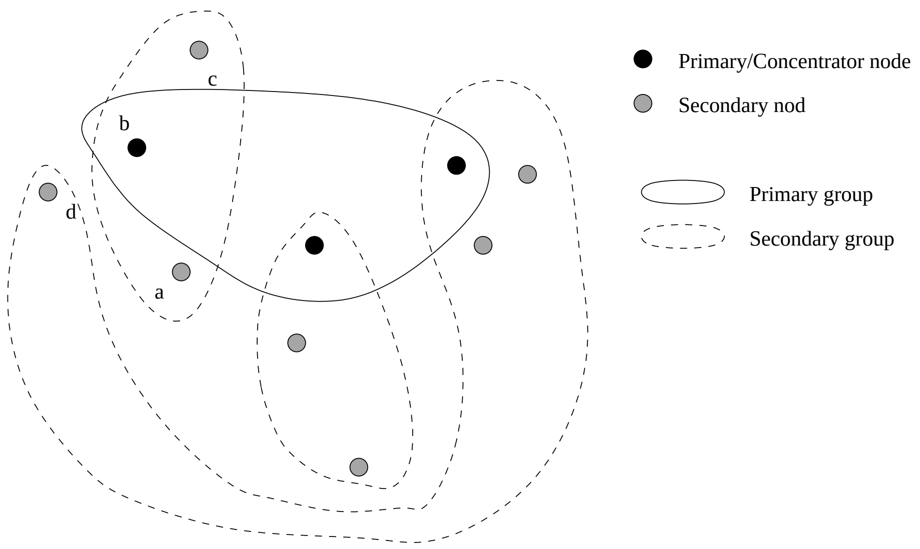

Figure 11: *Example on a group-neighbour solution which is probably not beneficial*

For these reasons, the number of group-solutions is reduced as follows. For a given group the nodes to move to the group is limited to be only the $\lceil |V|/|G| \rceil$ nodes, which are closest to the concentrator node of the group, i.e. has the lowest setup-cost to the concentrator. Since there are $|G|$ groups this amount result in approximately $|V|/G \cdot |G| = |V|$ group-neighbours, which is roughly as many as there are concentrator-neigbours. Thus when selecting neighbours randomly, each of the types are considered equally often and in total there are approximately $2|V|$ neighbours.

4.3 Temperature, Accepting Criteria and Stopping Criteria

The temperature used for iteration $i+1$ is found by multiplying the temperature of iteration i with a positive factor less than one.

Neighbour solutions are accepted if they are better than the current solution. If they are worse they are accepted with probability $e^{-v/t}$, where v is the increase in solution value expressed in percentages and t is the temperature. Thus the probability of acceptance increases with decreasing v and decreases with decreasing t.

The stopping criteria used is a sliding window criteria, i.e. the improvement is measured for the last k iterations, and if no change has occurred within these iterations, the algorithm is stopped. The number of iterations is adjusted in order to ensure that the algorithm converges to a local minimum. 200 iterations are carried out without any changes before the algorithm stops.

4.4 Tuning Parameters

In order to make simulated annealing perform well parameters have to be tuned properly. Given the above selections, the only parameters to be tuned are the initial temperature and the factor used to update the temperature.

Some effort has been made to make the choice of temperatures independent of costs (setup and flow). Therefore, in the following the increase in solution value used for the accepting criteria (v) is expressed in percentages. Also the choice of initial temperature and update factor depend critically on the number of nodes in the network. Schemes have been set up, which given the number of nodes in the network give default values which have been observed to perform well. If it is critical to obtain high quality solutions, the parameters should be tuned manually. This can of course only be done if few examples are optimized. The results reported for tests use default settings.

These default settings have been found by considering test runs for problems with a varying number of nodes. In particular it has been useful to record and depict the current solution value as a function of the iteration number.

5 Phase Two: Group Solution Algorithm

Given the division into groups and the selection of concentrators found in phase one, $|G|+1$ groups are created and optimized in turn. This is done using a greedy construction heuristic and a steepest descent local search algorithm. The algorithm is outlined in figure 12.

Find a minimum spanning tree solutionwith respect to the setup-cost
while solution is infeasible **do**
  Add an edge which relieves an overloaded edge
  Find demand paths
end while
Run local search

Figure 12: *Heuristic solution algorithm for groups*

Recall that flow is not allowed to split. Also, if we assume that for a group (and hence in general for a level), cap_{ij}^l is the same for all edges, we can check that a feasible solution exists as follows: If there exists two nodes i and j such that $d_{ij} > cap_{ij}^l$ where l is the level of the group we are optimizing, no feasible solution exists, since flow cannot split. If this is not the case, a feasible solution can be constructed by selecting all edges and for all demands d_{ij} use the path consisting of edge ij only. This way each edge will carry exactly one demand, and since no demand is larger than the capacity, the solution is feasible.

The construction algorithm finds a minimum spanning tree with respect to the setup-cost. This is only a good idea, if the setup-costs are higher than flow-costs. If this is not the case, an alternative algorithm, which takes into account the flow-cost, should be used.

Secondly, edges are added such that overloaded edges are relieved until the solution is feasible. This is in close connection with finding paths for the flow. The paths are assigned by first assigning demands d_{ij} for which an edge e_{ij} exists to the path consisting of edge ij only. These demands can all be routed, since initially we have checked that no demand exists, which exceeds the capacity of edges. Remaining demands are assigned to paths by considering demands in decreasing order along the shortest path. If some demand cannot be fulfilled, the edge which does not allow the demand to be routed is recorded for later use and temporarily removed. An alternative shortest path is found again. This is continued until either a feasible path is found or no path exists.

If a demand cannot be routed in the network, the recorded edge should be relieved. An edge which relieves the recorded edge is guaranteed to exists as can be seen from the following: Assume d_{ij} is the demand which cannot be routed. The recorded edge is on the shortest path between i and j, hence creating an alternative path for demand d_{ij} will relieve the edge. One such alternative path is the path consisting of edge ij only. Edge ij is not in the network since recall that all demands d_{ij} were edge ij was in the network were routed first and did not exceed the capacity. Hence edge ij is a relieving edge for the recorded edge. The particular edge selected is the shortest relieving edge.

Finally a local search which removes or adds a single edge at a time is run. Two versions have been tried. An iterative improvement algorithm and a steepest descent algorithm. The iterative improvement algorithm modifies the solution whenever it finds an improvement. The steepest descent algorithm on the other hand considers all possible adds or removes of a single edge and selects the best improving modification. The algorithms perform comparably as we shall see in subsection 6.2.

Recently we realized that Minoux in [21] suggests an alternative construction algorithm similar to the one we have suggested. This construction algorithm starts out by buying all edges and assigns demands to the shortest path. Iteratively, the edge which decrease the solution cost the most is removed until no edge can be removed. A preliminary implementation showed that using Minoux's algorithm for phase 2 in most cases resulted in inferior solutions and the algorithm ran slower. It is, however, expected that the runtime can be decreased substantially, since the algorithm were implemented in the framework of the original algorithm. Since the solutions were inferior, this was not pursued any further.

5.1 Reusing Solution Value for Groups

Since a neighbour solution only modifies one or two of the secondary groups of the current solution, it is possible to reuse solution values calculated for unmodified secondary groups. One strategy is to save group solutions for the current solution only, such that neighbour solutions can benefit from this. Since the algorithm may return to the same or similar solutions, we can in fact do better and save the solution values for more than a single step. The amount of memory required to do this is relatively small, since in general less than 10000 iterations are carried out, thus at most 20000 groups will have to be saved. The group solutions are saved using an open hashing scheme.

6 Computational Results

A number of tests have been carried out to measure the runtime and estimate the quality of the solutions. All tests have been carried out on a SUN Fire 3800 with 8 750MHz Sparc III CPUs and 8GB RAM, running Solaris 8. The implementation does, however, only use one CPU at a time.

6.1 Test Instances

The test-instances have between 10 and 100 nodes and approximately $\sqrt{|V|}$ groups. Capacity on edges are the same for edges at each level, and primary edges can carry 4 times as much flow as secondary edges.

Each node is randomly located in the Euclidean plane. The setup- and flow-costs for both primary and secondary edges are found by multiplying the Euclidean distance with a constant. The constants are selected such that the primary setup-cost is the double of the secondary setup-cost. Also the secondary flow-cost is the double of the primary flow-cost.

For each edge, the maximum flow-cost is the capacity times flow-cost. The ratio between the primary and secondary costs are chosen, such that the maximum flow-cost equals the setup-cost in both the primary and secondary case. The values chosen are shown in table 1.

Parameter	**Value**
Primary edge capacity	400
Secondary edge capacity	100
Primary setup-cost for unit distance	400
Secondary setup-cost for unit distance	200
Primary flow-cost for unit distance	1
Secondary flow-cost for unit distance	2

Table 1: *Parameters for the data set*

The demand is generated for all pairs of nodes. For each node a weight is generated (uniformly distributed). Demand d_{ij} is then the sum of the weight of nodes i and j. To obtain proper test instances, the total amount of demand in the network should be such that a feasible solution exists, but also should not be too low. If the demand is too low, good solutions are tree-like, in which case routing the flow is easy. Routing the flow is handled by the phase two algorithm, and hence the influence of this algorithm will not be reflected if routing is to easy.

The total amount of demand is controlled by multiplying each demand by a factor. Determining a suitable total amount of demand is done by running pre-tests with varying total amount of demand. The solutions are analysed and on the basis of this, the total amount of demand for the systematic testing is chosen.

6.2 Performance Test

The solution quality is hard to measure, since the optimal solution value is not known. As an alternative, we have compared the solutions with solutions found where the phase one algorithm is replaced by an iterative improvement algorithm. This is accomplished by modifying the simulated annealing algorithm, such that it does not accept any worse solution. In both cases an initial solution is determined as described in subsection 4.1.

The phase two algorithm which solves the groups is implemented in two versions, a steepest descent version and an iterative improvement version (see section 5).

Tests are run on networks generated as described above. The solution quality is estimated by recording the solution value in all cases and calculating the deviation from the best in percent.

The solution deviation from the best solution for each test instance is shown in table 2. In the table, simulated annealing is abbreviated SA, iterative Improvement is abbreviated II and steepest descent is abbreviated SD.

Phase one: Phase two:	SA II	SA SD	Greedy II	Greedy SD
Number of nodes	Solution value deviation			
5	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00
7	0.00	4.64	7.16	7.16
8	0.00	0.00	0.00	0.00
9	3.31	0.00	5.95	5.75
10	1.91	0.00	1.91	0.00
11	0.00	1.47	2.17	3.88
12	0.00	7.06	6.97	9.24
13	2.55	0.00	11.05	5.42
14	0.00	0.00	10.72	8.70
15	1.61	0.00	9.22	3.82
20	0.06	0.00	0.06	0.00
25	2.21	0.00	2.44	3.07
30	0.64	0.00	4.87	2.34
35	0.00	0.93	7.47	4.51
40	3.31	0.00	1.73	2.45
45	0.00	1.00	4.74	2.95
50	3.28	0.00	7.47	3.33
60	0.00	2.06	8.05	7.51
70	0.00	1.09	4.48	6.41
80	4.96	0.00	4.21	8.45
90	0.00	2.91	8.10	10.30
100	4.98	3.55	0.00	3.76

Table 2: *Solution value deviation from best solution*

The table shows that the simulated annealing algorithm finds better solutions than the greedy variant. For the phase 2 algorithm, it does not seem to matter much whether a steepest descent algorithm or an iterative improvement algorithm is used. In some cases one is the best in other cases the other one is.

Table 3 shows the runtime for both algorithms. As expected the simulated annealing algorithm requires considerably more time to run. The runtime using either iterative improvement or steepest descent in phase 2 performs similar for the greedy algorithm and the simulated annealing algorithm, respectively.

Phase one: Phase two:	SA II	SA SD	Greedy II	Greedy SD
Number of nodes	Runtime in seconds			
5	0.16	0.16	0.01	0.05
6	0.06	0.04	0.05	0.04
7	0.29	0.13	0.05	0.06
8	0.27	0.24	0.07	0.08
9	0.84	1.82	0.1	0.09
10	0.47	0.37	0.10	0.10
11	1.03	1.54	0.23	0.07
12	1.26	1.28	0.15	0.13
13	1.45	1.02	0.12	0.15
14	3.33	3.83	0.29	0.27
15	2.41	1.97	0.19	0.32
20	7.11	4.32	0.74	0.68
25	38.01	39.09	1.40	3.62
30	52.80	40.55	5.51	2.29
35	87.74	99.78	4.83	7.45
40	261.38	115.33	6.79	4.23
45	229.08	284.03	10.92	15.38
50	315.99	160.95	10.14	27.46
60	441.47	495.60	32.63	14.77
70	555.33	614.50	65.26	38.14
80	927.01	671.97	93.41	59.93
90	1093.97	591.71	82.73	60.30
100	1100.43	728.09	277.93	121.24

Table 3: *Runtime*

6.3 An example network

In figure 13 a solution to a HNP is shown. The HNP and solution is one of the instances used in the test described above.

The dotted lines indicate primary connections and the solid lines indicate the secondary connections. The thickness of the lines indicate how much of the capacity is used.

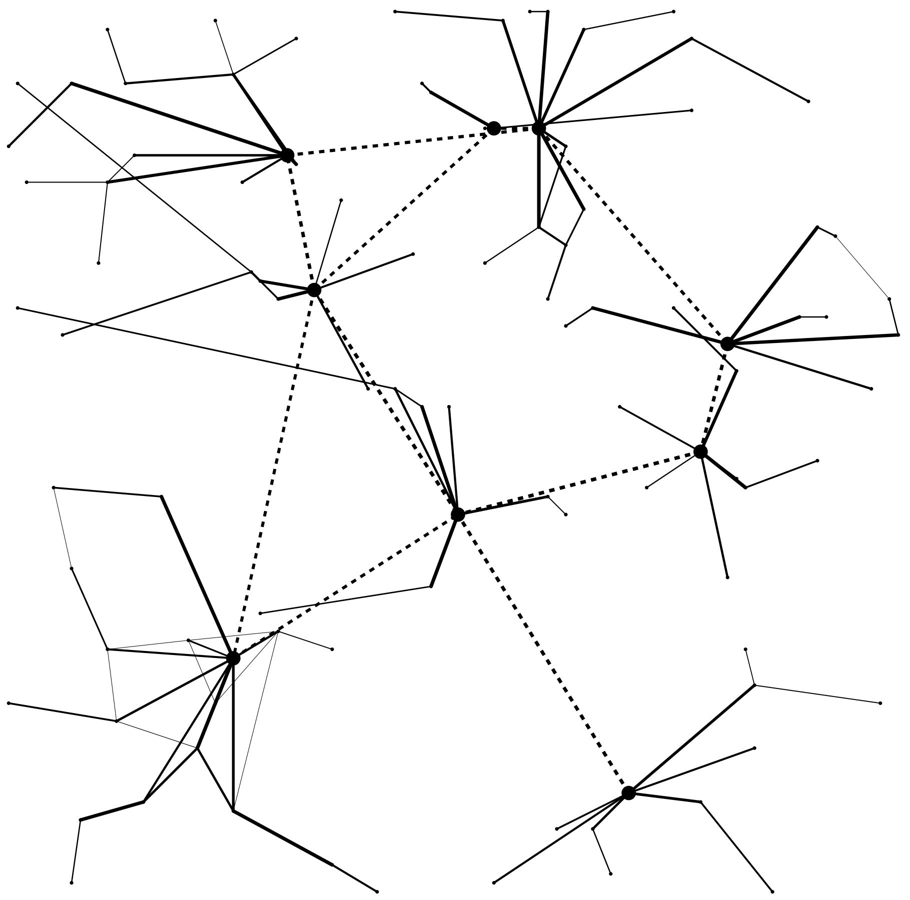

Figure 13: *Example network with 100 nodes in total and 9 concentrators.*

The primary connections between concentrators is a non-tree, whereas in most cases the groups are tree-like and in some cases even star-like with the concentrator as the root. The solutions for the HNP's in general appears to be like this.

In order to explain this, recall that between any pair of nodes there is a demand. Since the number of nodes in a group is usually much lower than the number of nodes outside the group, the demand to the concentrator is on average substantially higher than the demand between non-concentrator nodes in the group. Thus in order to ensure capacities and minimize the flow-cost, a tree- or star-like solution is usually a good choice.

7 Conclusion

In this paper we have presented and defined HN's and the HNP. This is to some extent based on, but not limited to, how telecommunication networks are organized and how these can be designed.

The HNP is solved heuristically using a simulated annealing algorithm and construction algorithms for optimizing groups. The algorithm is capable of finding reasonable solutions for the HNP where networks have up to 100 nodes. The running time is approximately one hour.

Topics for further investigation are to allow flows to split and to allow more than two levels. Also robustness of the network (e.g. ensuring that 2 disjoint paths exists for all demands) is of major interest. It would also be interesting to apply the method to a real world telecommunication network or to take the exact opposite direction and simplify the model (e.g. by ignoring capacities and flow-cost) and to solve the problem to optimality.

For the suggested solution method, it would be interesting to apply alternative meta-heuristics as e.g. GRASP and alternative construction algorithms for phase two. One possibility is to use Minoux's construction algorithm [21]. This way, solution quality and runtime might be improved, maybe combined with limiting the number of edges considered to be only the shortest ones.

References

[1] Anataram Balakrishnan, Thomas L. Magnanti, and Prakash Mirchandani. Designing hierarchical survivable networks. *Operations Research vol. 46, Issue 1*, 1998.

[2] R.R. Boorstyn and H. Frank. Large-scale network topological optimization. *IEEE Transactions on Communications*, COM-25(1):29–47, 1977.

[3] F.R.B. Cruz, J. MacGregor Smith, and G.R. Mateus. Algorithms for a multi-level network optimization problem. *European Journal of Operational Research*, 118(1):164–180, 1999.

[4] F.R.B. Cruz, J.M. Smith, and G.R. Mateus. Solving to optimality the uncapacitated fixed-charge network flow problem. *Computers Operations Research*, 25(1):67–81, 1998.

[5] J. Current and H. Pirkul. The hierarchical network design problem with transshipment facilities. *European Journal of Operational Research*, 51(3):338–47, 1991.

[6] J.R. Current. The design of a hierarchical transportation network with transshipment facilities. *Transportation Science*, 22(4):270–7, 1988.

[7] J.R. Current, C.S. ReVelle, and J.L. Cohon. The hierarchical network design problem. *European Journal of Operational Research*, 27(1):57–66, 1986.

[8] L.R. Esau and K.C. Williams. A method for approximating the optimal network. *IBM System Journal*, 5(3):142–147, 1966.

[9] V. Gabrel, A. Knippel, and M. Minoux. Exact solution of multicommodity network optimization problems with general step cost functions. *Operations Research Letters*, 25(1):15–23, 1999.

[10] Ioannis Gamvros, Bruce Golden, and S. Raghavan. An evolutionary approach to the multi-level capacitated minimum spanning tree problem. *Sixth INFORMS Telecommunications Conference*, 2002. http://www.isr.umd.edu/TechReports/ISR/2002/TR_2002-18/TR_2002-18.phtml.

[11] B. Gavish. Topological design of telecommunication networks-local access design methods. *Annals of Operations Research*, 33(1-4):17–71, 1991.

[12] B. Gavish. Topological design of computer communication networks-the overall design problem. *European Journal of Operational Research*, 58(2):149–72, 1992.

[13] Andre Girard, Brunilde Sanso, and Linda Dadjo. A tabu search algorithm for access network design. *Annals of Operations Research*, 106(1-4):229–262, 2001.

[14] Luis Gouveia and Joao Telhada. An augmented arborescence formulation for the two-level network design problem. *Annals of Operations Research*, 106(1-4):47–61, 2001.

[15] K. Holmberg and D. Yuan. A lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. *Operations Research*, 48(3):461–81, 2000.

[16] A. Kamath, O. Palmon, and S. Plotkin. Fast approximation algorithm for minimum cost multicommodity flow. *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 493–501, 1995.

[17] J.L. Kennington and J.E. Whitler. An efficient decomposition algorithm to optimize spare capacity in a telecommunications network. *INFORMS Journal on Computing*, 11(2):149–60, 1999.

[18] L. Kleinrock and F. Kamoun. Hierarchical routing for large networks-performance evaluation and optimization. *Computer Networks*, 1(3):155–74, 1977.

[19] L. Kleinrock and F. Kamoun. Optimal clustering structures for hierarchical topological design of large computer networks. *Networks*, 10(3):221–48, 1980.

[20] P.V. McGregor and D. Shen. Network design: algorithm for the access facility location problem. *IEEE Transactions on Communications*, COM-25(1):61–73, 1977.

[21] M. Minoux. Network synthesis and optimum network design problems: models, solution methods and applications. *Networks*, 19(3):313–60, 1989.

[22] Michel Minoux. Discrete cost multicommodity network optimization problems and exact solution methods. *Annals of Operations Research*, 106(1-4):19–46, 2001.

[23] S. Narasimhan and H. Pirkul. Hierarchical concentrator location problem. *Computer Communications*, 15(3):185–91, 1992.

[24] J. Petrek and V. Siedt. A large hierarchical network star-star topology design algorithm. *European Transactions on Telecommunications*, 12(6):511–22, 2001.

[25] J. Petrek and A. Speetzen. Assignment modification in the design of multipoint communication networks. *Journal of Electrical Engineering*, 52(5-6):158–61, 2001.

[26] M. Pioro and P. Gajowniczek. Simulated allocation: a suboptimal solution to the multicommodity flow problem. *Teletraffic Symposium, 11th. Performance Engineering in Telecommunications Networks. IEE Eleventh UK*, page 31/1, 1994.

[27] M. Pioro, A. Jutner, J. Harmatos, Szentesi. A, P. Gajowniczek, and Myslek A. Topological design of telecommunication networks - nodes and links localization under demand constraints. *submitted to 17th International Teletraffic Congress, Salvador de Bahia*, 2001. "http://www.tele.pw.edu.pl/˜amyslek/papers/itc2001.pdf".

[28] M. Pioro, A. Myslek, A. Juttner, J. Harmatos, and A. Szentesi. Topological design of mpls networks. *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, 1:12–16.

[29] M. Pióro and T. Szymański. Basic reconfiguration options in multi-layer robust telecommunication networks - design and performance issues. *Teletraffic Engineering in the Internet Era*, pages 271–284, 2001.

[30] H. Pirkul, J. Current, and V. Nagarajan. The hierarchical network design problem: a new formulation and solution procedures. *Transportation Science*, 25(3):175–82, 1991.

[31] H. Pirkul and V. Nagarajan. Locating concentrators in centralized computer networks. *Annals of Operations Research*, 36(1-4):247–61, 1992.

[32] Hasan Pirkul and Rakesh Gupta. Topological design of centralized computer networks. *International Transactions in Operational Research*, 4(1):75–83, 1997.

[33] D Saha and A Mukherjee. Design of hierarchical communication networks under node/link failure constraints. *Computer Communications*, 18(5):378–383, 1995.

[34] N.G.F. Sancho. A suboptimal solution to a hierarchial network design problem using dynamic programming. *European Journal of Operational Research*, 83(1):237–244, 1995.

[35] N.G.F. Sancho. The hierarchical network design problem with multiple primary paths. *European Journal of Operational Research*, 96(2):323–328, 1997.

[36] G.M. Schneider and M.N. Zastrow. An algorithm for the design of multilevel concentrator networks. *Computer Networks*, 6(1):1–11, 1982.

[37] Tommy Thomadsen. Methods for hiearchical network desing. Master's thesis, Technical University of Denmark, 2002. http://www.imm.dtu.dk/pubdb/views/publication_details.php?id=431.