Control and Cybernetics

vol. **25** (1996) No. 1

Power network design

by

Hans F. Ravn*,*, Jens M. Rygaard*** and **Berno Wibbels****

*The Institute of Mathematical Modelling (IMM),
The Technical University of Denmark,

**The Technical University of Eindhoven,

***Risø National Laboratory,
e-mail: jmr@imm.dtu.dk

Abstract: In this paper an electrical power network design problem will be presented. This will be formulated as a Capacitated Network Design Problem (CNTD), which has not previously been described in the literature. The problem is NP-complete. Two heuristic approaches are proposed to solve it, i.e. a simulated annealing approach and a greedy algorithm approach. Results from solving real problems from a Danish power network are presented. These results indicate that the application of the simulated annealing algorithm on the power network design problem might result in relatively large savings, e.g. for the Danish power network in Zealand the savings were between 100 and 200 million DKR.

Keywords: Power network design, NP-completeness, simulated annealing.

1. Introduction

This paper deals with the problem of constructing an optimal electrical power network layout. The main assumption is that all demand sites and production sites have fixed locations (extra sites cannot be introduced). The freedom for the design therefore is in the choice of the *connections*. The layout requirements are:

- when one cable cannot be used, it should still be possible to redirect the flow through the network, such that all demand is met,
- every site should be connected to at least two other sites, and
- the network itself should be *connected*, i.e., it should be possible to reach every site from every other site, only using the selected connections.

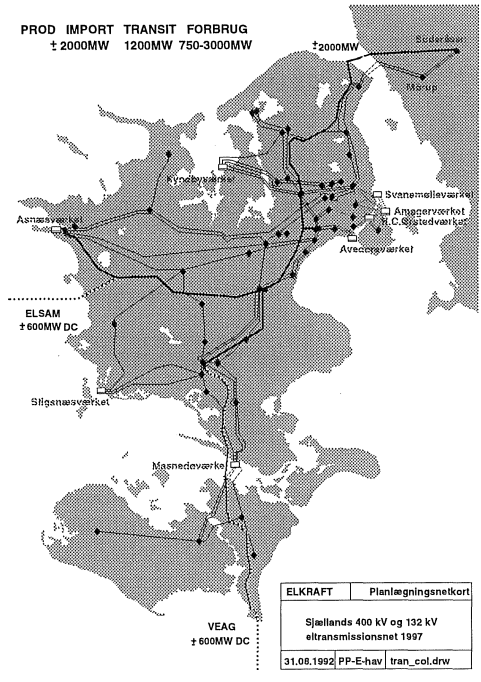

Figure 1. The current layout of the 132 kV and 400 kV network in Zealand

This problem of redesigning existing networks is relevant — for instance when overhead lines are put underground or to gradually improve the current network layout, making (local) extensions to an existing network, or even designing a network from scratch.

The motivation for the present work was the increased concern in Denmark about the potential danger that overhead power cables may interfere with the health.

The problem was defined by NESA, the power company in Zealand, and the work described in the present paper was done on close co-operation between NESA and IMM.

According to the literature, it seems that there are three mainstreams for designing networks. They are:

1. network design using a prescribed network layout (e.g. token ring),
2. network design including capacity constraints but no connectivity constraints, and
3. network design based on connectivity constraints, not taking capacity constraints into account.

The first strategy for network design is common in the computer industry, particularly in the design of LAN's (local area networks) and the like. However, in this field the *type* of layout is first chosen — using knowledge about the future task of the network — and only then optimized. This strategy is not relevant in the setting of our paper.

Articles that fall into the second category are, for instance, Andersen, Vidal and Iversen (1993), and Carniero, França and Silveira (1993). Articles that fall into the third category are, for instance, Monma and Shallcross (1989), and chapter 6 of Wu (1992).

From the descriptions it can be seen, that the second and third strategies both in a way solve half the problem. The problem at hand has both capacity constraints and connectivity constraints. Peculiarly enough, there seemed to be no articles that combined these two properties. For the second strategy it should be added that, although capacities are taken into account, an example of the use of multiple cables per connection (let alone *different types* of cables) was not found searching the literature.

One way to attack the problem could therefore be to solve it using approach 2 or 3, and then extending the resulting network such that it will also satisfy the relaxed constraint. This method however, should be used as a last resort only, since extending the network may make it far from optimal. In fact, extending the network optimally, means solving a new optimization problem that will be as difficult to solve as the original problem.

In general, capacity constrained problems are solved using branch and bound (B & B) schemes (where Andersen, Vidal and Iversen, 1993, is the exception; here, simulated annealing is used), and connectivity constrained problems, recognized as being closely related to the traveling salesman problem, are solved using heuristics. The examples shown for the B & B schemes are small; they have a small number of nodes to be connected (at most 20), or the number of connections to choose from, are limited. Note that the advantage of using B & B schemes instead of heuristics lies in the fact that one not only gets an upper bound on the solution, but also a lower bound. When one uses heuristics — simulated annealing being one of them — one only gets an upper bound.

Stoer (1992) uses the structure of the problem formulation (network design based on connectivity constraints) in a so called branch and cut scheme. This means that equations are added to the problem formulation — before and during the branching — that cut off parts of the solution space without excluding the optimal integer solution. She gets good results, but since we could not formulate the reserve flow constraint in a linear form, we could not apply her scheme.

Concludingly, one can say that what is new in this paper is the *combination* of capacity constraints and connectivity constraints, *and* the possibility of using multiple cables (even of different types) for one connection.

The problem is further motivated and defined in Section 2. Sections 3 and 4 summarize the problem formulation in verbal and symbolic forms, respectively. Section 4 also discusses computational complexity issues and solution

by mathematical programming methods. Section 5 describes the SA algorithm developed, and Section 6 gives illustrative examples and computational results. The paper is based on Wibbels (1995), where details may be found.

2. Problem description

The high voltage power networks operate on different voltage levels. In Denmark they are 10, 50, 132, and 400 kV. The communication between the various levels is through the transformer stations. Now one can argue that a different layout of, say, the 132 kV level influences the (optimal) layout of the 400 kV and the 50 kV level, simply because a 400 to 132 kV transformer station can be seen as a *production node* in the 132 kV level, but as a *demand node* in the 400 kV level. So if the transformer station should produce more, with respect to the 132 kV level, it may very well be the case that the layout of the 400 kV level has to change because of the greater *demand* of this node.

Nevertheless, it was decided to treat the levels separately, i.e. as if no connections existed between them. Apart from keeping the problem manageable within the time limits for the project, there was also a practical justification for this limitation. Namely, that the requirements for the design of the power network are only specified for the different voltage levels, i.e. there are no inter-level requirements. Therefore the *only* place where different levels influence each other is at the transformer stations, because of the aforementioned production/demand role these stations play.

The *locations* of the sites producing or requiring electricity, and also *how much* of this production or demand is assumed given. This is because creating new production sites or extending existing ones takes a very long time: partly because of all the procedures to go through before getting the allowance to build one, and also because acquiring the land to build them on is a difficult and lengthy process. And although energy consumption is growing steadily the increase is relatively slow, and therefore fairly easy to predict accurately. So the only freedom in designing the layout was in the connections, i.e. which sites to connect to each other.

The *capacities* of the cables have to be taken into account. When one designs other networks, e.g. fiber networks, where cable capacities are large compared to the flow that goes through, there is no need to do this, since the cable capacities almost never impose restrictions on the layout. As a consequence of the limiting factor that the cable capacities impose on the problem of designing power networks, it may be necessary to use *more than one* cable for one connection. So it is not an absolute limitation in that a *connection* has a maximum capacity, but rather a relative one: every extra cable used in a connection will result in extra fixed costs for it.

Furthermore, it is possible to use *different types* of cables, each characterized by its capacity (in MVA) and cost (in DKR per kilometer). As for the price of connecting site i to site j, only the fixed cost matter: there are no costs per

unit flow through a cable. (Fixed cost here means: a fixed cost per kilometer cable.) Therefore, the cost of connecting site i to site j, using one cable of some type, is the price per kilometer of that cable type, multiplied by the distance between i and j. Since it is possible to put cables in the ground wherever one wants, the distance is measured "as the crow flies".

Together with the possibility of using multiple cables per connection, the cost structure for a connection has a typical staircase form. As an example, assume we have two cable types, a and b, which have the following specifications:

	price per km ($\times 10^6$ DKR)	capacity (MVA)
a	2	3
b	3	6

The cost structure for a connection with a length of 1 kilometer is shown in Figure 2.

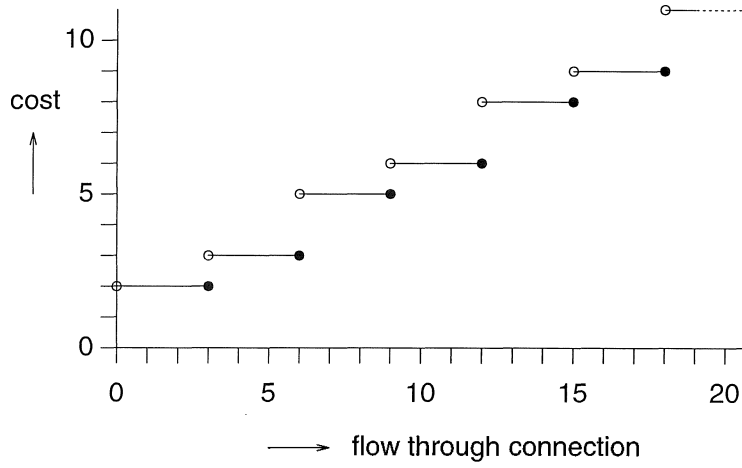

Figure 2. Illustration of the staircase cost function for a connection

So much for the parts the network is made of, now on to the requirements for the network layout.

An obvious and predominant requirement, is the following: all demands should be met, not overloading any connection. Because of the nature of electric current, it is not possible to *store* electricity at a site: it can only be used to fulfill demand, or sent on to another site.

Delivering the power under normal conditions is not enough. The network should also have a built-in safety, also sometimes called redundancy or survivability, ensuring that failure of cables does not automatically lead to sites being cut off from their electricity supply. This redundancy or survivability is different for every network and depends on the reliability of its building blocks, and the

damages that result from failures. For a highly reliable network, where failures do not lead to great inconveniences, it may even be completely absent.

When incorporated into the network design, there are two alternatives to state the degree of survivability. The first one defines the constraints in terms of possibilities, e.g. the possibility that a demand site in a network does not receive all demand is smaller than a certain parameter ρ, usually much smaller than one. Given the failure rates of the building blocks of the network, it is possible to calculate if a certain layout satisfies this constraint.

The second alternative is to require, for instance, that it should be possible to deliver the demand to a demand site through two *different* routings. Note that this formulation influences the design much more directly than the first alternative does. The advantage of this is that it is better suited to the use in a constructive approach to the network design problem.

The following constraints with respect to the survivability of the network were derived:

i when one cable cannot be used, it should still be possible to redirect the flow through the network, such that all demand is still met,
ii every site should be connected to at least two other sites, and
iii the network itself should be *connected*, i.e. it should be possible to reach every site from every other site, only using the selected connections.

Note that the implicit assumption underlying constraint *i* is that the possibility of two or more cables being out of order is negligible, or at least that *if* they do occur, they are not related. This means that it is assumed that if an error occurs in one of the cables, there will be no error in any of the cables used to redirect the flow. Also, we are talking about failure of *cables*, not of *connections*. This is because even if a connection consists of more than one cable, it is expected that *at most one* cable in this connection will fail to operate, or, in other words, that the failure of a cable in a certain connection has no influence on the failure rate of other cables in that connection. The assumption about the possibility of two or more cables being out of order therefore also includes the situation where these cables are in the same trench.

Finally, note that only the cables are mentioned in these constraints. The other building blocks of the network — production stations, transformer stations, and demand nodes — are left out.

Figure 3 is an illustration of the consequences that constraint *i* has. Suppose a choice is possible between two types of cables, one with capacity 50, the other with capacity 100, and that the site that is singled out, has a demand of 100 (the [100] in the figure). To deliver this 100, two cables of 50, or one of 100 can be used. Because of constraint *i*, it should be possible to still deliver 100 when a cable fails. Figure 3 a) now shows that another cable of 50 (the dashed line) should be put into the layout, in case two cables of 50 were used to connect the site to the rest of the network. When one cable of 100 was used, another one of 100 is needed. This is shown in Figure 3 b).

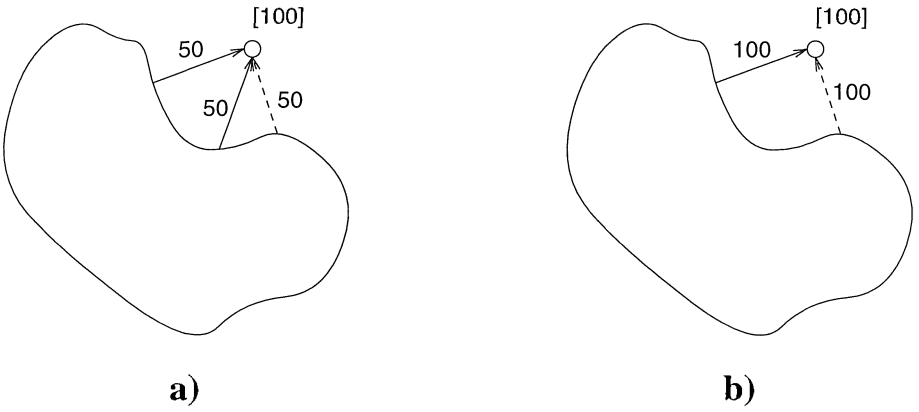

Figure 3. The concept of reserve flow

3. Formulation of the capacitated network topology design problem

Using the results from the previous section, the problem formulation for the capacitated network topology design problem (CNTD) can be stated as follows.

For given geographic locations of electricity stations, load points and transformer stations, it is required to determine the transmission network topology that requires minimal investments in connections. Observe the following items:

- ▷ Not only do we know the locations in advance, but also the demand and/or production capacities of the sites.
- ▷ There are different types of cables (different with respect to their *capacity*) available for making the connections, each with a specific cost per kilometer. For a certain connection, only these costs matter, i.e. the load of a connection is of no importance, just whether it is included in the design or not.
- ▷ The network connections have discrete MVA capacities.

Furthermore, the following restrictions apply:

- ▷ All demand must be met.
- ▷ Storage is not allowed.
- ▷ The network shall always be capable of transmitting the required electricity between the given points without overloading any connection.
- ▷ Reserve flow constraint (constraint *i* in Section 2).
- ▷ Connectivity constraints (constraints *ii* and *iii* in Section 2).

It should be possible to avoid or force using specific connections. For instance, cables that are already underground should be part of the solution, since they are already there and cost nothing extra when included in the design.

Some connections should not be used because of extra costs involved. Here we can think of connections going through large water areas: they are much more expensive than normal underground connections.

The possibility to avoid or force using specific connections, is also a good tool for the user of the program to see if a certain connection that he/she thinks should be part of the solution (or not) would result in a better design compared to the current situation.

4. Mathematical programming properties

In this section we give a more formalized description of the problem. This serves as the vehicle for two results. First, it may be argued that the problem is NP-hard and therefore it may be expected to be difficult or impossible to solve the problem of realistic size to optimality. Second, under certain simplifying assumptions the problem lends itself to solution by branch and bound, being an integer linear programming (ILP) problem.

For more information on NP-completeness, see Garey and Johnson, (1979); Papadimitriou and Steiglitz (1982). Let it suffice to say that a problem that is proven to be *NP-hard*, in general cannot be solved efficiently to optimality, meaning that the running times for algorithms that solve the problem increase exponentially as the size of the problem increases. For the CNTD the size is related to the number of sites that make up the network. In Wibbels (1995) it is shown that the CNTD problem is indeed NP-hard.

An integer linear programming (ILP) formulation is straightforward, as long as the constraints about *reserve flow* and *connectivity* (see section 2) are left out. In the literature the problem without these constraints and using only one cable between two sites (i.e. the Y_{ijk}'s in the ILP formulation below are *binary* instead of integer), is known as the fixed charge (or cost) network problem (FCNP) (see e.g. Sun et al., 1982).

It is easy to put the connectivity constraints into the formulation, using only *one* cable type (see (9)-(11)), but allowing for more than one cable per connection. However, we were not able to formulate the general case, where more than one cable type can be used, in a linear form. The difficulty arises because of the reserve flow constraint, which does not account for failing *connections*, but only for failing *cables*.

Therefore these constraints are just written down in words, without giving an expression for them. The corresponding formulation of the reserve flow and connectivity constraints is as follows.

First, we introduce the following symbol definitions.

The unknowns are the X_{ijk}'s and the Y_{ijk}'s. Note that we use a directed network. This means that in general $Y_{ijk} \neq Y_{jik}$ — this also holds for X_{ijk} and X_{jik}.

N	=	set of nodes, $N = \{1, \ldots, n\}$.
P	=	set of production nodes, $P = \{1, \ldots, p\}$, $p \leq n$.
K	=	set of cable types, $K = \{1, \ldots, q\}$.
f_k	=	cost per kilometer (in DKR) for cable of type k.
d_{ij}	=	Euclidean distance from node i to node j (in kilometers).
L_k, U_k	=	lower and upper bound on the capacity of a cable of type k, respectively.
S_i	=	available (resource) at node i ($i \in P$).
D_j	=	need (demand) at node j ($j \in N \setminus P$).
X_{ijk}	=	flow through cable of type k from node i to node j.
Y_{ijk}	=	number of cables of type k used to connect node i to node j.

The problem is formulated as follows:

$$min \quad \sum_{i \in N} \sum_{j \in N} \sum_{k \in K} f_k d_{ij} Y_{ijk}, \tag{1}$$

s.t.

$$\sum_{j \in N} \sum_{k \in K} X_{ijk} - \sum_{j \in N} \sum_{k \in K} X_{jik} \leq S_i \quad \forall i \in P, \tag{2}$$

$$\sum_{i \in N} \sum_{k \in K} X_{ijk} - \sum_{i \in N} \sum_{k \in K} X_{jik} = D_j \quad \forall j \in N \setminus P, \tag{3}$$

$$(L_k Y_{ijk} \leq)\ X_{ijk} \leq U_k Y_{ijk} \quad \forall i, j \in N \text{ and } \forall k \in K, \tag{4}$$

$$X_{ijk} \geq 0, \tag{5}$$

$$Y_{ijk} \in N, \tag{6}$$

$$\textit{connectivity constraints}, \tag{7}$$

$$\textit{reserve flow constraint}. \tag{8}$$

Explanation of the ILP formulation:

(1) : The objective function, equal to the fixed cost of all the connections chosen to be part of the network.

(2) : The first summation gives the total *outflow*, the second the total *inflow* of node i. So (2) says that the *net outflow* at node i should be less than or equal to the production capacity S_i of that node.

(3) : Alike (2). Now however, total *inflow* minus total *outflow*, i.e. the *net inflow* of node j should be equal to the demand D_j at that node (equal, and not greater than or equal, because no storage is allowed).

(4) : Capacity constraint on the edges. Flow from node i to node j, using cables of type k, cannot exceed (be lower than) the maximum (minimum) capacity of such a cable, times the number of cables of that type we are using. Usually, the minimum capacity of a cable equals zero.

(5) : The flow can be any nonnegative number.

(6) : The number of cables used of type k should be a (nonnegative) integer

number.

(7) : Every (demand) node should have a connection to at least two other nodes, and the network as a whole should be connected (constraints *ii* and *iii* in Section 2.)

(8) : When one cable fails, it should still be possible to make a feasible flow through the network, i.e. fulfill all demand (constraint *i* in Section 2.

It is assumed for the rest of this section that *only one* cable type is available. This means that in the ILP-formulation above the index k can be dropped.

The best way to implement the *reserve-flow* constraint seems to be the following. A *reserve demand* (RD_j) is added to the normal demand D_j of a site j. For one type of cable, the following applies (U is the capacity of the cable; mod denotes the *modulo* function):

$$RD_j = \begin{cases} U & \text{when } D_j \bmod U = 0, \\ D_j \bmod U & \text{otherwise.} \end{cases} \tag{9}$$

Or, in words, the reserve demand is set to the difference between the demand and the capacity of the connecting cables *minus one*. For example, let the demand D_j equal 27, and the capacity of a cable 10. To transport the demand, three cables are needed. The reserve demand RD_j now equals 27 mod 10 = 7. This is exactly equal to the capacity that is *short*, when one of those three cables fails.

To ensure that a site is connected to at least two other sites, the following restrictions are used:

$$Y_{ij} \leq (\sum_i Y_{ij}) - 1, \qquad \forall i \in N,\ j \in N \setminus P, \tag{10}$$

$$Y_{ij} + Y_{ji} \leq (\sum_i Y_{ij} + \sum_i Y_{ji}) - 1, \qquad \forall i \in N,\ j \in P. \tag{11}$$

Equation (10) assures that every *demand* site has *incoming* arcs from at least two other sites, and (11) assures that every *production* node has *incoming or outgoing* arcs to two other sites. This means that production sites just need connections to at least two other sites, whereas the demand sites need connections to at least two other sites *from which they (can) receive flow.*

Note that the second connectivity constraint, which says that the resulting network should be *connected*, is not implemented. The reason for this is that tests showed that the (10) and (11) were sufficient to also provide a connected network. If the network would not be connected, it suffices to add only a few extra constraints — those connectivity constraints that are violated by the current solution.

5. Simulated annealing

The explosion of running times for algorithms that solve NP-hard problems to optimality justifies the heuristic approach, i.e. to use an algorithm that does not

guarantee to solve the problem to optimality, but that has acceptable running times, even when the problem is large. Of course the algorithm should produce good solutions, i.e. the gap between the cost of the optimal solution and the cost of the best heuristic solution should be small (say, within 5 - 10 % of the optimal solution).

The simulated annealing algorithm (SA-algorithm) has been applied on various kinds of optimization problems with good results. In the following we will describe a SA-algorithm adapted to the CNTD problem.

Let $\omega(x)$ be the set of neighbor solutions to a solution x, T a temperature and $obj(x)$ the objective value of the objective function corresponding to x. The SA-algorithm can be described as follows:

1. Given a starting solution x_0, and starting temperature T_0.
   Set $x^* = x_0$, $T = T_0$.
2. **iteration** i:
   if [$\omega(x_i) \neq \emptyset$ and stopcriterion is not satisfied] then
   begin
   pick an element $x' \in \omega(x_i)$;
   if $obj(x') < obj(x_i)$ then $x_{i+1} = x'$;
   else
   begin
   if $obj(x') \geq obj(x_i)$ then $x_{i+1} := x'$ with probability $\exp(-\frac{obj(x')-obj(x_i)}{T})$;
   if $obj(x') < obj(x^*)$ then $x* := x'$;
   end;
   i:=i+1;
   Decrease temperature T;
   end
   else
   stop;

5.1. The neighborhood for undirected networks

A main step in the development of an SA-algorithm is the definition of a neighborhood generator (corresponding to "pick an element $x' \in \omega(x_i)$" above). The one we use in the undirected case is called an X-change by the authors of Steiglitz, Weiner, and Kleitman (1969). This mechanism yields a different *feasible* solution, when applied to an already feasible one.

The idea is as follows: select two edges belonging to the solution, say $[a, c]$ and $[b, d]$. Now there are two possibilities: remove these edges and connect the nodes again, using the edges $[a, d]$, and $[b, c]$ (as shown in Figure 4 (i)), or using the edges $[a, b]$, and $[c, d]$ (as shown in Figure 4 (ii)). Figure 4 (i) nicely shows why this mechanism is called an X-change.

A count of the degrees of the nodes a, b, c, and d, will show that these remain the same: before and after the X-change they are connected to the same

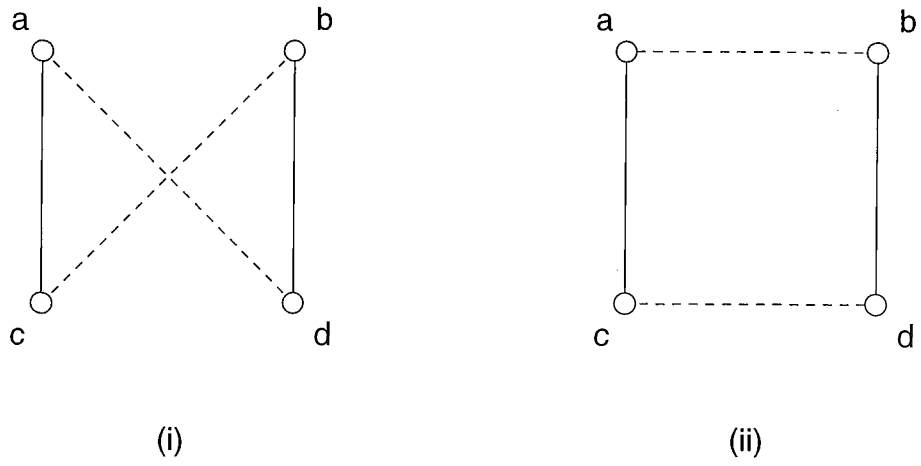

Figure 4. X-change possibilities

number of (other) nodes. This property of the X-change, which holds for all four nodes on which it acts, assures that a feasible solution will be changed into another feasible solution, which is exactly what we want.

In case the degree of the nodes equals two, we have to be careful. Suppose that the nodes a and b are connected via a path, and also that this is the case for the nodes c and d. When we would apply the second variant of the X-change, i.e. the one depicted in Figure 4 (ii), we end up with a *disconnected* network. That is why — in this undirected case — we check if the X-change preserves the connectivity. If not, we know we should take the other variant.

This X-change was used in an SA-algorithm. The implementation was tested on a small example, presented here before going on to the second stage of the development of the algorithm, in which we add flow to the network — and thereby direct the edges — and implement the capacity restrictions on cables as well as on the production and demand nodes. The example shows a 14-node network in the plane, which has connectivity requirement $s = 2$.

The first graph in Figure 5 shows a problem that may sometimes arise. The greedy heuristic produced a *disconnected* network. To make the network connected, the X-change can be used: select two edges from *different components* of the network, and apply one X-change. This will connect the two components from which the edges were chosen. Repeat this until the network is connected. Note that this is simply the reverse of the effect an X-change may sometimes have in that it *disconnects* the network at hand, and which made a check for connectivity necessary.

In the example, the edges $[3, 5]$, and $[10, 14]$ are chosen. During the X-change they are removed and replaced by the edges $[5, 14]$, and $[3, 10]$ (the *dashed* lines in the second graph in Figure 5).

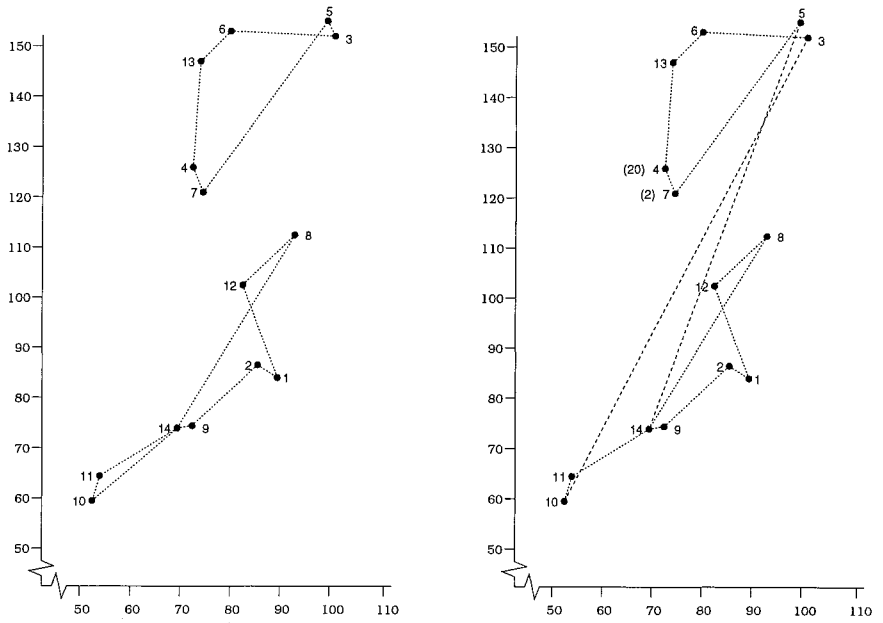

Figure 5. The greedy solution: disconnected and after connecting subnetworks

Using this initial connected solution as input for the simulated annealing program resulted in the solution shown in Figure 6. The solution is a tour on all points, i.e. if one starts in a certain node and follows the edges (clockwise, for instance), all nodes are visited *exactly once* before one returns to the starting point. This is not a surprise, since a version of CNTD is equivalent to the geometrical traveling salesman problem, see Wibbels (1995).

5.2. Orienting the network

If the cables used in the network would have a capacity which is large enough to transport all the flow, the development of the algorithm could stop here. But since the cables do have restrictive capacities, this means that to be sure that they are not overloaded, *flow-calculations* are needed. Therefore the choice was made to make the network a *directed* one. This section shows how the initial *directed* solution is obtained and how the X-change is adapted to cope with arcs instead of edges.

5.2.1. From edges to arcs

The first part in the construction of an initial solution remains the same, i.e. first an *undirected* initial solution is made, using the heuristic shown in Section 5. To *direct* the network, which means transforming the edges to arcs, these are

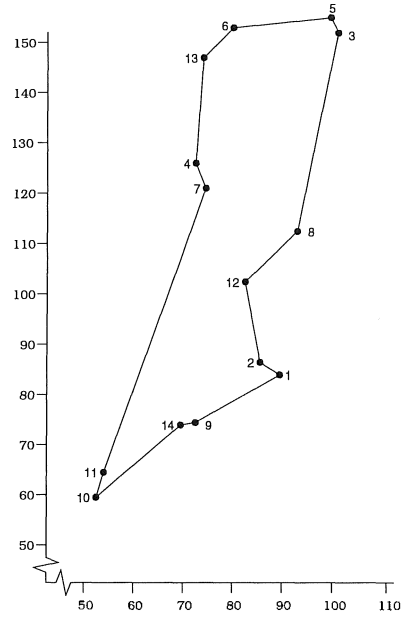

Figure 6. The final result using the normal X-change

simply doubled, as shown in Figure 7.

Assume that there is a directed network, satisfying the connectivity constraints. The next thing to do is to put the flow in. This is done in two phases. In the first phase, the demand of a certain node is added to the current flow. In order to do this, a path is calculated from a production node to this demand node. Then the demand of this node is added to the flow on the path, recalculating what cable types are needed for the connections. In the second phase, the reserve demand is added to the current flow.

To calculate which types of cables are used, and *how many*, a dynamic programming approach is used. This resulted in a recursive procedure which computes the cheapest connection — given the flow through it.

Finally, we should add that the connectivity constraints have been altered slightly because of the change from edges to arcs. To account for reserve flow, *demand nodes* should have *incoming* arcs from at least two different nodes. For *production nodes* it does not matter if the arcs are incoming or outgoing. So for those nodes, the connectivity requirement stays the same: connections to at least two different nodes. To check the total connectivity, it does not matter in which way the arcs are traversed. So the requirement is that the underlying *undirected* network — i.e. the current network with all the arcs replaced by edges — is connected.

The neighborhood sometimes does not preserve the feasibility of the network.

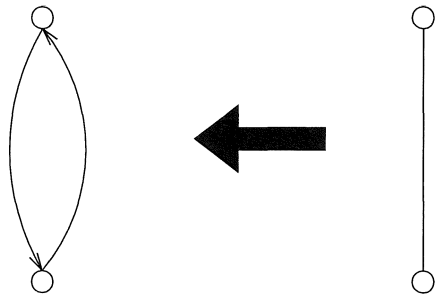

Figure 7. From edges to arcs

The *structure* of the network layout remained correct, but the reserve flow constraint is sometimes violated. Since the structure of the network is correct, i.e. it meets the connectivity constraints, we add a flow check to ensure that also the reserve flow constraint is satisfied again. Every time a subrun in the SA algorithm is completed, this flow check is used to check if the reserve flow constraint is not violated. If it is, extra flow is added to *existing* connections until this is no longer the case.

The reserve demand is added to the network with the help of a flow algorithm. After this flow check, all connections with zero flow that are not needed to fulfill the connectivity constraints, are removed. The resulting network serves as the initial solution for the simulated annealing algorithm.

To ensure a feasible solution after a neighbor generation, the *flow balance* at every node must remain equal to that before the generation. The way it is done in the program is that the X-change is adapted to incorporate flow on arcs, instead of working on edges. This mechanism will be denoted *adapted X-change.*

Instead of selecting two edges, as was the case with the X-change, we now select two *arcs.* Compute the minimum flow through these two arcs, and then perform an almost normal X-change. Almost normal, because now there is *just one* possibility which will result in a feasible solution, and it is known beforehand.

An example may clarify the above.

Suppose the arcs (c, a) and (d, b) were selected, with flow 50 and 100, respectively (as depicted in Figure 8 (i)). The minimum of the flows is 50, so do an almost normal X-change on the arcs, with flow 50. This means that afterwards there will still be an arc from d to b, having 50 flow.

Now the *only* way to ensure that the flow balance remains the same, is to remove 50 flow from the selected arcs (thereby removing arc (c, a)) and creating the arcs (d, a), and (c, b), each with 50 flow. The result is shown in Figure 8 (ii). Note that the flow balance indeed remained the same, i.e. the nodes a, b, c, and d send and receive *the same* amount of flow as before the adapted X-change.

This is deliberately done, to avoid that nodes receive neither more nor less

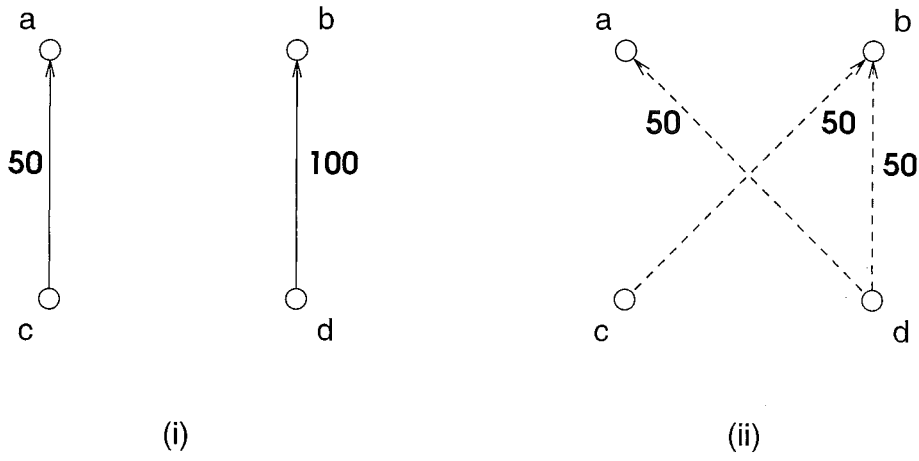

Figure 8. The adapted X-change

than they need. Therefore, no storage or shortage will occur, and we do not need to check this every time an X-change is performed. There are, however, checks needed for the connectivity of the solution. This is because we do not check if, for instance, the arc (d, a) was already part of the current layout. Then it follows that the X-change may result in a *decrease* of the number of incoming arcs. If, as a consequence, this leads to a node which does not have the desired connectivity any more, the X-change is reversed.

Intermediate tests showed that the adapted X-change did perform well, but that the final solution depended heavily on the initial one. This was a consequence of the strong point of the X-change: the fact that the flow balance remains the same. Therefore, the initial flow had too much influence on the final solution. Below we show the *second* neighborhood generator, called the flow swap, which was added to overcome this problem. It is a variant on the *one-optimal* heuristic from Monma and Shallcross (1989).

The Flow Swap

The Flow Swap consists of the following moves: first, select an arc (p_1, d), where p_1 is a *production node*. Second, select another production node, p_2, and connect this node to d, while removing the arc (p_1, d). This is shown in Figure 9. For node p_1 it is checked whether it still has the required connectivity after the Flow Swap. If not, the Flow Swap is reversed. Also, the remaining capacity of p_1 is increased with the flow through the arc, and that of p_2 is decreased by the same quantity. Node p_2 is chosen such that this is possible, i.e. its remaining capacity *before* the Flow Swap was larger than the flow through the selected arc.

The Flow Swap did improve the performance of the program considerably. Also, the influence of the initial solutions on the final result was diminished.

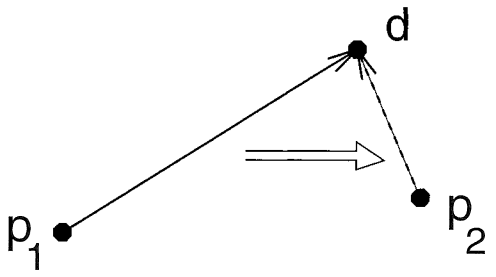

Figure 9. The Flow Swap mechanism

However, looking at the solutions, we could sometimes easily improve it and it seemed that neither the X-change nor the Flow Swap could. Therefore, we added a post optimization mechanism, called Triangle, which will be discussed below.

Post Optimization

Since we could easily improve some of the final solutions through a very simple mechanism, called Triangle, we implemented this and added it to the program.

In some solutions we found a layout as depicted in Figure 10: a node which had two outgoing arcs where at least one of the two had enough capacity to carry the flow of both connections. One can see that the cost for the layout (a, b) - (b, c) (and also for the layout (a, c) - (c, b)) is less than the cost for current layout, with the arcs (a, b) and (b, c).

Therefore, Triangle removes the current arcs, and replaces them with the cheapest alternative. In the figure, this would be the layout using arcs (a, b) and (b, c), where the arc (a, b) now will carry a flow of 35, and 25 flow is added to the arc (b, c) (if it existed, otherwise the connection is created first). Note that the Triangle post optimization again is a variant of the one-optimal improvement

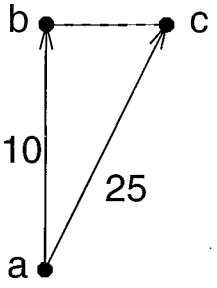

Figure 10. The Triangle post optimization

heuristic, as was the Flow Swap. Moreover, if a and b were production nodes, the Triangle mechanism and the Flow Swap look almost the same.

In fact, the next step would have been to merge the two mechanisms, creating a neighborhood generator on three points. This may still be done, but tests showed that the combination of X-change and Flow Swap neighborhoods together with the Triangle post optimization worked well enough.

The Cooling Schedule

In this subsection we describe the chosen cooling schedule. We will successively show our choices for the initial temperature, the stop criterion, the length of a subrun, and the decrement rule.

Initial temperature

The initial temperature t_0 is determined in the following way. Define an *acceptance ratio* χ, as the number of accepted transitions divided by the number of proposed transitions. Denote the acceptance ratio at the start: χ_0. The value of χ_0 should be close to 1, in order to assure that (almost) all of the solution space can be reached at the start (values range from 0.8 to 0.99 in the literature). We set χ_0 to 0.9. (A parameter value of 0.99 has also been tried, but that did not give better results).

To determine t_0, calculate the average increase in cost, $\overline{\Delta C}^{(+)}$, for a number of random transitions and compute t_0 from the equation

$$\chi_0 = exp\left(\frac{-\overline{\Delta C}^{(+)}}{t_0}\right).$$

Rewriting this equation leads to:

$$t_0 = \frac{\overline{\Delta C}^{(+)}}{\ln\left(\chi_0^{-1}\right)}.$$

These equations can also be found in van Laarhoven and Aarts (1987).

Stop criterion

The stop criterion that we implemented is a simple one: stop when no successful changes were made to the current solution in a subrun. A successful change is one that is *accepted*, i.e. when the current solution is replaced by a neighbor.

Length of subrun

The length of a subrun is limited by two parameters, the number of tries, and the number of successful tries, respectively. The first one is limited to a maximum of 100 times the number of nodes, meaning that no more than this number of neighbors is generated in a subrun. The subrun can end at an earlier stage, when the number of *successful* tries reaches its limit, which is set to 10 times the number of nodes. Typically, at a high temperature the maximum number of successful tries will limit the length of the subrun, whereas at a lower temperature the maximum number of tries is the limiting factor.

Decrement rule

A frequently used decrement rule, which is also implemented by us, is the *geometrical decrement rule*:

$$t_{k+1} = \alpha t_k, \quad k = 0, 1, 2, \ldots,$$

where α is a constant smaller than but close to 1, and t_k denotes the temperature at the kth subrun. In the program, this parameter is set to 0.9.

6. Examples

In this section we present a short overview of results obtained with the implementation of the simulated annealing algorithm. It is implemented in HP-UX Pascal — the Hewlett Packard variant of standard pascal — and in Turbo Pascal 7.0 (TP) — compiled in protective mode. Graphics are generated using the TP version.

All test problems presented are derived using practical data. We present results for networks with 8, 14, 25 and 61 nodes, respectively. Table 1 shows the data for the 8-node network.

name	x-coordinate	y-coordinate	demand	production
BEL030	121.3	179.1	80	0
AMK132	125.8	172.8	115	0
AMV030	128.2	177.4	45	540
HCV030	123.2	173.6	165	570
SMV030	125.2	180.0	40	95
VIK030	119.0	174.5	55	0
LIN050	120.2	176.7	50	0
GLN050	119.5	184.5	155	500

cable type	capacity	cost per kilometer
1	125	3 million DKR
2	170	4 million DKR
3	285	5 million DKR

Table 1. Data for the 8-node network

All results presented here are obtained on the average of solution using ten different initial solutions. The timing of the algorithm is done on a HP-750 computer, using the command line "`nohup time nice` *program*"

The tests showed, that the simulated annealing algorithm performs slightly better — in general — than the downhill algorithm. The downhill algorithm is implemented in the same way as the simulated annealing algorithm. It just does not accept neighbors that are worse than the current solution. For the networks

up to 25 nodes, we also used CPLEX to obtain lower bounds by relaxing the integer constraint for the Y_{ijk}'s and solving a one-cable type model with both reserve flow and connectivity constraints, (9)-(11). This means that we allowed the use of fractional cables. For networks larger than 30 nodes, CPLEX complained that the number of coefficients was too large. The maximum number of coefficients allowed for was 240 thousand.

Whenever possible we give the results for three solution methods described above: (i) a greedy solution (downhill) using the neighbor generators; (ii) a SA; (iii) a mathematical programming solution using CPLEX.

In the following tables, we present the test results. We only know the optimal solution for the 8-node network. For the other networks — except for the 61-node network — we give the lower bound obtained with CPLEX, as well as the best integer solution. We do not include running times for CPLEX, since this program ran out of memory after approximately 15 minutes. Furthermore, we show the results for the downhill algorithm and the simulated annealing algorithm in the one-type case, and in case more than one type of cable is available.

Optimal: 116.14		
	downhill	simulated annealing
best result (#)	116.14 (1)	116.14 (2)
worst result	159.77	160.62
average (% over optimum)	132.49 (14.1 %)	132.09 (13.7 %)
time	58.5	1:57.1

Table 2. Results for the 8-node network

In the optimal solution for the 8-node network only the smallest type of cable is used. It did not matter if we allowed the use of more than one type of cable or not.

	downhill	simulated annealing
best result (#)	3021.75 (1)	3080.67 (1)
worst result	3548.66	3839.61
average	3282.28	3497.26
time	6:21.0	8:58.8

Table 3. Results for the 14-node network (2 cable types)

The tables 3 and 4 show that the performance of the downhill algorithm for the 14-node network with two cable types is slightly better than the performance

of the simulated annealing algorithm. When only one cable type is present, the simulated annealing algorithm is clearly better. Both yield better results than the best integer solution found with CPLEX.

CPLEX		downhill		simulated annealing	
lb	best integer	1 type	2 types	1 type	2 types
3072.37	3343.09	3324.28	3021.75	3179.41	3080.67

Table 4. Results for 14-node network

Tables 5 and 6 show the results obtained for a 25-node network. The running time of the simulated annealing algorithm is almost five times that of the downhill algorithm, but the results justify this extra computation time, since the best solution found is more than 5 % cheaper — a difference of more than 200 million DKR.

Table 6 shows that CPLEX is no longer an alternative — when used without a more intelligent B & B scheme. This is also clear from the large gap between the lower bound and the best integer solution.

	downhill	Simulated Annealing
best result (#)	2818.53 (1)	2603.03 (1)
worst result	3310.69	3292.56
average	3007.34	2878.46
time	5:17.7	24:00.3

Table 5. Results for a 25-node network (3 cable types)

CPLEX		downhill		Simulated Annealing	
lb	best integer	1 type	3 types	1 type	3 types
2245.19	3130.75	3051.63	2818.53	2704.49	2603.03

Table 6. Results for a 25-node network

For the 61-node network, we can only give the results of the heuristics. The best solution found with the downhill algorithm is 2.4 % more expensive than the best solution found with the simulated annealing algorithm. Not much, but still a difference of over 100 million DKR. A picture of the best 61-node network found using three different initial solutions is shown in Figure 11. It clearly

	downhill	Simulated Annealing
best result (#)	4501.64 (1)	4443.82 (1)
worst result	5776.76	5150.68
average	4943.09	4827.17
time	1:50:03.9	5:50:24.9

Table 7. Results for the 61-node network

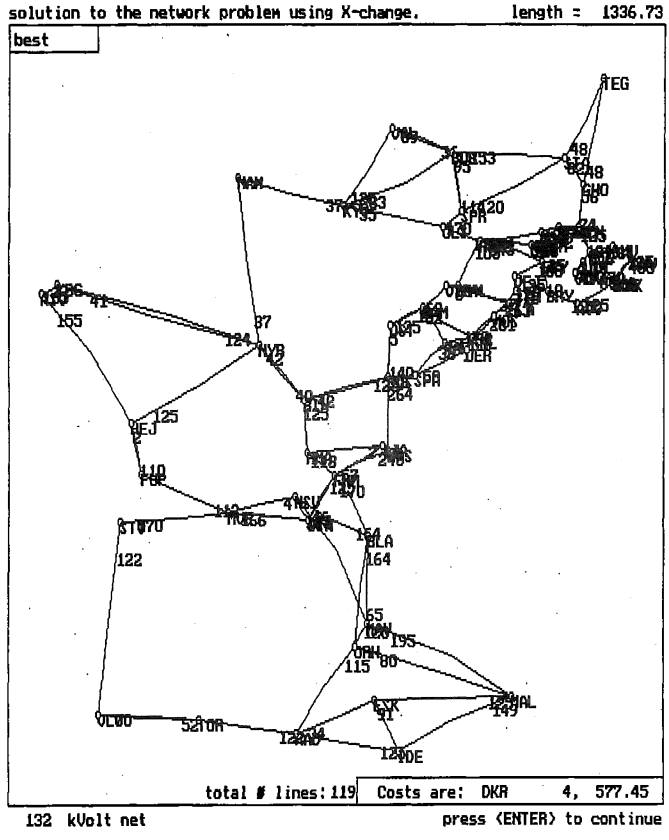

Figure 11. A solution for the 61-node network

shows that it is almost impossible to tell if a network design that consists of a lot of nodes can be improved.

7. Conclusions

The present paper has described the formulation of and the motivation for the CNTD problem, which is relevant for the redesigning of networks. The problem seems to be new. The motivation for the work was the concern in relation to redesign of the electrical power network at Sjælland. This problem is NP-hard and can not be solved to optimality in an efficient way. Therefore a simulated annealing strategy is chosen. The strategy was implemented as a computer algorithm.

The results described in Section 6 indicate that the SA-algorithm determines solutions that seem to be relatively close to the optimal solution. Although the greedy algorithm in general does not find as good solution as the SA-algorithm does, the greedy algorithm reaches a good solution much faster than the SA-algorithm does. Therefore a hybrid approach can be applied to solve this problem, i.e. to determine a good starting solution by the greedy algorithm is applied, and then this solution is the initial solution to the SA-algorithm.

From a practical point of view, the project was succesfull. The program yields good results with respect to the costs of the layout but also with respect to the structure of the layout, as judged by the end users at the planning department of NESA.

Further research should be done on optimizing networks with different voltage levels.

It may even be possible to construct an overall network, without splitting it up into several levels. Again, Balakrishnan, Magnanti and Mirchandani (1994) can be used as a start, but also other references about the design of multi-level networks.

Acknowledgement

We are grateful to Hugh Boyd, Claus Stefan Nielsen, Karin Lomholt and Jan Havsager, all at NESA, for helpful comments during the project.

More literature on the design of survivable networks can be found in the extensive Bibliography in Stoer (1992). For a survey on network design and synthesis, *without* survivability constraints, Minoux (1989) is recommended. Also, Ahuja, Magnanti and Orlin (1993) is a good start.

References

AHUJA, R.K., MAGNANTI, T.L. and ORLIN, J.B. (1993) *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, London.

ANDERSEN, K., VIDAL, R.V.V. and IVERSEN V.B. (1993) Design of a teleprocessing communication network using simulated annealing. In: R.V.V. Vidal, editor, *Applied Simulated Annealing*, Vol. 396 of *Lecture Notes in Economics and Mathematical Systems*, 201–216. Springer, Berlin.

BALAKRISHNAN, A., MAGNANTI. T.L. and MIRCHANDANI, P. (1994) Modeling and heuristic worst-case performance analysis of the two-level network design problem. *Management Science*, **40**, 7, 846–867.

CARNIERO, S., FRANÇA, P.M. and SILVEIRA, P.D. (1993) Long-range planning of power distribution systems: Primary networks. *Electric Power Systems Research*, **27**, 223–231.

GAREY, M.R. and JOHNSON, D.S. (1979) *Computers and Intractability: a Guide to the Theory of NP-Completeness.* Freeman, San Fransisco.

LAARHOVEN VAN, P.J.M. and AARTS, E.H.L. (1987) *Simulated Annealing: Theory and Practice.* Mathematics and its Applications. Reidel, Dordrecht.

MINOUX, M. (1989) Network synthesis and optimum network design problems: Models, solution methods, and applications. *Networks*, **19**, 313–360.

MONMA, C.L. and SHALLCROSS, D.F. (1989) Methods for designing communications networks with certain two-connected survivability constraints. *Operations Research*, **37**, 4, 531–541.

PAPADIMITRIOU, C.H. and STEIGLITZ, K. (1982) *Combinatorial Optimization: Algorithms and Complexity.* Prentice Hall, Englewood Cliffs (NJ).

STEIGLITZ, K., WEINER, P. and KLEITMAN, D.J. (1969) The design of minimal cost survivable networks. *IEEE Transactions on Circuits Theory*, **CT-16**, 4, 455–460.

STOER, M. (1992) *Design of Survivable Networks*, Vol. 1531 of *Lecture Notes in Mathematics.* Springer, Berlin.

SUN D.I. et al (1982) Optimal distribution substation and primary feeder planning via the fixed charge network formulation. *IEEE Transactions*, **PAS-101**, 602–609.

WIBBELS, B. (1995) *Power Network Design.* ECMI-project, no. 10. Institute of Mathematical Modelling, The Technical University of Denmark, Lyngby Denmark.

WU, T.-H. (1992) *Fiber Network Service Survivability.* Artech House, Norwood (MA).