Fixed-Charge Network Design and Protection Problems

Yao Liang

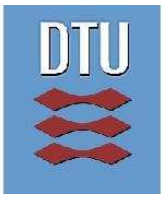
Supervisor: Thomas K. Stidsen

Master thesis

IMM-M.Sc-2006-47

Informatics and Mathematical Modelling

Technical University of Denmark



May 2, 2006

Abstract

This thesis describes eleven models in all for fixed-charge network design problems. Seven models, five of which implement protection schemes, are presented for large scale mesh and ring networks. Other four models are presented for small scale tree, star and bus networks. To check whether these models work well, some test instances are used, and their solutions are compared.

The discussion of different network types is also appended. The components and hardwares in the networks are analyzed. And from the perspective of cost, networks are divided into fixed-charge dominating style and cable costs dominating style. All these factors are considered in mathematical modelling.

Acknowledgements

With regard to the introduction of operation research in my study and the development of mathematical models in my final project, I would like to express my sincere gratitude to my supervisor Prof. Thomas K. Stidsen. I would like to thank him not only for his patience in answering my questions and instructing on my programming but also for his help in managing my master project schedule. He always makes me clear on what to do.

Since the day I was born, my parents have been supporting me incessantly for 25 years. They keep doing their best to foster me and I am provided with the best education physically, intellectually and mentally. They never prevent me from leaving to pursue my dreams, even though I am the only child in the family. We have not been living together for almost 7 years, and during this period, they have suffered from missing me, busy working and illness. All of these make me feel owing to them. My parents deserve my appreciation and respect all through my life, and I promise to be a good son.

Yao Liang

KGS.Lyngby
Denmark
May 2, 2006

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Optimization Objective

Telecommunication Networks have became indispensable in people's daily life and have always been changing people's mind. Telecommunication Networks provide numerous kinds of services, including telephone, community antenna television, internet, mobile phone and so on. The attempts to introduce networks into most fields of the society never stop, and people's demands of bandwidth keep growing. Therefore network design problems will keep being an important subject. In addition, the development of transmitting materials and routing schemes result in the improvement of design methods.

Several aspects need to be considered when designing Telecommunication Networks. However, in a single network, there should be an unique optimization objective. It could be minimizing the cost, minimizing the transmission time, minimizing the failure recovery time, maximizing the robustness and so on. Fixed-charge network models are basic and can describe the practical situations well. The emphasis of this thesis is minimizing the total cost of constructing a new network. The final solution consists of two parts, the fixed-charge and the cable laying cost.

In addition, more requirements, such as protection and network topologies, should be discussed in the models according to technical restriction. In a word, what this thesis focuses on is the fixed-charge network design models for

different topologies.

1.2 Thesis Outline

The reminder of the thesis is structured as follows:

In Chapter 2, the concept of graph model is introduced. The shape of five topologies, mesh, ring, tree, star and bus, are also presented. Moreover, a brief summary of network instances makes the following analysis easier.

The non-protective mesh network model is presented in Chapter 3. Afterwards, two protection schemes, 1+1 path protection method and link protection method, are implemented in mesh design. Furthermore, column generation algorithm is applied in link protection mesh model.

In Chapter 4, fixed-charge ring models, both non-protective and protective, are demonstrated. A simplified TSP model is furthermore introduced to replace the 1+1 protection ring model. And a special model of two-fibre unidirectional self-healing ring is presented.

In Chapter 5, fixed-charge tree, star and bus models are built. Since all these 3 topologies are for small scale networks, protection is not considered. Star topology is divided into two models according to the technical difference.

Some future works are listed in Chapter 6.

Chapter 2

Telecommunication Network Topologies

2.1 The Graph Model of Networks

Before mathematical models are built, we need to transfer the practical Telecommunication Networks, including components, locations and architectures, into *graphes* which can later be easily formulated. Something is ignored or simplified in this kind of transformation. The graph of a general Telecommunication Network is shown in fig. 2.1[3].

Nodes, links and paths are the three basic elements of Telecommunication Networks. A *node* is located in a position where requirement of bandwidth is concentrating, and a *link* is the straight connection between two nodes. Nodes need to communicate between each another. We denote DEMAND as a demand matrix, in which d_{kl} is the statistical quantity of flow in a certain time period from node k to node l.

A flow of data, i.e. packets in a packet switching network, goes from one node to another through a *path* which is a sequence of nodes. For example, in fig. 2.1, node A sends packets to node B through path $A \rightarrow C \rightarrow D \rightarrow B$, and correspondingly, node B sends packets to node A through an opposite directional path $B \rightarrow D \rightarrow C \rightarrow A$. All paths are directed. There might exist more than one paths connecting a nodes pair, e.g. two paths connecting A and

B are depicted in fig. 2.1.

In order to make sure that a communication path exists, capacities need to be reserved on some links. We denote $\text{SETUPCOST}_{(i,j)}$ as the cost, e.g. digging cost, to build link (i, j). It is a *fixed-charge* which relies on the distance, set up technology and so on. We also denote $\text{CABLEPRICE}^{c}_{(i,j)}$ as the cost to add a cable/fibre c, with certain CABLECAPACITY_c, on link (i, j).

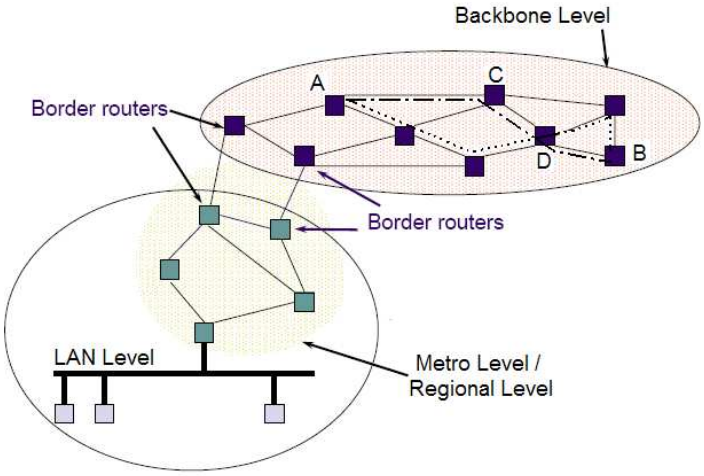


Figure 2.1. A General Telecommunication Network.

Physically, it is possible to have two-way communication on a single wire/cable.[1] This factor and the undirected feature of links result in the following principles (unless otherwise noted, we perform the simplification below for all network models):

- **Principle 2.1**: links are considered undirected, i.e. link (i, j) and all parameters on it equal link (j, i) and the relevant parameters.
- **Principle 2.2**: Flows and paths are directed.
- **Principle 2.3**: On every single link, reservation of capacity caused by different-directed flows are same.

[1]The only exception which we will introduce later is two-fibre unidirectional self-healing ring. In this topology, flows only transmit in one direction.

- **Principle 2.4**: We do not distinguish between d_{kl} and d_{lk}, and we simply add them up and result in a triangular demand matric.

However, considering the network hierarchy, nodes and links in different networks may represent different hardwares. For instance, nodes can be servers, switches/routers, multiplexers, de-multiplexers, add-drop multiplexers, terminals, telephones, personal computers and so on. And links can be coaxial lines, twisted-pair cables, fibres, wireless transmission path and so on. Typical equipments are usually related with typical network topologies. Therefore when we construct Telecommunication Networks, we need not only to consider the topologies, but also to take the effects of different hardwares into account.

2.2 Five Basic Network Topologies

"A network topology is the pattern of links connecting pairs of nodes of a network."[2] Therefore it is determined only by the configuration of connections between nodes. Five common network topologies, *mesh*, *ring*, *tree*, *star* and *bus* are studied in this thesis, and so are the hardwares equipped in each one of them. Fig. 2.2 demonstrates the affiliation of topologies.

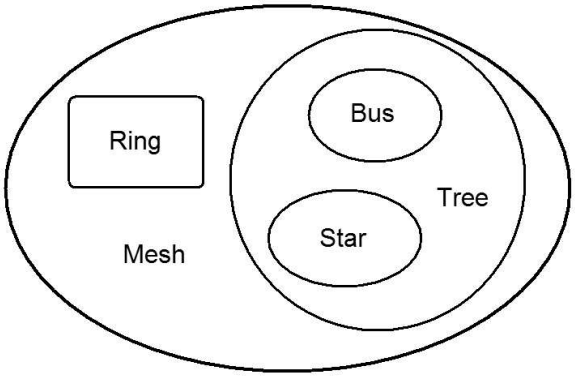


Figure 2.2. The Affiliation of Network Topologies.

It is not specific that one kind of network topology is only implemented in a certain occasion or scale. Telecommunication Networks are various and complicated, however, what we discussed below is just based on normal situations.

2.2.1 Backbone Network: Multiplexing Technology, Mesh and Ring Topologies

Mesh and ring topologies are widely applied in *backbone networks* which cover most area of nations. The nodes are switches/routers, and they are usually placed in big cities where the demands for bandwidth are huge. Correspondingly, the links are fibres[2] which provide huge capacity.

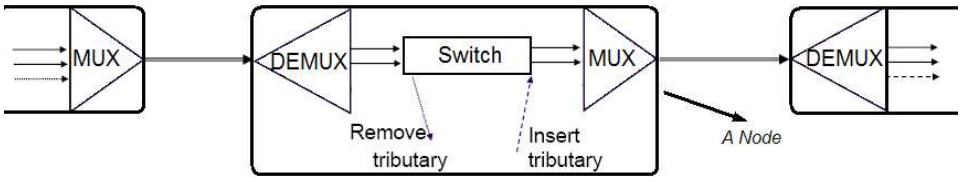


Figure 2.3. The Multiplexing Function of A Node with MUX and DEMUX Pair.

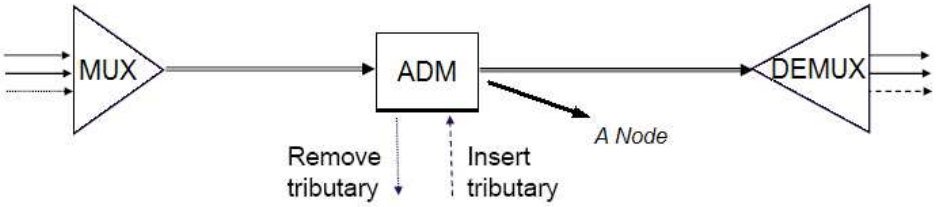


Figure 2.4. The Multiplexing Function of A Node with An ADM.

A backbone network must be able to support Tbit/s bandwidth transmission. A key factor to organize and manage such huge data flow is the application of *multiplexing* technology. Low-level user signals are multiplexed to high-level transport signal, so a pair of *multiplexer(MUX)* and *de-multiplexer(DEMUX)* are equipped at each node to take out or add in some tributaries. Moreover, the cost of the whole network would be reduced if we introduce a *add-drop multiplexer(ADM)* instead of the MUX DEMUX pair. An ADM can add and drop tributaries *without* slowing down the transmission of signals.

In a narrow sense, "there are at least two nodes with two or more paths between them in a mesh topology."[2] Whereas, in a broad sense, any kind of

[2]For backbone networks, fibre is gradually popular in the last decade. Sometimes wireless medium with large bandwidth, such as satellite and microwave communication, are used.

topology can be seen as mesh, and it makes mesh network design to be a basic problem.

Fig. 2.6 indicates a special kind of mesh topology, the *fully-connected mesh*. In this case, a link exists between all pairs of nodes, and there are totally $\frac{n(n-1)}{2}$ direct links. Fully-connected mesh is the most expensive network, have the most paths between every two nodes and makes the protection of network considerably simple. But usually people just construct ordinary mesh networks as demonstrated in fig. 2.5.

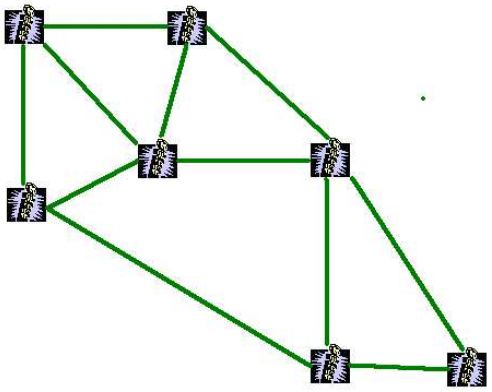


Figure 2.5. A Mesh Network.

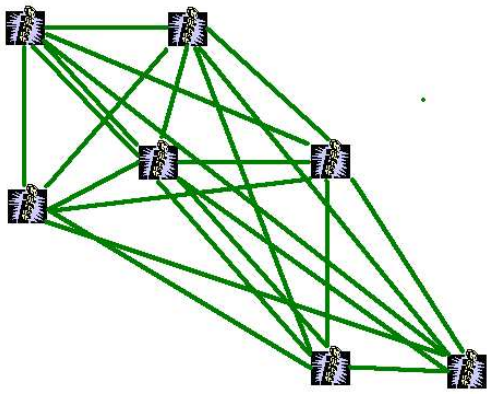


Figure 2.6. A Fully Connected Mesh Network.

In a ring network, each node is connected to two other nodes. An outstanding characteristic of ring is that, comparing with other topologies, data normally travels through more nodes before arriving its destination. For instance, data from node A has to pass at least one other node before reaching B in fig. 2.7. Meanwhile it is possible to add a new link AB if it was of mesh topology. This feature allows ring networks to span greater distances, but also make them only cheaper than fully-connected mesh. Furthermore, without protection, a failure of a single node or link may destroy the whole network.

We will discuss the protection of ring networks in Chapter 4, but here we introduce the structure of *two-fibre unidirectional self-healing ring*, which is described in fig. 2.8. One fibre is *active*, and only carries signals in one direction. From a technical point of view, unidirectional transmission brings high speed. Unlike the other topologies, we need to distinguish demand AB from

demand BA, since the previous one travels in path $A \to C \to ETC. \to B$ and the latter one goes in path $B \to A$. On the other hand, another fibre carries *protection* signals in the reverse direction. The protection signals would only be used when failure occur.

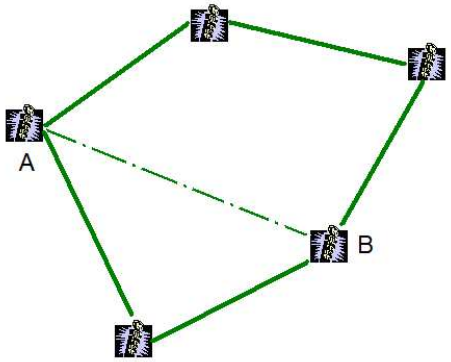


Figure 2.7. A Ring Network.

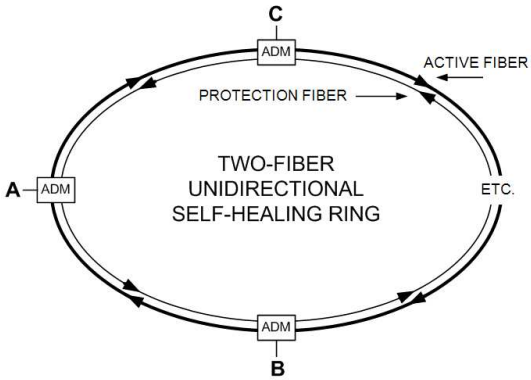


Figure 2.8. A Two-Fibre Unidirectional Self-Healing Ring Network.[4]

2.2.2 Regional Network and LAN: Tree, Star and Bus Topologies

According to the network hierarchy, nodes will be sorted when we design tree, star and bus topologies. One type is *hub*, *gateway* or *central* nodes which are the roots of every level in networks, and the other type is *terminal* or *peripheral* nodes.

Hubs have these functions: Since many network facilities, i.e. servers and switches, are expensive, we only equip them in limited-numbered hubs. Moreover, all terminals straightly-connected to a hub are controlled by it when they want to communicate with each others or with outside environment. And this achieves the hierarchical management of some big networks.

Tree topology is applicable in a *regional network* which connects several *local area networks(LAN)*. For example, an university may have a network as indicated in fig. 2.9. Node D and E are two terminals which connect their department's switch A. Their communication needs to pass through

A, but not necessarily through node B, G and the other nodes. Similarly, The communication between A and B has to pass through G. G is a border switch which probably accesses to a *metro area network(MAN)*, or an even bigger education network. It is also very possible that the principal in node C connects directly to G. If node D and F have demand between each other or with some nodes outside the tree, they need the permission of their respective roots and G.

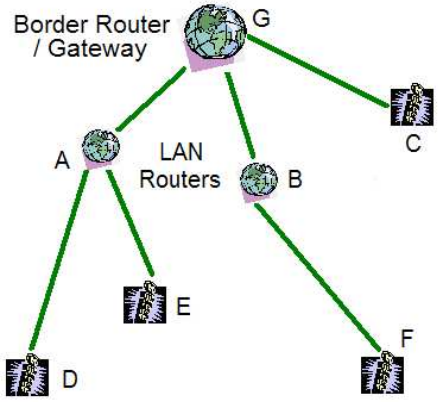


Figure 2.9. A Tree Network.

A tree with only two levels, i.e. one root and its "sons", is a star. Since all terminals only transmit data with the hub, a star network is easy to implement and extend, and the failure of any peripheral will not have effect on the whole network. We do not study bus-based star networks where hubs rebroadcast all transmission. Instead, we choose the local loop of telephone networks as an example to discuss the two models of star.

In fig. 2.10, all peripheral telephones connect straightly to hub A by twisted-pair cable. Though the cable from node D pass through node C, the two distribution cables, DA and CA, are independent and will just be packed into a feeder cable from C. So node C is not only a terminal telephone, but also a pedestal or a serving area interface where cables concentrate.

Nowadays, some feeder cables are substituted by fibres since users' demands of bandwidth are increasing. A MUX DEMUX pair must be placed at node

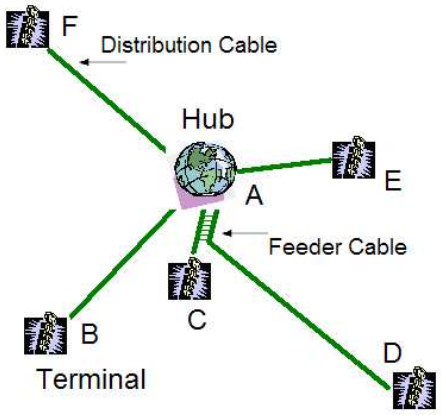


Figure 2.10. A Star Network without Multiplexing.

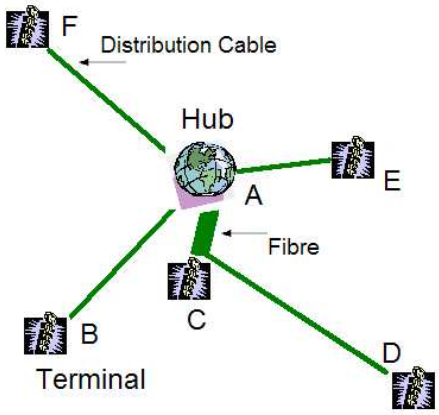


Figure 2.11. A Star Network with Multiplexing.

C as shown in fig. 2.11. The cost of MUX, DEMUX and fibre gets in return with the save of bandwidth. For example, if each terminal has a demand of 1 Mbit/s with outside while we only have distribution cables provide 1.55 Mbit/s bandwidth, we should provide a piece of cable for every single user. However, if a fibre provide 20 Gbit/s bandwidth, it carry data for 20,000 terminals while 11 Gbit/s bandwidth is saved. When the fibres extend to buildings which are very close to terminals, the coverage of twisted-pair cables will be greatly decreased. Two models of star topology are built in Chapter 5.

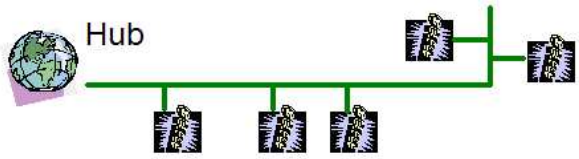


Figure 2.12. A Bus Network.

Terminals are connected by sharing a single cable with fixed capacity in a bus topology, and at one end of the cable is a hub. Bus network is easiest and cheapest, however, there are problems when two terminals want to transmit data at the same time.[3] Thus bus systems normally have scheme of collision

[3]This is the situation in LAN. On the other hand, we can surely handle the bandwidth problem by setting more cables in a backbone bus.

avoidance, but it is not a consideration of physical network construction.

2.2.3 A Brief Summary

What the Section 2.2.2 demonstrates is summed up in table 2.1. As mentioned before, we only discuss the common application of network topologies. In real life, Telecommunication Networks are much more complicated, and their whole structures are often the combination of the basic topologies.

Topology	Node	Link	Mostly Applied
Mesh	Switch, ADM	Fibre	Backbone network, MAN
Ring	Switch, ADM	Fibre	Backbone network, MAN
Tree	Switch, server, etc. (roots); PC, telephone, etc. (terminals)	Twisted-pair cable, coaxial-cable, fibre	Regional network
Star (no multiplexing)	Switch, server, etc. (hub); PC, telephone, etc. (terminals)	Twisted-pair cable, coaxial-cable	LAN, regional network
Star (multiplexing)	Switch, server, etc. (hub); telephone, PC, etc. (terminals); MUX, DEMUX, etc. (serving area interface)	Twisted-pair cable, coaxial-cable, fibre	LAN, regional network
Bus	Switch, server, etc. (hub); PC etc. (terminals)	Twisted-pair cable, coaxial-cable	LAN

Table 2.1. A Brief Summary of Network Topologies and Hardwares.

2.3 Network Instances and Statistical Features

Some operation researchers have provided a library of fixed-charge network design instances on web.[1] We chose three from the set, and made other two examples following the instances' pattern. They are used to test our mathematical models. All the five graph models are listed in Appendix A, however, those will appear frequently in later chapters are shown below:

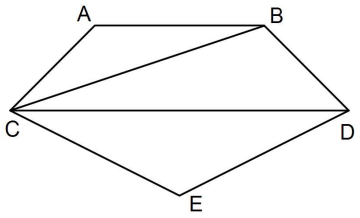


Figure 2.13. Arne Network with 5 Nodes and 7 Links.

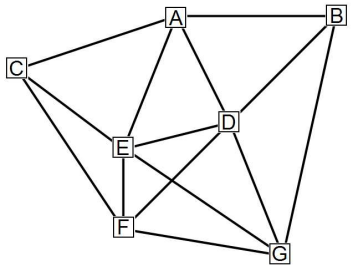


Figure 2.14. A Network of Germany with 7 Nodes and 14 Links.

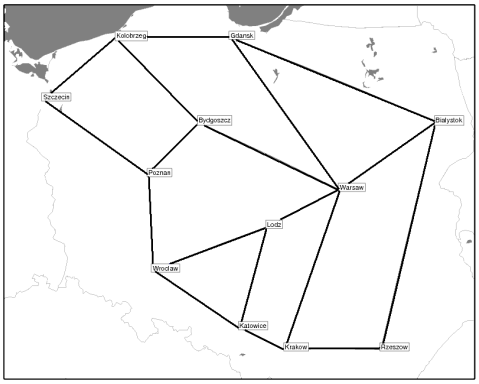


Figure 2.15. A Network of Poland with 12 Nodes and 18 Links.

Statistical features of the instances are important, and can, to some extent, explain the solutions. Table 2.2 lists the instances' details, and there are three main factors:

- We define *Average Node Degree* as the fraction of double links' number over nodes' number. It shows the number of nodes which a node connects to on average.

- Similarly, we define *Max Node Degree*. It tells the hot spot of a graph which is potentially the hub node when we build tree, star or bus network.
- On every link, we calculate the fraction of its fixed-charge over the cost of cable with least capacity. And *Fixed-Charge Weight* is defined as the mean value of the previous quantity. It tells which cost is dominant, the fixed-charge or the cable cost.

Details	Arne	Germany	Poland	France	Pioro
No. of Nodes	5	7	12	25	40
No. of Links	7	14	18	45	89
No. of Demands	10	21	66	300	780
Max Node Degree	4	5	5	10	5
Hot Nodes	C	D, E	$Warsaw$	$N15$	$N0$, $N3$, $N8$, etc.
Average Node Degree	2.8	4	3	3.6	4.45
Fixed-Charge Weight	0	5.17	1	10.86	1

Table 2.2. Instances Statistics.

Chapter 3

Mesh Network and Protection

Fig. 2.2 indicates that mesh category covers all the other network topologies, thus mesh network design is a basic problem. In fact, when we build models for other topologies, we just modify the mesh model and add more constraints.

3.1 Fixed-Charge Network Design

We describe the mathematical model for the *simplest* fixed-charge network design problem in this section. The topology is mesh, and there is no protection scheme.

3.1.1 Mathematical Model

There are three sets: N for nodes, L for links and C for cables. And the indices are:

i, j, k, l	:	Nodes in the network; i, j, k, $l \in N$
(i, j)	:	Links in the network; if $(i, j) \in L$, $(j, i) \in L$
c	:	Types of cables; $c \in C$

The parameters are:

$\text{SetupCost}_{(i,j)}$	:	The fixed-charge to establish link (i, j)
CableCapacity_c	:	The capacity that cable c can provide
$\text{CablePrice}^c_{(i,j)}$	:	The cost of adding a cable c on link (i, j)
Demand_{kl}	:	The statistical demand of data between node k and l

We perform the simplification based on principle 2.1 and 2.4. Thus on every single link, $\text{SETUPCOST}_{(i,j)}$ equals to $\text{SETUPCOST}_{(j,i)}$, and $\text{CABLEPRICE}^c_{(i,j)}$ equals to $\text{CABLEPRICE}^c_{(j,i)}$. We add up DEMAND_{kl} and DEMAND_{lk} to make an *upper triangular* matrix. Afterwards, only DEMAND_{kl}, $k < l$ are possibly nonzero. It will decrease the number of variables.

The model contains three types of variables: one continuous flow decision $x^{kl}_{(i,j)}$, one binary setup decision $y_{(i,j)}$ and another integer capacity decision $z^c_{(i,j)}$.

$x^{kl}_{(i,j)}$	:	The fraction of demand kl which flows on link (i,j) from node i to node j (principle 2.2)
$y_{(i,j)}$	:	Whether to set up link (i,j)
$z^c_{(i,j)}$	:	How many cables c are laid on link (i,j)

Finally, the *mixed integer programming(MIP)* model is:

Minimize:

$$\sum_{(i,j)} \sum_{c} z^c_{(i,j)} \times \text{CABLEPRICE}^c_{(i,j)} + \sum_{(i,j)} \text{SETUPCOST}_{(i,j)} \times y_{(i,j)} \tag{3.1}$$

Subject To:

$$\sum_j x^{kl}_{(i,j)} - \sum_j x^{kl}_{(j,i)} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{otherwise;} \\ -1 & \text{if } i = l \end{cases} \quad \forall i \in N \ ; \ k, l \in N \text{ and } \text{DEMAND}_{kl} \neq 0 \tag{3.2}$$

$$\sum_{kl} (x^{kl}_{(i,j)} + x^{kl}_{(j,i)}) \times \text{DEMAND}_{kl} \leq \sum_c z^c_{(i,j)} \times \text{CABLECAPACITY}_c \ ; \ (i,j) \in L \text{ and } i < j \tag{3.3}$$

$$x^{kl}_{(i,j)} \leq y_{(i,j)} \ ; \quad (i,j) \in L \text{ and } i < j \ ; \forall k, l \in N \tag{3.4}$$

$$x^{kl}_{(i,j)} \leq y_{(j,i)} \ ; \quad (i,j) \in L \text{ and } j < i \ ; \forall k, l \in N \tag{3.5}$$

$$x^{kl}_{(i,j)} = 0 \ ; \quad \forall (i,j) \in L \ ; k,l \in N \text{ and } \textsc{Demand}_{kl} = 0 \tag{3.6}$$

$$y_{(i,j)} = 0 \ ; \quad (i,j) \in L \text{ and } i > j \tag{3.7}$$

$$z^{c}_{(i,j)} = 0 \ ; \quad \forall c \in C \ ; (i,j) \in L \text{ and } i > j \tag{3.8}$$

$$x^{kl}_{(i,j)} \in \mathbb{R}^{+} \ ; \quad \forall (i,j) \in L \ ; \forall k,l \in N \tag{3.9}$$

$$y_{(i,j)} = 0 \text{ or } 1 \ ; \quad \forall (i,j) \in L \tag{3.10}$$

$$z^{c}_{(i,j)} \in \mathbb{Z}^{+} \ ; \quad \forall (i,j) \in L \ ; \forall c \in C \tag{3.11}$$

Eq. 3.2 describes the paths for every $\textsc{Demand}_{kl}$: Both the sum of fractional outward flows at the origin node k and the sum of fractional inward flows at the destination node l equal to 1. Meanwhile, at all the other nodes, flows for $\textsc{Demand}_{kl}$ only pass but not enter or drop. Eq. 3.3 ensures that enough cables will be laid on links, and it is contributing for the first part of objective, i.e. eq. 3.1. Eq. 3.4 and eq. 3.5 ensure that if there are flows going through link (i,j), we should at least set up link (i,j). These two equations are contributing for the second part of objective.

Since we do not need flows for zero demands, half of $x^{kl}_{(i,j)}$ are set to zero in eq. 3.6. Moreover, eq. 3.7 - eq. 3.8 are based upon principle 2.1 and principle 2.3. We consider either $y_{(i,j)}$ or $y_{(j,i)}$ and either $z^{c}_{(i,j)}$ or $z^{c}_{(j,i)}$, thus set all $y_{(i,j)}$ and $z^{c}_{(i,j)}$, $(i,j) \in L$, $i > j$ to zero.

3.1.2 Comments and Solutions

The objective consists of two parts, the cable cost and the fixed-charge. If the fixed-charge is much larger than the cable cost, the solution would set up links as fewer as possible. When cable cost is zero, the problem becomes a exact *minimum spanning tree(MST)* problem. Contrarily, if the cable cost is dominating, we will just concern more on routing the flow to use cables

as fewer as possible. When the setup cost is zero, the problem becomes an approximative *all-to-all shortest path* problem. The solutions(which links are set up) for Germany and Poland networks are shown below, the instances' details are recorded in table 2.2:

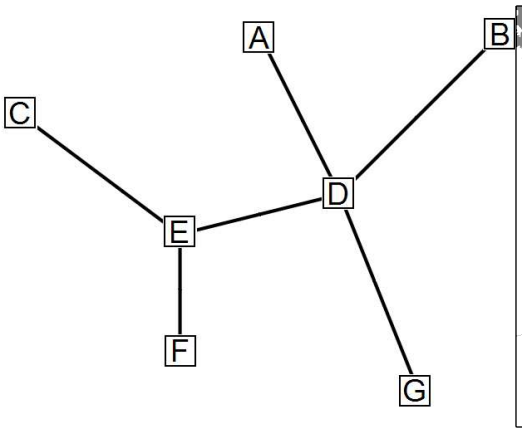


Figure 3.1. Mesh Solution of Germany Network. The Fixed-Charge is Dominating.

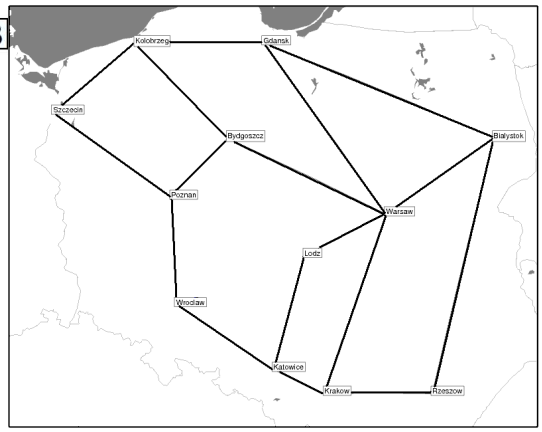


Figure 3.2. Mesh Solution of Poland Network. The Cable Cost is Dominating.

If we eliminated the integer variable $z^c_{(i,j)}$ by introducing a continuous parameter, *cost per unit capacity*, the problem described by eq. 3.1 - eq. 3.11 would be much easier. There would be no otiose capacities in the solutions. Moreover, we will have an exact *all-to-all shortest path* problem when setup costs are zero.

Nevertheless, the actual problem is hard. The solution is not only measuring the weights between fixed-charge and cable cost, but is also trying to route the flows precisely to deplete cable capacities.

Table 3.1 shows the running time of GAMS programs and the MIP solutions for 4 network instances. Within the time limitation, 270,000 seconds(almost 3 days), the *CPLEX* solver could not solve Pioro network completely. However, it provides a solution with 0.63% relative gap away from the best possible lower bound.

	Germany	Poland	France	Pioro
No. of Nodes	7	12	25	40
No. of Links	14	18	45	89
Running Time	2 s	32 m 20 s	14 h 40 m	75 h
Obj. Value	22650	27691	52775	430680 (Best Possible: 427988.2)

Table 3.1. Running Time and Objective Values for Mesh Networks.

3.2 1+1 Protection for Mesh Network

No matter how much it cost, we should guarantee that networks are running continuous. The network operators surely realize the huge amount of money that they invest to construct such networks, but they just do not want to take the consequences of losing clients' data. The importance of network reliability and the increasingly cheaper fibre price make *protection* necessary for all Telecommunication Networks.

The protection schemes discussed in this thesis only aim at single link failures. What a protection method does is to ensure that data transmitted on the failed link will be transmitted through other links without troubling other data. Thus extra protective bandwidth should be reserved aforehand. The research of protection is based on how to reserve protective capacities. There are mainly two protection methods, the *path protection* scheme and the *link protection* scheme. An kind of link protection method is studied in Section 3.3, and a widely used path protection method, *1+1 protection*, is accomplished in this section.

3.2.1 1+1 Protection

Just as its name implies, path protection scheme tries to protect link failures by protecting all flows of different paths on that link. Moreover, a path flow contains a series of link flows as demonstrated in eq. 3.2. Since we do not know which link would fail, we need to protect all links for a path.

In 1+1 protection, at least two paths[1] that *do not share any link* are provided for each demand as indicated in fig. 3.3[5]. One is called the primary path, and the alternate one is called the backup path. Signals are transmitted in both paths, and in normal status, the destination node just receives data by primary path. When a link failure occurs on the primary path, the destination will just switch to the backup path. Thus 1+1 protection scheme has a fast recovering time.

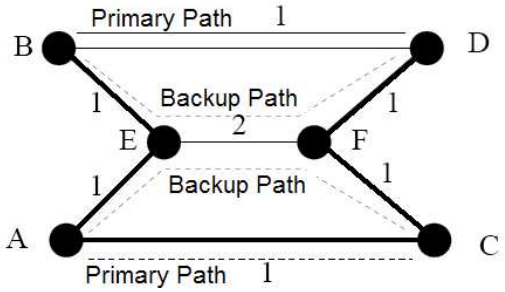


Figure 3.3. How 1+1 Protection Works: There are 1 unit demand for BD and AC respectively. Path $B \to D$ and $A \to C$ are primary paths. Path $B \to E \to F \to D$ and $A \to E \to F \to C$ are backup paths. All numbers in the figure show the capacities that reserved on links.

In the basic fixed-charge network design problem, the solution paths for DEMAND_{kl} are possibly cheapest. Therefore the new alternate paths must be more expensive, and this characteristic makes 1+1 protection "requires more than 100% and most often more than 130% extra capacity".[6] In a word, it is a fast but also an expensive protection method.

3.2.2 Mathematical Model

Compared with the data in previous model, the only difference is the two continuous flow decision variables, $xa^{kl}_{(i,j)}$ for primary path and $xb^{kl}_{(i,j)}$ for backup path.

$xa^{kl}_{(i,j)}$: The fraction of demand kl which flows on link (i,j) from node i to node j, primary path

$xb^{kl}_{(i,j)}$: The fraction of demand kl which flows on link (i,j) from node i to node j, backup path

[1] We may already have more than one primary path for a single demand.

And the MIP model for 1+1 protection mesh network is:

Minimize:

$$\sum_{(i,j)}\sum_{c} z^{c}_{(i,j)} \times \textsc{CablePrice}^{c}_{(i,j)} + \sum_{(i,j)} \textsc{SetupCost}_{(i,j)} \times y_{(i,j)} \tag{3.12}$$

Subject To:

$$\sum_{j} xa^{kl}_{(i,j)} - \sum_{j} xa^{kl}_{(j,i)} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{otherwise;} \\ -1 & \text{if } i = l \end{cases} \quad \forall i \in N \ ; \ k,l \in N \text{ and } \textsc{Demand}_{kl} \neq 0 \tag{3.13}$$

$$\sum_{j} xb^{kl}_{(i,j)} - \sum_{j} xb^{kl}_{(j,i)} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{otherwise;} \\ -1 & \text{if } i = l \end{cases} \quad \forall i \in N \ ; \ k,l \in N \text{ and } \textsc{Demand}_{kl} \neq 0 \tag{3.14}$$

$$\begin{gathered} \textstyle\sum_{kl}(xa^{kl}_{(i,j)} + xa^{kl}_{(j,i)} + xb^{kl}_{(i,j)} + xb^{kl}_{(j,i)}) \times \textsc{Demand}_{kl} \\ \leq \textstyle\sum_{c} z^{c}_{(i,j)} \times \textsc{CableCapacity}_{c} \ ; \ (i,j) \in L \text{ and } i < j \end{gathered} \tag{3.15}$$

$$xa^{kl}_{(i,j)} + xa^{kl}_{(j,i)} + xb^{kl}_{(i,j)} + xb^{kl}_{(j,i)} \leq 1 \ ; \quad \forall (i,j) \in L \ ; \forall k,l \in N \tag{3.16}$$

$$xa^{kl}_{(i,j)} \leq y_{(i,j)} \ ; \quad (i,j) \in L \text{ and } i < j \ ; \forall k,l \in N \tag{3.17}$$

$$xa^{kl}_{(i,j)} \leq y_{(j,i)} \ ; \quad (i,j) \in L \text{ and } j < i \ ; \forall k,l \in N \tag{3.18}$$

$$xb^{kl}_{(i,j)} \leq y_{(i,j)} \ ; \quad (i,j) \in L \text{ and } i < j \ ; \forall k,l \in N \tag{3.19}$$

$$xb^{kl}_{(i,j)} \leq y_{(j,i)} \ ; \quad (i,j) \in L \text{ and } j < i \ ; \forall k,l \in N \tag{3.20}$$

$$xa^{kl}_{(i,j)} = 0 \ ; \quad \forall (i,j) \in L \ ; \ k,l \in N \text{ and } \textsc{Demand}_{kl} = 0 \tag{3.21}$$

$$xb^{kl}_{(i,j)} = 0 \ ; \quad \forall (i,j) \in L \ ; \ k,l \in N \text{ and } \textsc{Demand}_{kl} = 0 \tag{3.22}$$

$$y_{(i,j)} = 0 \ ; \quad (i,j) \in L \text{ and } i > j \tag{3.23}$$

$$z^{c}_{(i,j)} = 0 \ ; \quad \forall c \in C \ ; \ (i,j) \in L \text{ and } i > j \tag{3.24}$$

$$xa^{kl}_{(i,j)} \ , \ xb^{kl}_{(i,j)} \in \mathbb{R}^{+} \ ; \quad \forall (i,j) \in L \ ; \forall k,l \in N \tag{3.25}$$

$$y_{(i,j)} = 0 \text{ or } 1 \ ; \quad \forall (i,j) \in L \tag{3.26}$$

$$z^{c}_{(i,j)} \in \mathbb{Z}^{+} \ ; \quad \forall (i,j) \in L \ ; \forall c \in C \tag{3.27}$$

Actually, $xa^{kl}_{(i,j)}$ and $xb^{kl}_{(i,j)}$ are the same type of variables, so eq. 3.15 shows that they both contribute to the reservation of cable capacities. However, eq. 3.16 ensures that at most 100% of flow pass through link(i,j) for every $\textsc{Demand}_{kl}$, no matter whether it is primary flow or backup flow. I.e. the two paths do not share any link.

All other equations have the same functions as those of eq. 3.1 - eq. 3.11.

3.2.3 Comments and Solutions

Since at least two disjunct paths have to be found, each node must be the ends for at least two links, and this factor makes solutions to set up more links. It may brings high setup cost in those fixed-charge-dominating networks, e.g. the Germany network. The solutions(which links are set up) for Germany and Poland networks are shown in fig. 3.4 and fig. 3.5. The primary and backup paths for $\textsc{Demand}_{'D''F'}$ and $\textsc{Demand}_{'Szczecin''Warsaw'}$ in their respective networks are indicated.

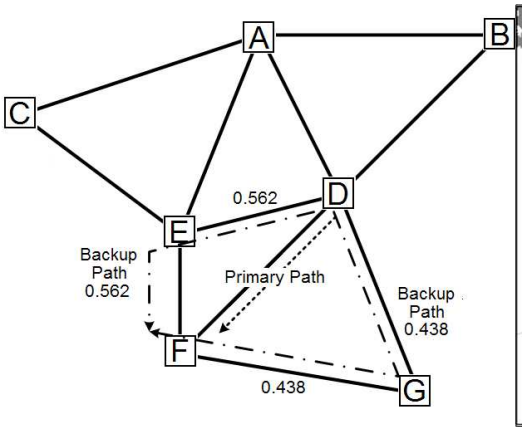


Figure 3.4. 1+1 Mesh Solution for Germany Network: The primary path for DEMAND$'_{D''F'}$ is $D \rightarrow F$. The protection paths for DEMAND$'_{D''F'}$ are $D \rightarrow E \rightarrow F$ with 0.562 of demand and $D \rightarrow G \rightarrow F$ with 0.438 of demand.

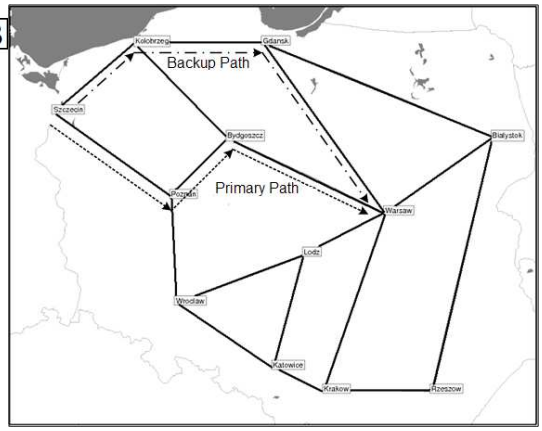


Figure 3.5. 1+1 Mesh Solution for Poland Network: The primary path for DEMAND$'_{Szczecin''Warsaw'}$ is $Szczecin \rightarrow Poznan \rightarrow Bydgoszcz \rightarrow Warsaw$. The protection path is $Szczecin \rightarrow Kolobrzeg \rightarrow Gdansk \rightarrow Warsaw$.

Compared with the basic problem, 1+1 protection mesh is harder to solve because there are more variables. Table 3.2 shows the running time of GAMS programs, the MIP solutions and the relative rise of objective values compared with the previous problem. Pioro network can still not be solved completely, and neither can France network because of the limitation of resource. The relative gap of Pioro network solution is 0.36%, and the relative gap of France network is 0.70%.

	Germany	Poland	France	Pioro
No. of Nodes	7	12	25	40
Running Time	12 s	10 m 36 s	75 h	75 h
Obj. Value	46090	62757	112688 (Best Possible: 111897.6)	981042 (Best Possible: 977531.6)
Rise in Obj.	103.5%	126.6%	113.5%	127.8 %

Table 3.2. Running Time and Objective Values for 1+1 Mesh Networks.

3.3 Link Protection for Mesh Network

1+1 protection scheme is expensive , and the problems are so hard to be solved that we need to find another protection method and build another model. A link protection scheme for mesh network is introduced in this section, and we use a 2-level solving method, *column generation*, to handle the new problem.

3.3.1 A Link Protection Method

1+1 protection provides primary paths and backup paths. In fact, if a failure occurs on an individual link, all the paths going through this link will break down. However, though the other links on these paths are still good, we leave capacities reserved for paths on the healthy links unused. It results in a kind of waste, and it is a disadvantage of 1+1 protection.

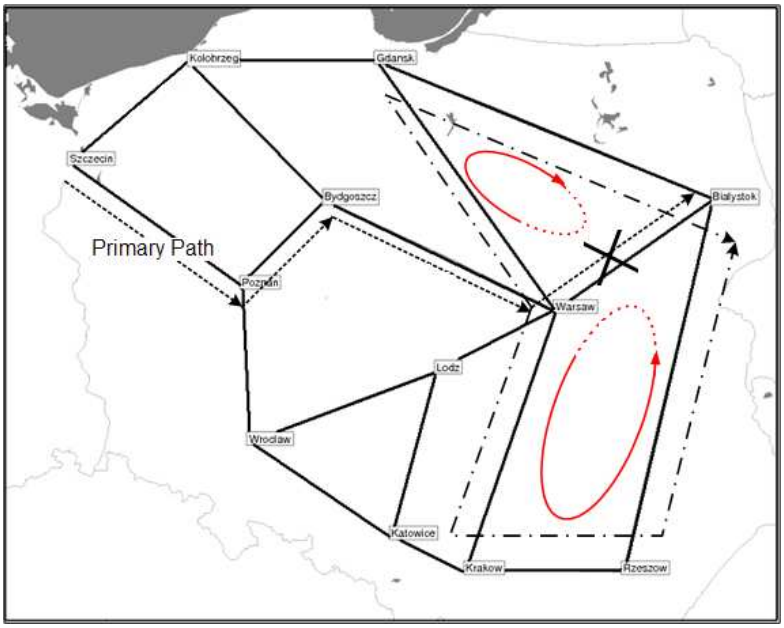


Figure 3.6. Link Protection Scheme.

In link protection schemes, we protect single links without concerning the other links which are on the same paths. One way to accomplish that is to find one or more paths which connects the two ends of the failed link and let the data goes through them instead.

Fig. 3.6 demonstrates an example: We only consider the communication for $\text{DEMAND}_{'Szczecin','Bialystok'}$ and assume that link $('Warsaw','Bialystok')$ fails. In 1+1 protection, a backup path would be used, and the capacities reserved for $\text{DEMAND}_{'Szczecin','Bialystok'}$ on link $('Szczecin','Poznan')$, $('Poznan','Bydgoszcz')$ and $('Budgoszcz','Warsaw')$ would be useless. On the other hand, in link protection, we still use the first three links on the path. In addition, we find other paths, e.g. path $Warsaw \rightarrow Gdansk \rightarrow Bialystok$, to transmit data instead of $Warsaw \rightarrow Bialysrok$.

These link-protection paths and the failed link would constitute sub-rings[2], e.g. the red circles depicted in fig. 3.6. Actually, the $Warsaw \leftrightarrow Gdansk \leftrightarrow Bialystok$ sub-ring can be used for the protection of three links, $('Warsaw','Gdansk')$, $('Warsaw','Bialystok')$ and $('Bialystok','Gdansk')$. Similarly, if we reserve capacities on other more sub-rings, it is possible to guarantee the protection of every single link in the whole mesh network.

Now the problem is: With a primarily designed network at hand, how to discover enough cheap sub-rings and how to reserve capacities on them only for protection usage? It is the mission of building a link protection model.

3.3.2 Mathematical Model

The new model we discuss in this section is reserving protection bandwidth, and the precondition is that a working network with primary capacities is given. For the primary networks, the solutions of basic fixed-charge network design problems can be used. Or we may simply perform all-to-all shortest path algorithms[3], e.g. *Floyd-Warshall* algorithm, to obtain such networks. However, MST algorithms can not be applied, since links jointing the bottom nodes of the tree can not be protect by any sub-rings. We do not set up additional links in the new model.

As mentioned before, our intention now is to discover sufficient cheap sub-rings. In fact, all the sub-rings of a given network are determinate. Thus

[2]We call it sub-ring to distinguish with ring topology.

[3]Ignore the discrete cable capacities.

comparison of sub-rings instead of searching is implemented. And we get a new set and a new index, R and r for sub-rings.

There is also some change in parameters:

$\text{PATTERN}^r_{(i,j)}$: Whether sub-ring r contains link (i,j); 1 if contains, otherwise 0; $\text{PATTERN}^r_{(i,j)} = \text{PATTERN}^r_{(j,i)}$

CABLECAPACITY_c : The capacity that cable c can provide

$\text{CABLEPRICE}^c_{(i,j)}$: The cost of adding a cable c on link (i,j); $\text{CABLEPRICE}^c_{(i,j)} = \text{CABLEPRICE}^c_{(j,i)}$

$\text{FLOW}_{(i,j)}$: The primary flow of data going through link (i,j); $\text{FLOW}_{(i,j)} = \text{FLOW}_{(j,i)}$

The continuous flow decision x_r is endowed with new meaning as well: The bandwidth reserved for sub-ring r. MIP model for protection sub-rings is:

Minimize:

$$\frac{1}{2}\sum_{(i,j)}\sum_{c} z^c_{(i,j)} \times \text{CABLEPRICE}^c_{(i,j)} \tag{3.28}$$

Subject To:

$$\sum_{r} x_r \times \text{PATTERN}^r_{(i,j)} \geq \text{FLOW}_{(i,j)} \ ; \quad \forall (i,j) \in L \tag{3.29}$$

$$\sum_{r} x_r \times \text{PATTERN}^r_{(i,j)} \leq \sum_{c} z^c_{(i,j)} \times \text{CABLECAPACITY}_c \ ; \forall (i,j) \in L \tag{3.30}$$

$$x_r \in \mathbb{R}^+ \ ; \quad \forall r \in R \tag{3.31}$$

$$z^c_{(i,j)} \in \mathbb{Z}^+ \ ; \quad \forall (i,j) \in L \ ; \forall c \in C \tag{3.32}$$

Eq. 3.29 ensures that on every link (i,j), the sum of protection bandwidth provided by different sub-rings is greater or at least equal to the primary bandwidth. Though the left-hand-side of eq. 3.29 and eq. 3.30 are same, we can not combine them together to eliminate x_r. Because x_r is the only variable to keep link protection feature, i.e. the existence of sub-rings.

A disadvantage of the new model is that we have to list all sub-rings of a given network. If a primary network is a fully-connected mesh and has n nodes, there would be $\sum_{m=1}^{n} \frac{n!}{m!(n-m)!}$ sub-rings![4] Whereas, eq. 3.1 - eq. 3.11 solve problems for $O(n^4)$ number of variables. Therefore link protection model may break down for huge amount of variables.

On the other hand, link protection model contains fewer constraints($O(n^2)$) than the basic network design problem($O(n^4)$). It implies that column generation method could be applied.

3.3.3 Column Generation

Column Generation is a 2-level method for solving *Linear Programming(LP)* problems with many columns(variables). A generic structure of LP problems is

$$\begin{array}{ll} Minimize & c^T x \\ s.t.: & Ax \leq b \\ & x \geq 0 \end{array} \tag{3.33}$$

where A is a $m \times n$ matrix. When n is much larger than m, linear algebra theory indicates that setting all non-basic variables to zero and solving the basis can give a feasible solution. The structure is shown below:

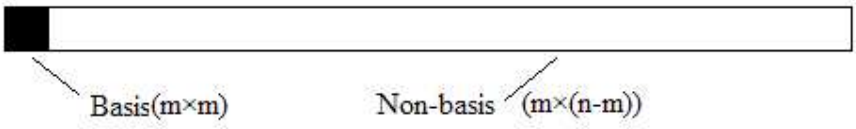


Figure 3.7. The Structure of Basis and Non-basis.

In a 2-level representation of LP problems, suppose now a basic solution is found. Looking for a possible better solution, a column from the non-basis is added to the basis. Hence, the (master) problem in each iteration is

[4] $\frac{n!}{m!(n-m)!}$ is the number of sub-rings which contain m links(nodes).

$$
\begin{aligned}
Minimize \quad & \begin{bmatrix} c' \\ c_p \end{bmatrix}^T \begin{bmatrix} x' \\ x_p \end{bmatrix} \\
s.t. : \quad & [A', A_p] \begin{bmatrix} c' \\ c_p \end{bmatrix} \leq b \\
& \begin{bmatrix} c' \\ c_p \end{bmatrix} \geq 0
\end{aligned} \tag{3.34}
$$

where A' is composed of a subset of A, and c' and x' are chosen correspondingly. A_p is a new-added column and x_p is a new-added variable.

Let π be a *multiplier vector* associated with the current basis, and it is usually an array of *constraints' dual values*. Applying column generation method, the master problems sends π to the sub-problems. The task of the sub-problems is to generate a new column A_p *as it is needed* with a minimal *reduced cost*, $c_p - \pi A_p$, and introduce the column to the master problem. When it is determined that there is no more profitable column to submit, the current solution is optimal.

The vector π and reduced cost have different sense in various problems. So is the method of generating a new column.

3.3.4 Modification of Mathematical Model and The Master Problem

Column generation is only applied in LP problems. We need to remove eq. 3.30, since it contains integer variables and can not provide dual values. This remind us of the old imagination: Can we introduce a continuous parameter, i.e. cost per unit capacity?

Statistical data demonstrates that prices and capacities of cables are not in a linear relationship. For example, all the 5 network instances have at most two types of cables(electrical signals or optical carriers), one for 155.52 Mbit/s

STM-1 signals and the other for 622.08 Mbit/s STM-4 signals.[5] On each link, the price of STM-1 is one third of STM-4's, meanwhile its capacity is one forth of STM-4's. Therefore the solution normally use 622.08 Mbit/s cables as more as possible and set 155.52 Mbit/s cables for the remainder. This makes the simplification possible:

- It would not waste much money and bandwidth if STM-4 signal cables are the only used cables.
- Dividing the prices of STM-4 signal cables by its bandwidth, we can get a rough parameter, cost per unit capacity.

Assuming that parameter $\textsc{UnitCost}_{(i,j)}$, the cost per unit bandwidth, is available at hand, we can further introduce another parameter $\textsc{PatternCost}_r = \frac{1}{2}\sum_{(i,j)} \textsc{UnitCost}_{(i,j)} \times \textsc{Pattern}^r_{(i,j)}$. It is the unit capacity cost of sub-ring r. The integer capacity decision $z^c_{(i,j)}$ is removed, and the modified LP model for link protection is:

Minimize:

$$\sum_{r \in R} \textsc{PatternCost}_r \times x_r \tag{3.35}$$

Subject To:

$$\sum_{r \in R} x_r \times \textsc{Pattern}^r_{(i,j)} \geq \textsc{Flow}_{(i,j)} \ ; \quad \forall (i,j) \in L \tag{3.36}$$

$$x_r \in \mathbb{R}^+ \ ; \quad \forall r \in R \tag{3.37}$$

x_r is the only continuous variable. The precondition of column generation is that we have a basis A' and the corresponding variables x', but not all of them. So in every iteration, the master problem of link protection model is substituting set R in eq. 3.35 - eq. 3.37 by its subset R'. Then the goal is to generate new columns A_p and x_p one by one.

[5] *STM-n*, *synchronous transfer modules-n*, are levels of transmitted signals in *SDH*, *synchronous digital hierarchy*.

A *dummy pattern* could be used as the basis at the beginning of every master problem. As its name implies, a dummy pattern could not be found in real life, but it could be recognized by the master problem. Furthermore, it helps in the way that we are not necessary to list even one pattern. All sub-rings are left to be generated by the model. A dummy sub-ring in this problem is a "ring" which contains all the links and suffering from its immense pattern cost. Anyway, it would soon disappear.

3.3.5 The Sub-problems

In the master problem, each constraint's dual value, $\pi_{(i,j)}$, represents *the currently most expensive pattern which contains link* (i, j). If a unit flow on link (i, j) is taken by another new sub-ring p, $\pi_{(i,j)}$ would be saved. Similarly the new sub-ring p would totally save $\sum_{(i,j)} \textsc{Pattern}^{p}_{(i,j)} \times \pi_{(i,j)}$. On the other hand, we pay $\textsc{PatternCost}_p$ for that sub-ring. Hence, to decide whether a column in the non-basis should be added is to decide *whether there exists a profitable new sub-ring.*

In a sub-problem, the most profitable sub-ring is generated by inspecting $\textsc{PatternCost}_p - \sum_{(i,j)} \textsc{Pattern}^{p}_{(i,j)} \times \pi_{(i,j)}$. The termination criterion is: if the sub-problem solution is non-positive, there still exists a cheaper sub-ring. Contrarily, no pattern should be generated and the optimal solution is found. Thus the *integer programming(IP)* model of sub-problem is:

Minimize:

$$\sum_{(i,j)} (\textsc{UnitCost}_{(i,j)} - \pi_{(i,j)}) \times y(i,j) \tag{3.38}$$

Subject To:

$$\sum_{j} y(i,j) - \sum_{j} y(j,i) = 0 \ ; \quad \forall i \in N \tag{3.39}$$

$$y(i,j) + y(j,i) \leq 1 \ ; \quad \forall (i,j) \in L \tag{3.40}$$

$$y_{(i,j)} = 0 \text{ or } 1 \ ; \quad \forall (i,j) \in L \tag{3.41}$$

The binary setup decision $y_{(i,j)}$ is the only variable in sub-problems. It represents whether the solution sub-ring contains arc (i, j). Eq. 3.39 generate a sub-ring, and eq. 3.40 ensures that the two arcs on a same link will not be considered as a pattern.

In fact, the solution might generate several sub-rings, since eq. 3.39 does not restrict the number of rings. It means that these rings are *equally profitable*, and they can together be considered as one pattern. Moreover, eq. 3.40 also makes the sub-ring to be directed, and we need to keep the symmetry of patterns before passing them back to the master problems.

3.3.6 Solving Process, Solutions and Comments

Fig. 3.8 demonstrates the solving process of Arne network as an example. It last 6 iterations, and 5 new sub-rings were generated. The dummy basis was abandoned after the third iteration.

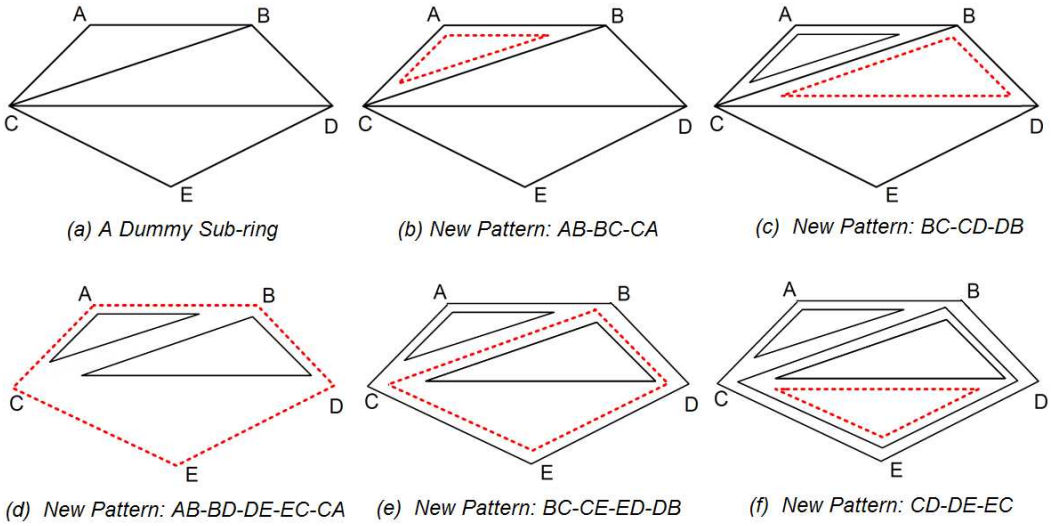


Figure 3.8. Solving Process of Arne Network.

Link protection problems are much easier to be solved by column generation method. Firstly, listing and inspecting of expatiatory columns is avoided. Secondly, integer capacity decisions are replaced by continuous capacity cost parameters. Table 3.3 shows the running time of GAMS programs. Number

of generated sub-rings is number of iterations minus 1. Since the link flow parameter and the unit capacity costs parameter are approximate, we do not list the objective values of the last master problems.

	Arne	Germany	Poland	France	Pioro
No. of Nodes	5	7	12	25	40
Running Time	<1 s	4 s	4 s	11 s	5 m 7 s
No. of Iterations	6	16	17	36	99
No. of Generated Patterns	5	15	16	35	98

Table 3.3. Running Time and Number of Iterations for Link Protection Mesh Networks.

In Appendix B, a simple example illustrates that link protection is usually more economical than 1+1 protection. The saving of capacity derives from the fact that sub-rings which consist of nearby links are normally *shorter* than alternate backup paths.

Chapter 4

Ring Network and Protection

Ring topology is significant, and it is broadly used in various network levels. However, in this chapter, we only design large scale ring networks, i.e. backbone networks and metro area networks. Ring topology for local area networks, e.g. token ring scheme, is not studied.

4.1 Non-Protective Ring Network Design

The importance of ring relies on the factor that this topology is compatible with network protection problems. It allocates additional capacities on all links, and data is just transmitted through the other links when a single link breaks down. Anyway, we start from the design of basic ring networks, i.e. rings without protection.

4.1.1 Mathematical Model

Ring topology can be considered as a subset of mesh topology. Hence, what we need to do when designing rings is to add more constraints to the mesh model. Considering the MIP model of basic fixed-charge network, eq. 3.1 - eq. 3.11, only one more constraint is appended:

$$\sum_j y_{(i,j)} + \sum_j y_{(j,i)} = 2 \ ; \quad \forall i \in N \tag{4.1}$$

Eq. 4.1 indicates that every node is the ends of two links and joints to exact two other nodes. Actually, it is possible to result in sub-rings. Nevertheless, eq. 3.2 shows that the existence of demands lead to the paths between every nodes pair. And the paths rush out of sub-rings and generate only one ring.

All sets, indices, parameters and variables are same as those of the basic fixed-charge network model, so they are not specified. Neither are the other constraints.

4.1.2 Comments and Solutions

The fixed-charge network design model is a *relaxation* of the non-protective ring model, since the latter one has an extra constraint. According to the ring characteristic, number of setup links must be equal to the number of nodes. Therefore the objective values of latter problems must be larger, especially for those cable cost dominating networks which have to reserve more expensive cables.

Table 4.1 supports the comments in the last paragraph. Rise in objective values are compared with solutions in table 3.1. France network has no integer solution, since no ring could be found based on the current network. Pioro network can even not be solved within 270,000 seconds.

	Germany	Poland	France	Pioro
No. of Nodes	7	12	25	40
Running Time	5 s	86 m 29 s	5 s	75 h
Obj. Value	28210	38911	No Integer Solution	Not Solved
Rise in Obj.	24.5%	40.5%	No Integer Solution	Not Solved

Table 4.1. Running Time and Objective Values for Non-Protective Ring Networks.

The non-protective ring problem is almost as hard as the basic fixed-charge network problem. Solutions(which links are set up) for Germany and Poland networks are illustrated in fig. 4.1 and fig. 4.2:

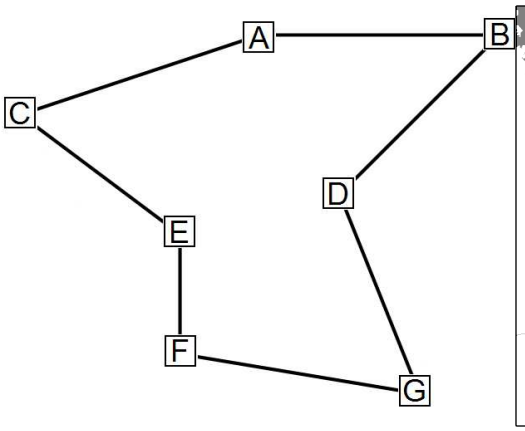


Figure 4.1. Non-Protective Ring Solution of Germany Network. (Also The Solution of 1+1 Protection Model and Two-Fibre Unidirectional Self-Healing Ring Model.)

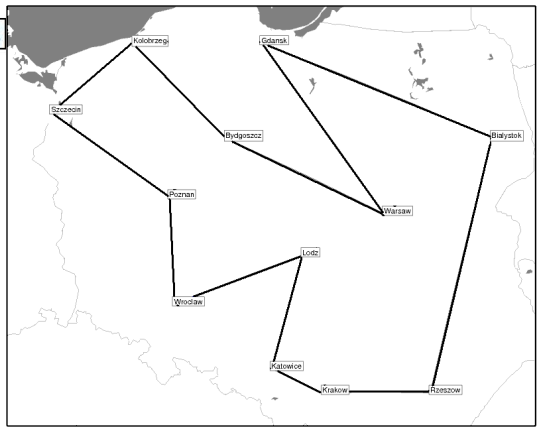


Figure 4.2. Non-Protective Ring Solution of Poland Network. (Also The Solution of 1+1 Protection Model and Two-Fibre Unidirectional Self-Healing Ring Model.)

4.2 1+1 Protection for Ring Network

Protection of backbone networks must be guaranteed. If link protection scheme is performed, the unique generated pattern is the original ring. Thus path and link protection methods result in the same solution to ring topology. And we only apply 1+1 method for ring networks.

4.2.1 Mathematical Model, Solutions and Comments

As the same idea in non-protective ring network design, the MIP model of 1+1 protection ring consists of 1+1 mesh model and a ring constraint, i.e. eq. 3.12 - eq. 3.27 plus eq. 4.1. Sets, indices, parameters and variables are not changed. This problem is also hard to be solved. Solution of setup links for Germany and Poland networks keep unaltered as the previous problem.

There is a distinct feature of ring topology: Every nodes pair have *exactly two paths* connecting them. The two paths do not share any link, and meanwhile they contain all links in the ring together.

Hence, 1+1 protection ring is much more expensive than non-protective ring. In any case, 100% flow of a demand travel in the two paths respectively, and the backup path might be much longer than the primary path. In those cable cost dominating networks, e.g. Poland network, long backup paths are inevitably costly. Comparison of objective values is shown in table 4.3.

	Germany	Poland	France	Pioro
No. of Nodes	7	12	25	40
Running Time	7 s	28 s	12 s	75 h
Obj. Value	62790	133280	No Integer Solution	Not Solved

Table 4.2. Running Time and Objective Values for 1+1 Protection Ring Networks.

In addition, 100% flow of every demand goes through every link on the ring. In other words, certain numbers of cables would be laid if a link is decided to be set up, because the allocated capacity on a link equals to the total demands. This factor is proved by the values of integer capacity decision, $z^c_{(i,j)}$, in solution.

$$z^c_{(i1,j1)} = z^c_{(i2,j2)} \ ; \quad (i1,j1),\ (i2,j2) \in L \text{ and } i1 < j1,\ i2 < j2 \tag{4.2}$$

Therefore we can define new fixed costs on all links by adding their former setup costs and the determinate capacity costs. Now the problem is simplified to a *travelling salesman problem(TSP)*.

4.2.2 An Improved Mathematical Model: TSP Problem

Given a collection of cities and the cost of travelling between each pair of them, the TSP problem is to find the cheapest way of visiting all the cities and returning to the starting point. Normally the formulation is:

Minimize:

$$\sum_l d_l y_l \tag{4.3}$$

Subject To:

$$y(\delta(i)) = 2 \; ; \quad \forall i \in N \tag{4.4}$$

$$y(\delta(S)) \geq 2 \; ; \quad \varnothing \subset S \subset N \tag{4.5}$$

$$y_l \in \{0, 1\} \; ; \quad \forall l \in L \tag{4.6}$$

d_l is the distance of link l, and δ is the symbol of *cut*. Eq. 4.4 tells that the cut of every node i has two links, i.e. it is the ring constraint. Eq. 4.5 tells that there are at least two links jointing nodes between two subsets. It is used to eliminate *sub-tours*(sub-rings). However, the huge number of subsets brings some difficulty to implement constraints in GAMS programs. So eq. 4.5 is substituted in the later mathematical model.

Translating TSP to protection ring design problems, some new parameters need to be introduced:

$\text{FIXEDCOST}_{(i,j)}$	:	A replacement of d_l
CABLENUMBER_c	:	How many cable c are laid on each link
TOTALDEMAND	:	Sum of all demands; $\text{TOTALDEMAND} = \sum_{kl} \text{DEMAND}_{kl}$

As mentioned before, the unit costs of larger capacity cables are comparatively cheaper, so we tend to use them as many as possible. CABLENUMBER_c is obtained by dividing TOTALDEMAND by high-level capacities and leaving the reminder to low-level capacities. Furthermore, we have:

$$\text{FIXEDCOST}_{(i,j)} = \text{SETUPCOST}_{(i,j)} + \sum_c \text{CABLENUMBER}_c \times \text{CABLEPRICE}^c_{(i,j)} \; ; \quad \forall (i,j) \in L \tag{4.7}$$

and the MIP model is:

Minimize:

$$\sum_{(i,j)} \text{FIXEDCOST}_{(i,j)} \times y_{(i,j)} \tag{4.8}$$

Subject To:

$$\sum_j y_{(i,j)} + \sum_j y_{(j,i)} = 2 \ ; \quad \forall i \in N \tag{4.9}$$

$$\sum_j x'^{Node1'l}_{(i,j)} - \sum_j x'^{Node1'l}_{(j,i)} = \begin{cases} 1 & \text{if } i = \text{'Node1'} \\ 0 & \text{otherwise;} \quad \forall i \in N \ ; l \in N \text{ and } l \neq \text{ 'Node1'} \\ -1 & \text{if } i = l \end{cases} \tag{4.10}$$

$$x^{kl}_{(i,j)} \leq y_{(i,j)} \ ; \quad (i,j) \in L \text{ and } i < j \ ; \forall k,l \in N \tag{4.11}$$

$$x^{kl}_{(i,j)} \leq y_{(j,i)} \ ; \quad (i,j) \in L \text{ and } j < i \ ; \forall k,l \in N \tag{4.12}$$

$$x^{kl}_{(i,j)} = 0 \ ; \quad \forall (i,j) \in L \ ; k \in N \text{ and } k \neq \text{ 'Node1'} \tag{4.13}$$

$$y_{(i,j)} = 0 \ ; \quad (i,j) \in L \text{ and } i > j \tag{4.14}$$

$$x^{kl}_{(i,j)} \in \mathbb{R}^+ \ ; \quad \forall (i,j) \in L \ ; \forall k,l \in N \tag{4.15}$$

$$y_{(i,j)} = 0 \text{ or } 1 \ ; \quad \forall (i,j) \in L \tag{4.16}$$

Eq. 4.10 is a bit different from 3.2 that index k is set to be a certain node, 'Node1'. 'Node1' could be any single node in the network, and eq. 4.10 just illustrates that there exists a path between 'Node1' to every other node. Thus sub-tours would be eliminated, and meanwhile there is not redundant non-zero flow between any other nodes pairs. In a word, compared with eq. 4.3 - eq. 4.6, the additional variable $x^{kl}_{(i,j)}$ makes the model easier to be formulated.

We get the same solutions as expected. Running time is greatly reduced by removing variable $z^c_{(i,j)}$.

	Germany	Poland	France	Pioro
No. of Nodes	7	12	25	40
Running Time	<1 s	<1 s	2 s	25 s
Obj. Value	62790	133280	No Integer Solution	3371049
Rise in Obj. (Compared with Basic Ring)	122.6%	242.5%	No Integer Solution	?
Rise in Obj. (Compared with 1+1 Mesh)	36.2%	112.4%	No Integer Solution	243.6%

Table 4.3. Running Time and Objective Values for Protection Ring(TSP Model) Networks.

4.3 Two-Fibre Unidirectional Self-Healing Ring

4.3.1 Mathematical Model

The structure of two-fibre unidirectional self-healing ring is described in Section 2.2.1 fig. 2.8. Compared with the former topologies, its outstanding feature is that demands are directed. I.e. parameter DEMAND_{kl} is the statistical demand of data *from* node k *to* node l. On both active and protection fibre, the two demands between a nodes pair goes through *complementary* links. Thus principle 2.3 and principle 2.4 are not suitable for this model.

Moreover, an active flow, which must carry 100% of a demand, transmits in an unique path and direction. And the respective protection flow, which also carries 100% of the demand, transmits in the opposite directed path. In other words, all active flows transmit in the same direction, either clockwise or anticlockwise. And all protection flows transmit in the opposite direction. Therefore we introduce the concept of *arc*, i.e. a directed link. It will later simplify the formulation. A graph contains arcs(but not links) is called a *digraph*. In this thesis, concepts of arc, digraph and directed demands are merely defined in two-fibre unidirectional self-healing ring model.

The modification of variables are noticeable: Since there is only one active flow in one direction for every demand, $x^{kl}_{(i,j)}$ becomes a binary decision. And

optical carriers on different fibre are identified into two integer capacity decisions, $za^c_{(i,j)}$ and $zb^c_{(i,j)}$. We list all of them below:

$x^{kl}_{(i,j)}$	:	Whether the active flow of demand kl transmits on arc (i,j) from node i to node j
$y_{(i,j)}$	:	Whether to set up arc (i,j) on the active fibre ring
$za^c_{(i,j)}$	:	How many optical carriers c are leased on arc (i,j) for active fibre
$zb^c_{(i,j)}$	:	How many optical carriers c are leased on arc (i,j) for protection fibre

Ultimately, the MIP model of two-fibre unidirectional self-healing ring is:

Minimize:

$$\begin{aligned}\textstyle\sum_{(i,j)}\sum_c za^c_{(i,j)} \times \textsc{CablePrice}^c_{(i,j)} + \sum_{(i,j)}\sum_c zb^c_{(i,j)} \times \textsc{CablePrice}^c_{(i,j)} \\ + \textstyle\sum_{(i,j)} \textsc{SetupCost}_{(i,j)} \times y_{(i,j)}\end{aligned} \tag{4.17}$$

Subject To:

$$\sum_j x^{kl}_{(i,j)} - \sum_j x^{kl}_{(j,i)} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{otherwise;} \\ -1 & \text{if } i = l \end{cases} \quad \forall i \in N \; ; \; k,l \in N \text{ and } \textsc{Demand}_{kl} \neq 0 \tag{4.18}$$

$$\sum_{kl}(x^{kl}_{(i,j)} + x^{kl}_{(j,i)}) \times \textsc{Demand}_{kl} \leq \sum_c za^c_{(i,j)} \times \textsc{CableCapacity}_c \; ; \; (i,j) \in L \text{ and } i < j \tag{4.19}$$

$$\sum_{kl}(x^{kl}_{(i,j)} + x^{kl}_{(j,i)}) \times \textsc{Demand}_{lk} \leq \sum_c zb^c_{(i,j)} \times \textsc{CableCapacity}_c \; ; \; (i,j) \in L \text{ and } i < j \tag{4.20}$$

$$x^{kl}_{(i,j)} + x^{lk}_{(i,j)} = y_{(i,j)} \; ; \quad \forall (i,j) \in L \; ; \; k,l \in N \text{ and } \textsc{Demand}_{kl} \neq 0 \tag{4.21}$$

$$x^{kl}_{(i,j)} = 0 \ ; \quad \forall (i,j) \in L \ ; \ k,l \in N \text{ and } \textsc{Demand}_{kl} = 0 \tag{4.22}$$

$$\sum_{j} y_{(i,j)} = 1 \ ; \quad \forall i \in N \tag{4.23}$$

$$\sum_{j} y_{(j,i)} = 1 \ ; \quad \forall i \in N \tag{4.24}$$

$$za^{c}_{(i,j)} = 0 \ ; \ zb^{c}_{(i,j)} = 0 \ ; \quad \forall c \in C \ ; \ (i,j) \in L \text{ and } i > j \tag{4.25}$$

$$x^{kl}_{(i,j)} = 0 \text{ or } 1 \ ; \quad \forall (i,j) \in L \ ; \forall k,l \in N \tag{4.26}$$

$$y_{(i,j)} = 0 \text{ or } 1 \ ; \quad \forall (i,j) \in L \tag{4.27}$$

$$za^{c}_{(i,j)}, \ zb^{c}_{(i,j)} \in \mathbb{Z}^{+} \ ; \quad \forall (i,j) \in L \ ; \forall c \in C \tag{4.28}$$

Eq. 4.23 and eq. 4.24 indicate that, on the active fibre, there is exact one arc goes into and out of every node. This ensures the active ring to be unidirectional. Moreover, we do not need to consider arcs on the protection ring, because they are exactly opposite arcs of those on the active ring. The protection ring is also unidirectional. In objective eq. 4.17, setup costs on links are accounted once, i.e. only for active arcs. The existence of two fibres only effects on the different reservation costs for optical carriers.

Eq. 4.21 indicates that, on the active fibre, $x^{kl}_{(i,j)}$ and $x^{lk}_{(i,j)}$ transmit in arcs complementarily. I.e. if $x^{kl}_{(i,j)}$ is not passing through arc (i,j), then $x^{lk}_{(i,j)}$ is, and vice versa. Eq. 4.21 also makes active flows keep transmitting in the same direction as active arcs.

Eq. 4.20 indicates that if the active flow of $\textsc{Demand}_{kl}$ transmits through arc (i,j) on the active fibre, the protection flow of $\textsc{Demand}_{lk}$ must transmit through the opposite arc (j,i) on the protection fibre. It is also the reason why we leave out the potential variable, "$xb^{kl}_{(i,j)}$", for flows on the protection fibre.

4.3.2 Comments and Solutions

Basically, the flow of DEMAND_{kl} on the active fibre and the flow of it on the other fibre are two paths of DEMAND_{kl}. In other words, the active path of DEMAND_{kl} contains the same links as the protection path of DEMAND_{lk} does. The only difference is direction. Fig. 4.3 illustrates this comment. For example, on the active fibre, demand AB goes through arc AB(of link AB), and demand BA goes through arc BC(of link BC) $\rightarrow$ arc CA(of link AC). On the protection fibre, demand AB goes through arc AC(of link AC) $\rightarrow$ arc CB(of link BC), and demand BA goes through arc BA(of link AB).

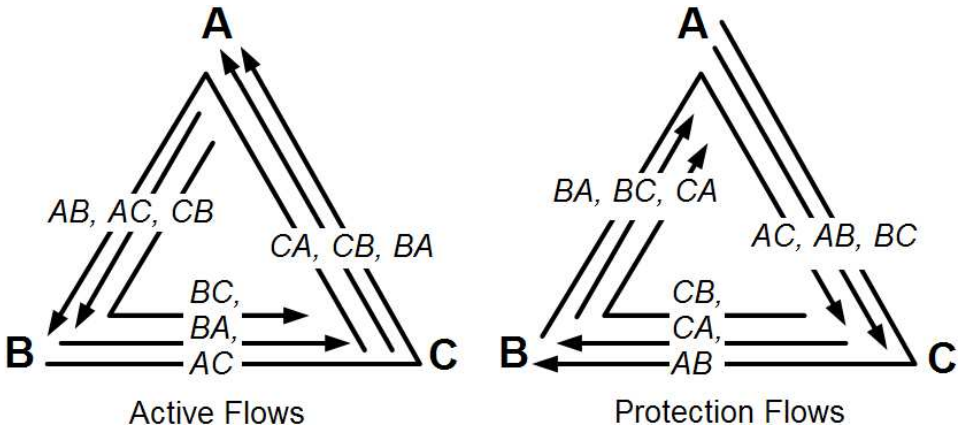


Figure 4.3. An Example of Two-Fibre Unidirectional Self-Healing Ring Solution.

Fig. 4.3 also shows that, on every link, the sum of active and protection capacities is the total demand. It implies that the solution of two-fibre unidirectional self-healing ring would be a little more expensive than the solution of 1+1 protection ring. The reason for increased costs is: The capacities on active ring and protection ring are accounted individually. Hence, the discrete cable capacity feature results in a little more otiose capacity on all links. However, if we have continuous parameter, the unit capacity cost, the objective values of both problems must be the same.

There are no continuous variable in the two-fibre unidirectional self-healing ring model. Large number of binary flow decision $x^{kl}_{(i,j)}$ makes the problem very hard to be solved. Solutions of setup links for Germany and Poland networks

are shown in fig. 4.1 and fig. 4.2. Uncomplete Pioro solution has a 0.35% relative gap away from its best possible lower bound.

	Germany	Poland	France	Pioro
No. of Nodes	7	12	25	40
Running Time	54 s	10 m 3 s	18 s	100 h
Obj. Value	65410	136768	No Integer Solution	3378485 (Best Possible: 3366722.7)
Rise in Obj. (Compared with 1+1 Ring)	4.2%	2.6%	No Integer Solution	0.2%

Table 4.4. Running Time and Objective Values for Two-Fibre Unidirectional Self-Healing Ring Model.

Chapter 5

Tree, Star and Bus Network

In this chapter, we discuss the fixed-charge network design models of tree, star and bus topologies for small scale networks. Although the network instances at hand are of large scale, they are still suitable for programming, solving and comparing.

Different from mesh and ring network design, protection is not studied, and we need to answer the following questions:

- As mentioned in Chapter 2, nodes are sorted according to the network hierarchy. Some nodes might be equipped with hardwares of which costs can not be neglected. How to express that in the model?
- How to implement the management of networks?

5.1 Tree Network Design

5.1.1 Sort of Nodes and Mathematical Model

As shown in fig. 2.9, all nodes in a tree network, except those at the bottom, are roots in various levels. A bottom node is simply a peripheral which is not equipped, and they can only be "sons". We furthermore divide hubs into two types, the unique border router(gateway) which can only be a "father" in the top level and the middle roots which can be a "father" and a "son" in other levels. It is assumed that same hardwares, e.g. LAN switches, are placed at

all middle roots, and they have the same costs. Some more costly hardwares, e.g. important servers, are only placed at the border router.

Thus a new binary variable s_i, whether node i is a root, comes into being. In addition, we have new scalars:

LanSwitchCost	:	Cost of facilities which equipped at middle roots
SonsNumber	:	Number of nodes which joint to the border router
NodesNumber	:	Number of nodes

The value of scalars are given in advance. They might differ in various instances. The cost of gateway is not considered, since it is unique and fixed. Unless otherwise noted, 'Nodetop' is denoted as the gateway from now on. And its location is also given beforehand.

Communication between any single node in the tree and the outside environment of the tree pass through 'Nodetop'. Therefore although demands to outside are non-zero, they are uniformly counted as demands to 'Nodetop'. Unchanged demand matrices make the basic fixed-charge mesh model available. Tree topology can also be considered as a subset of mesh topology. All sets, indices, the other parameters and variables are same as those of the basic mesh model. The MIP formulation of fixed-charge tree network design is:

Minimize:

$$\begin{aligned}\sum_{(i,j)} \sum_c z^c_{(i,j)} \times \text{CablePrice}^c_{(i,j)} + \sum_{(i,j)} \text{SetupCost}_{(i,j)} \times y_{(i,j)} \\ + \left(\sum_i s_i - 1 \right) \times \text{LanSwitchCost}\end{aligned} \tag{5.1}$$

Subject To:

$$\sum_j x^{kl}_{(i,j)} - \sum_j x^{kl}_{(j,i)} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{otherwise;} \\ -1 & \text{if } i = l \end{cases} \quad \forall i \in N \text{ ; } k, l \in N \text{ and } \text{Demand}_{kl} \neq 0 \tag{5.2}$$

$$\sum_{kl} (x^{kl}_{(i,j)} + x^{kl}_{(j,i)}) \times \text{Demand}_{kl} \leq \sum_c z^c_{(i,j)} \times \text{CableCapacity}_c \text{ ; } (i,j) \in L \text{ and } i < j \tag{5.3}$$

$$x^{kl}_{(i,j)} \leq y_{(i,j)} \ ; \quad (i,j) \in L \text{ and } i < j \ ; \forall k,l \in N \tag{5.4}$$

$$x^{kl}_{(i,j)} \leq y_{(j,i)} \ ; \quad (i,j) \in L \text{ and } j < i \ ; \forall k,l \in N \tag{5.5}$$

$$\sum_j y_{('Nodetop',j)} + \sum_j y_{(j,'Nodetop')} = \text{SonsNumber} \tag{5.6}$$

$$\sum_j y_{(i,j)} + \sum_j y_{(j,i)} \leq s_i \times \text{NodesNumber} + 1 \ ; \quad \forall i \in N \tag{5.7}$$

$$\sum_{(i,j)} y_{(i,j)} = \text{NodesNumber} - 1 \tag{5.8}$$

Eq. 3.6 - Eq. 3.11

One more term appears in the objective, and $\sum_i s_i$ is the number of roots. However, the cost of 'Nodetop' is not counted, so the additional term represents the total cost of middle roots.

Eq. 5.6 ensures that certain numbers of middle roots connect straightly to 'Nodetop'. It is a kind of management constraint. In practical cases, we can add more management description in the model by adding more constraints. Eq. 5.7 indicates that i is a root if more than one nodes joint it. Eq. 5.8 is a tree constraint. It demonstrates that in any tree, the number of links must be the number of nodes minus one.

5.1.2 Comments and Solution

In fact, there exists only one path for every nodes pair in tree topology. If node A is the nearest common "ancestor" of both node B and C, the path for Demand_{BC} should be the combination of path for Demand_{AB} and path for Demand_{AC}. Eq. 5.8 supports this feature well: A connected graph with links one less than nodes must be of tree topology, and thus provide only one path for every nodes pair.

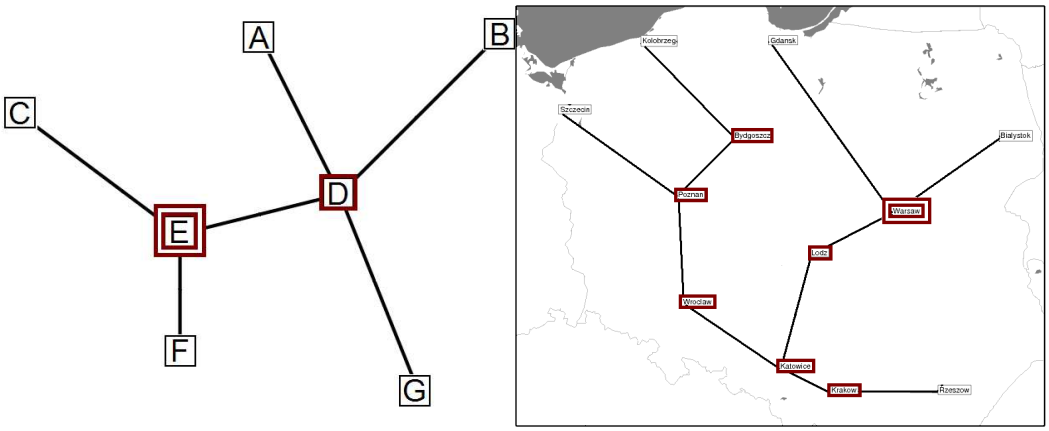


Figure 5.1. Tree Solution of Germany Network. The Fixed-Charge is Dominating.

Figure 5.2. Tree Solution of Poland Network. The Cable Cost is Dominating.

Fig. 3.1 shows that the mesh solution of a fixed-charge dominating network is tend to be a tree. Therefore the tree solutions for these networks are similar. Table 5.1 indicates that, for Germany network, the rise of objective only comes from an additional LAN switch at node D. On the other hand, tree solution cost much more in cable cost dominating networks. Setup links for Germany and Poland networks are shown in fig. 5.1 and fig. 5.2. The bound router and middle roots are lined out. Fixed-charge tree design problems are as hard as basic mesh and ring design problems.

	Germany	Poland	France	Pioro
No. of Nodes	7	12	25	40
SonsNumber	3	3	4	5
Running Time	1 s	16 m 27 s	181 m 30 s	75 h
Obj. Value	23050	37085	57424	525456 (Best Possible: 506727.6)
Rise in Obj. (Compared with Basic Mesh)	1.8%	33.9%	8.8%	22.0%

Table 5.1. Running Time and Objective Values for Tree Networks.

Cost of middle roots effect on the solutions as well. If these middle hardware costs are considerable, the network would set up middle roots as fewer as possible. And the number of levels would correspondingly decrease. Nevertheless, LAN switch cost is comparatively little, and it might result in "long history" trees, e.g. Poland network.

5.2 Star Network Design

5.2.1 Star Characteristics

In a logical point of view, the distinct feature of star topology is that communication between every nodes pair passes through the only hub. Therefore the demands between peripheral i and the other peripherals, the hub and the outside environment can be considered as the demands between i and the hub. This truth is so useful that the number of variables would be decreased.

'Nodetop' is still denoted as the hub. We define parameter DEMANDHUB_k as the total demands from terminal k to outside.[1]

$$\begin{gathered}\text{DEMANDHUB}_k = \textstyle\sum_l \text{DEMAND}_{kl} + \sum_l \text{DEMAND}_{lk} \\ \text{DEMANDHUB}'_{Nodetop'} = 0.\end{gathered} \tag{5.9}$$

Correspondingly the flow decision is modified to $x^k_{(i,j)}$.

Like the tree topology, another important feature of star is that the path between any node i to 'Nodetop' is unique. Hence, $x^k_{(i,j)}$ could be defined as a binary variable which tells whether the path for $\text{DEMANDHUB}'_{k'}$ contains link (i, j). IP mathematical models based on the binary flow decision were built. Though the running time for Germany, Poland and France networks are within 10 seconds, the largest Pioro network could not be solved. So the formulation with binary $x^k_{(i,j)}$ is not presented in this chapter, but GAMS code is attached in Appendix C.11.

[1] Principle 2.1 - principle 2.4 are available.

On the other hand, star topology is a subset of tree. In a physical point of view, a star may have more than two levels as shown in fig. 2.10 and fig. 2.11. That means not all terminals are jointed straightly to the hub. Eq. 5.8 is a tree constraint,[2] and it can be applied in star models. As mentioned before, eq. 5.8 also ensures the unique path feature. Thus the flow decision $x^k_{(i,j)}$ could be kept continuous, though its only possible value in solutions is either 0 or 1.

5.2.2 Mathematical Model: No Multiplexing

Fig. 2.10 describes a star network with no multiplexing. Cables are independent from each other, and so are flows.

Since the path is unique, cables laid for every single $\text{DEMANDHUB}_{k'}$ is fixed in advance. A new parameter CABLENUMBER^c_k is defined as the number of cable c for DEMANDHUB_k. Like the disposal in 1+1 ring protection models(TSP), it is obtained by dividing DEMANDHUB_k by high-level capacities and leaving the reminder to low-level capacities. Hence, the MIP model is:

Minimize:

$$\begin{aligned}\sum_k \sum_{(i,j)} \sum_c \text{CABLENUMBER}^c_k \times x^k_{(i,j)} \times \text{CABLEPRICE}^c_{(i,j)} \\ + \sum_{(i,j)} \text{SETUPCOST}_{(i,j)} \times y_{(i,j)}\end{aligned} \tag{5.10}$$

Subject To:

$$\sum_j x^k_{(i,j)} - \sum_j x^k_{(j,i)} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{otherwise;} \quad \forall i \in N \text{ ; } k \in N \text{ and } k \neq \text{'Nodetop'} \\ -1 & \text{if } i = \text{'Nodetop'} \end{cases} \tag{5.11}$$

$$x^k_{(i,j)} \leq y_{(i,j)} \text{ ; } \quad (i,j) \in L \text{ and } i < j \text{ ; } \forall k \in N \tag{5.12}$$

$$x^k_{(i,j)} \leq y_{(j,i)} \text{ ; } \quad (i,j) \in L \text{ and } j < i \text{ ; } \forall k \in N \tag{5.13}$$

[2]Since it is a constraint for setup decision $y_{(i,j)}$, it only effects on physical level but not logical level.

$$\sum_{(i,j)} y_{(i,j)} = \text{NodesNumber} - 1 \tag{5.14}$$

$$x_{(i,j)}^{'Nodetop'} = 0 \; ; \quad \forall (i,j) \in L \tag{5.15}$$

$$y_{(i,j)} = 0 \; ; \quad (i,j) \in L \text{ and } i > j \tag{5.16}$$

$$x_{(i,j)}^{k} \in \mathbb{R}^{+} \; ; \quad \forall (i,j) \in L \; ; \forall k \in N \tag{5.17}$$

$$y_{(i,j)} = 0 \text{ or } 1 \; ; \quad \forall (i,j) \in L \tag{5.18}$$

Eq. 5.14 ensures the unique path feature, and furthermore results in the {0,1} solution of $x_{(i,j)}^{k}$, i.e. $x_{(i,j)}^{k}$ could only be 0 or 1. That is the reason why $x_{(i,j)}^{k}$ operates like an integer decision in objective eq. 5.10. The first part of eq. 5.10 is cable cost and the second part is setup cost.

5.2.3 Comments and Solution: No Multiplexing

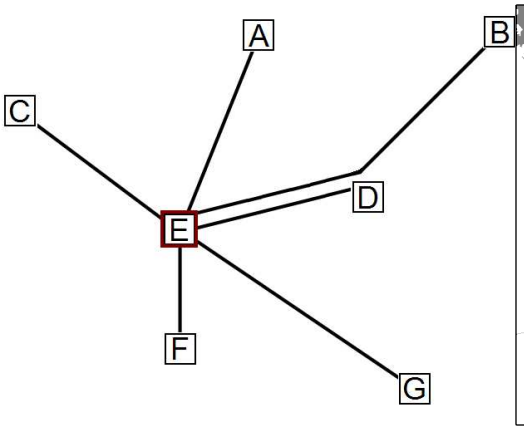


Figure 5.3. Star Solution(No Multiplexing) of Germany Network.

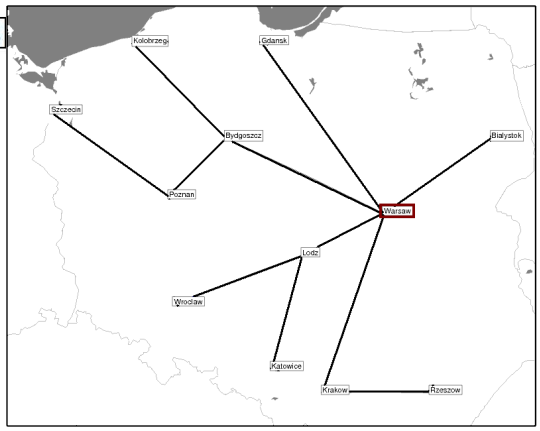


Figure 5.4. Star Solution(No Multiplexing) of Poland Network.

Compared with tree networks, star networks have extra cost on capacities. In a tree network, nodes pairs which are close and have considerably large demands are tend to be dealt with the "father"-and-"son" relationship. However, in a star network, all terminals only connect to the hub, no matter how long the distances are.

Thus peripherals in a star network always connect to the hub as closely as possible. The solutions of Germany and Poland networks show that all links which joint the hub are set up. That also results in fewer physical levels. When setup costs are zero, the star network design problem(without multiplexing) can be decomposed to several independent *one-to-one shortest path problems*, i.e. one for every peripheral.

Removing of $z^c_{(i,j)}$ and decreasing of variables(from $O(n^4)$ to $O(n^3)$) make the non-multiplexing star model very easy to be solved. Basically, the objective values of star networks are more expensive, especially in large scale. Table 2.2 indicates that the average node degree of a network is normally less than 5, so in large scale star networks, most nodes are far away from the hubs(physically several levels lower than the hubs). Therefore much more capacities need to be reserved.

	Germany	Poland	France	Pioro
No. of Nodes	7	12	25	40
Running Time	<1 s	<1 s	1 s	2 s
Obj. Value	24235	38104	57916	860026
Rise in Obj. (Compared with Tree)	5.1%	2.7%	0.9%	63.7%

Table 5.2. Running Time and Objective Values for Star(No Multiplexing) Networks.

5.2.4 Mathematical Model: With Multiplexing

Fig. 2.11 describes a star network which bases on the implementation of multiplexing. For saving capacities, a MUX DEMUX pair is placed at every

pedestal, i.e. a middle root in physical perspective. The hardware cost is denoted as scalar MuxCost, and we have the corresponding binary variable m_i, whether node i is equipped with MUX.

The continuous flow decision $x^k_{(i,j)}$ follows the previous star model. And since capacities are multiplexed at pedestals, cables which are laid for DemandHub_k on every link of its unique path are not determinate beforehand. Thus the integer capacity decision $z^c_{(i,j)}$ appears again. The MIP formulation is:

Minimize:

$$\begin{aligned}\sum_{(i,j)} \sum_c z^c_{(i,j)} \times \text{CablePrice}^c_{(i,j)} + \sum_{(i,j)} \text{SetupCost}_{(i,j)} \times y_{(i,j)} \\ + \left(\sum_i m_i - 1 \right) \times \text{MuxCost}\end{aligned} \tag{5.19}$$

Subject To:

$$\sum_j x^k_{(i,j)} - \sum_j x^k_{(j,i)} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{otherwise;} \quad \forall i \in N \text{ ; } k \in N \text{ and } k \neq \text{'Nodetop'} \\ -1 & \text{if } i = \text{'Nodetop'} \end{cases} \tag{5.20}$$

$$\sum_k (x^k_{(i,j)} + x^k_{(j,i)}) \times \text{DemandHub}_{kl} \leq \sum_c z^c_{(i,j)} \times \text{CableCapacity}_c \text{ ; } (i,j) \in L \text{ and } i < j \tag{5.21}$$

$$\sum_j y_{(i,j)} + \sum_j y_{(j,i)} \leq m_i \times \text{NodesNumber} + 1 \text{ ; } \quad \forall i \in N \tag{5.22}$$

$$z^c_{(i,j)} = 0 \text{ ; } \quad \forall c \in C \text{ ; } (i,j) \in L \text{ and } i > j \tag{5.23}$$

$$z^c_{(i,j)} \in \mathbb{Z}^+ \text{ ; } \quad \forall (i,j) \in L \text{ ; } \forall c \in C \tag{5.24}$$

Eq. 5.12 - Eq. 5.18

The fixed-charge star design model(with multiplexing) is almost the same as the tree model, and it implies the subset affiliation. Actually, it is a special tree problem in which the demand matrix only have one row[3]. Hence, $x^k_{(i,j)}$ is not a change of the tree model but a simplification which expresses the star characteristic.

5.2.5 Comments and Solution: With Multiplexing

Fig. 5.5 and fig. 5.6 show the solution of setup links for Germany and Poland networks. The hubs and pedestals are lined out.

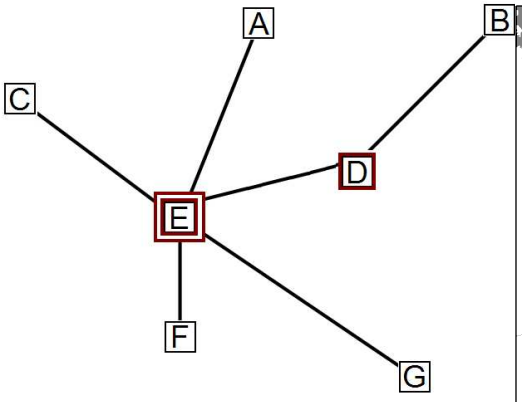


Figure 5.5. Star Solution(With Multiplexing) of Germany Network.

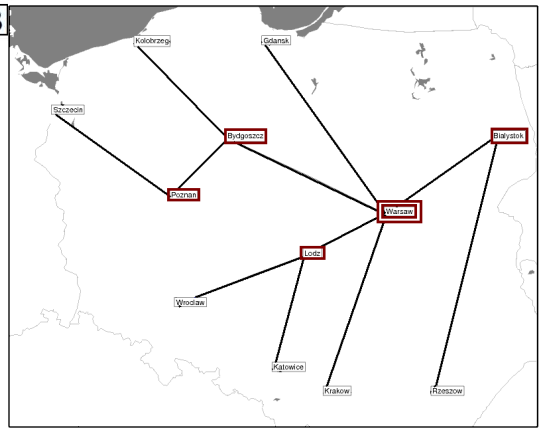


Figure 5.6. Star Solution(With Multiplexing) of Poland Network.

In any network, the star solution can be considered as the tree solution. If fig. 5.5 was looked upon as a tree, there would be no reserved capacity for demand BD on link DE. And the tree solution would save this cost. However, as mentioned before, all peripherals connect to the hub as closely as possible in a small scale star network. Therefore links like DE, which lays cables for more than one demands, are limited. In a word, the tree solution, which may have the same setup links as the star solution, is not too much cheaper in a small network.

[3]The index of this row is $'Nodetop'$.

Contrarily, in a large scale star network, there are much more links which apply multiplexing technology like DE, and these links even exist in much lower physical levels. Thus the tree solution would save a lot.

	Germany	Poland	France	Pioro
No. of Nodes	7	12	25	40
Running Time	<1 s	<1 s	1 m 27 s	75 h
Obj. Value	24050	37204	56453	847842 (Best Possible: 846391.5)
Rise in Obj. (Compared with Tree)	4.3%	0.3%	1.7%	61.4%
Save in Obj. (Compared with Non-Multiplexing Star)	0.07%	2.4%	2.5%	1.4%

Table 5.3. Running Time and Objective Values for Star(No Multiplexing) Networks.

In addition, star solutions with multiplexing are apparently a little cheaper than star solutions without multiplexing. It is explained in Section 2.2.2. Comparison of objective values is demonstrated in table 5.3. Uncomplete Pioro solution has a 0.17% relative gap away from its best possible lower bound.

5.3 Bus Network Design

5.3.1 Mathematical Model

In a bus network of LAN scale, terminals are connected by *sharing* a single cable with *fixed* bandwidth. Hence, reservation of capacity is not described in this model. Since the setup costs are in direct proportion with the cable costs, what needs to be concerned in this model is just a general "distance" parameter between cities. Furthermore, the bus network design problem can be considered as an Open TSP problem which has different starting city and end city. We assume that the hub node 'Nodetop' is the staring point.

All sets, indices, parameters and variables have the same meaning as those of the star model. Ultimately, the MIP model is:

Minimize:

$$\sum_{(i,j)} \textsc{SetupCost}_{(i,j)} \times y_{(i,j)} \tag{5.25}$$

Subject To:

$$\sum_j x^k_{(i,j)} - \sum_j x^k_{(j,i)} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{otherwise;} \quad \forall i \in N \ ; \ k \in N \text{ and } k \neq \text{'Nodetop'} \\ -1 & \text{if } i = \text{'Nodetop'} \end{cases} \tag{5.26}$$

$$x^k_{(i,j)} \leq y_{(i,j)} \ ; \quad (i,j) \in L \text{ and } i < j \ ; \forall k \in N \tag{5.27}$$

$$x^k_{(i,j)} \leq y_{(j,i)} \ ; \quad (i,j) \in L \text{ and } j < i \ ; \forall k \in N \tag{5.28}$$

$$\sum_{(i,j)} y_{(i,j)} = \textsc{NodesNumber} - 1 \tag{5.29}$$

$$\sum_j y_{(i,j)} + \sum_j y_{(j,i)} \leq 2 \ ; \quad \forall i \in N \tag{5.30}$$

$$\sum_j y_{(\text{'Nodetop'},j)} + \sum_j y_{(j,\text{'Nodetop'})} = 1 \tag{5.31}$$

Eq. 5.15 - Eq. 5.18

The function of eq. 5.26 - eq. 5.28 is only eliminating sub-tours. Eq. 5.29 indicates that bus topology is also a subset of tree topology. Physically, it is a tree in which every "father" only has one "son". Eq. 5.29 and eq. 5.30 together ensure that only two nodes in the network will joint to one link, and the other nodes will joint to two links. Eq. 5.31 tells that 'Nodetop' is one of the two nodes, and is the staring point of the tour.

Eq. 5.29 - eq. 5.31 are the bus constraints. They can be applied in large scale network design models where cost of capacity will be taken into account. However, we do not discuss more on that.

5.3.2 Solutions

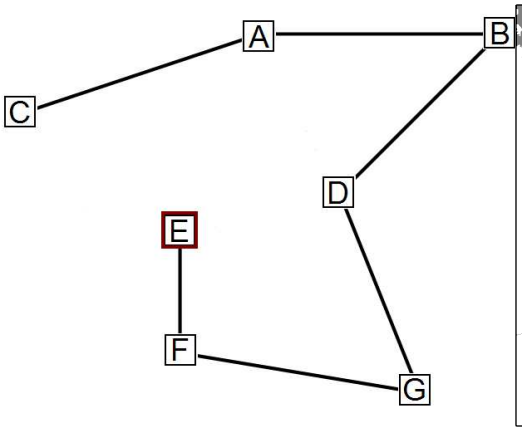


Figure 5.7. Bus Solution of Germany Network.

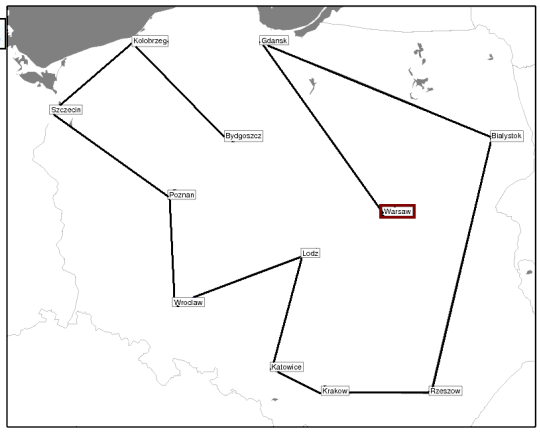


Figure 5.8. Bus Solution of Poland Network.

Eq. 5.25 illustrates that setup costs(representing the "distances") are the only contribution to the objective. Thus it is not necessary to compare bus solutions with tree and star topology. Like TSP problem studied in Section 4.2.2, the bus network design problem is very easy to be solved.

	Germany	Poland	France	Pioro
No. of Nodes	7	12	25	40
Running Time	<1 s	<1 s	5 s	28 s

Table 5.4. Running Time for Bus Networks.

Chapter 6

Conclusion

6.1 Future Research

The possible future research for fixed-charge network design problems include the improvement of mathematical models and the application of large scale optimization algorithms.

6.1.1 Improvement of Mathematical Models

The mathematical models shown in the previous chapters are still not accurate enough to describe the real Telecommunication Networks. And we need to consider several more aspects when designing networks.

Firstly, the concept of *hop-limit paths* can be introduced. In the old models, the length of paths are not restricted. Though the solutions indicate that most paths are comparatively short, there always exist few long paths which are generated for filling the otiose capacities on other links. For example, there might be a 9-links path which carrying 5.6% data for a single demand, and it is different from the practical routing scheme.

We can deal with hop-limit paths by adding indices or new binary counting decisions to mark the length of paths, but both methods bring a great amount of variables which would make the problem much harder. We can also decompose the problems into 2 levels, one of which only concerning how to generate the network with a required topology and the hop-limit paths.

Secondly, the hardwares equipped at the nodes should be specified even in backbone networks. Their costs may result in totally different solutions. In addition, there might be limitation on the usage of some particular facilities.

Thirdly, the modelling for tree management has to be improved. We can control the number of physical levels to restrict the "history" of the trees. It is similar to the hop-limit paths constraints. We can furthermore control the number of "sons" that a "father" could have. And this can simply be implemented by modifying eq. 5.6 to $\sum_j y_{(i,j)} + \sum_j y_{(j,i)} \leq \textsc{SonsNumber}_i \ ; \quad \forall i \in N.$

In a word, the more precise constraints the models describe, the more variables there are and the harder the problems would be. Moreover, large networks, e.g. Pioro network which has 40 nodes, could not be solved completely in most previous models. Therefore it is necessary to apply large scale optimization algorithms at the same time.

6.1.2 Large Scale Optimization Algorithms

Basically, as demonstrated in Section 3.3.4, the elimination of integer capacity decision $z^c_{(i,j)}$ would not effect a lot on signal routing and the final results, meanwhile it greatly simplifies all models. Hence, we even do not have to implement large scale optimization algorithms on network instances at hand. However, there always exist extreme huge networks for which application of methods would greatly decrease the solving time.

Column generation has already been applied to solve the link protection mesh problems. In fact, this method is possible to be applied for models of all topologies. Assuming that a network of given topology is available at hand, the sub-problems could generate few more links which can, together with original links, keep that topology. Only using the generated links, the master problems would find solutions(rings by TSP algorithms, trees by MST algorithms and so on) base on a simple "distance" parameter.

However, the expression of "distance" is difficult. Ignoring the influence of $z^c_{(i,j)}$, "distance" would be a balance of setup costs and continuous capacity costs. And it varies in different models.

In addition, *branch-and-cut* and *heuristics* are assistant methods which can be applied combinatively with column generation. It is attemptable to design the network part by part.

6.2 Summary of Objective Values

The objective values(except bus solutions and link protection mesh solutions) are summed up in table 6.1. Germany and France are representatives of fixed-charge dominating networks, and Poland and Pioro are representatives of cable costs dominating networks. Obviously, the non-protective fixed charge design is cheapest. On the other hand, the 1+1 protection ring design and the two-fibre unidirectional self-healing ring design are most expensive. Costs of star and tree designs are close.

	Germany	Poland	France	Pioro
No. of Nodes	7	12	25	40
Obj. Value (Basic Mesh)	22650	27691	52775	430680 (Best Possible: 427988.2)
Obj. Value (1+1 Mesh)	46090	62757	112688 (Best Possible: 111897.6)	981042 (Best Possible: 977531.6)
Obj. Value (Basic Ring)	28210	38911	No Integer Solution	Not Solved
Obj. Value (1+1 Ring)	62790	133280	No Integer Solution	3371049
Obj. Value (Self Healing Ring)	65410	136768	No Integer Solution	3378485 (Best Possible: 3366722.7)
Obj. Value (Tree)	23050	37085	57424	525456 (Best Possible: 506727.6)
Obj. Value (Non-Multiplexing Star)	24235	38104	57916	860026
Obj. Value (Multiplexing Star)	24050	37204	56453	847842 (Best Possible: 846391.5)

Table 6.1. Comparison of Objective Values.

From the perspective of protection, ring protection designs are much more expensive than mesh protection designs. However, ring networks require the simplest routing scheme, and furthermore provide the fastest recovering time. This is also an important advantage in nowadays network design issues. Hence, mesh and ring topologies are both referencable for backbone networks which can not survive without protection.

6.3 Achievement

In this thesis mathematical models are built for fixed-charged network design problems.

Six models are presented for non-protective mesh, ring, tree, star and bus topologies. The models of non-multiplexing star and bus topologies are easy to be solved. However, the other models are hard to be solved, and the complete solutions for some large networks are not obtained. Thus these models still need to be improved.

Besides, five protection models are presented for backbone mesh and ring topologies. 1+1 protection mesh and ring models are built respectively, but they are as hard as the non-protective designs. Therefore link protection mesh model and TSP model for 1+1 protection ring are presented. Both two models guarantee 100% protection and short solving time. Finally, the mathematical model of a special practical ring, the two-fibre unidirectional self-healing ring, is provided.

Other achievements are the comments on solutions and the comparison of objective values. The analysis indicates which topology fits into the network(either fixed-charge dominating or cable costs dominating) at hand. And this is the designers' ultimate task.

Bibliography

[1] Library of test instances for survivable fixed telecommunication network design. *http://opt36.zib.de:8080/sndlib/home.action*.

[2] Network topology from wikipedia, the free encyclopedia. *http://en.wikipedia.org/wiki/Network-topology*.

[3] Indra Widjaja Alberto Leon-Garcia. *Communication Networks: Fundamental Concepts and Key Architectures*. McGraw-Hill Companies, Inc., 2003.

[4] I. Saniee O. Wasem S. Cosares, D. Deutsch. *SONET Toolkit: A Decision Support System for Designing Robust and Cost-Effective Fiber-Optic Networks*. Interfaces 25: 1 January - February, 1995.

[5] T. Stidsen. *Efficient Protection*. Informatics and Mathematical Modelling, DTU, 2005.

[6] T. Stidsen. *Routing and Protection*. Informatics and Mathematical Modelling, DTU, 2005.

Appendix A

Fixed-Charge Network Instances

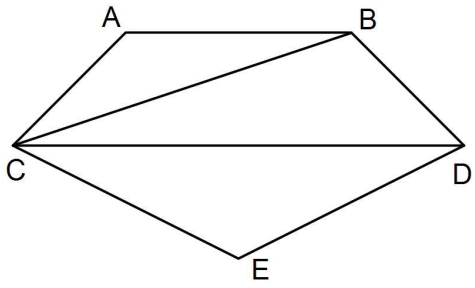


Figure A.1. Arne Network with 5 Nodes and 7 Links.

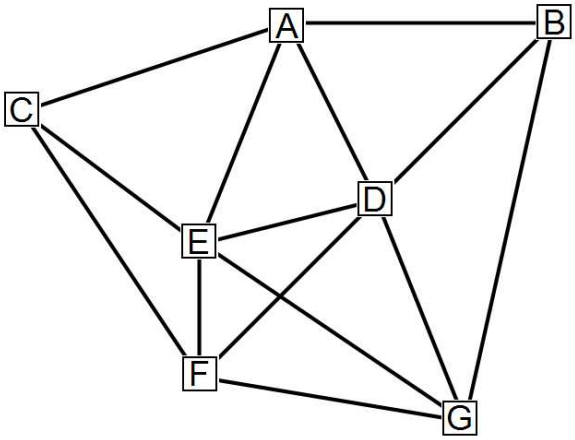


Figure A.2. A Network of Germany with 7 Nodes and 14 Links.

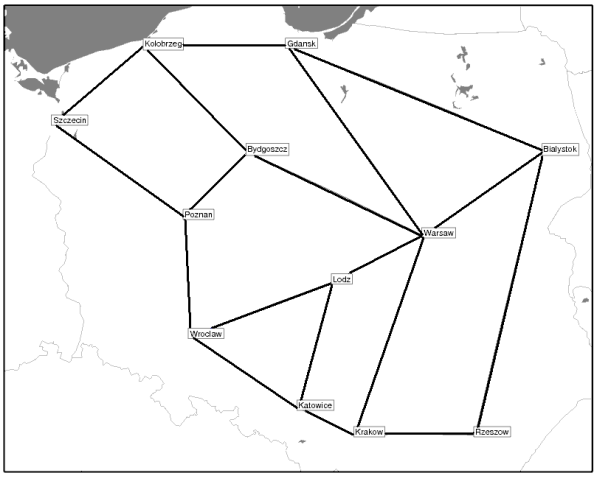


Figure A.3. A Network of Poland with 12 Nodes and 18 Links.

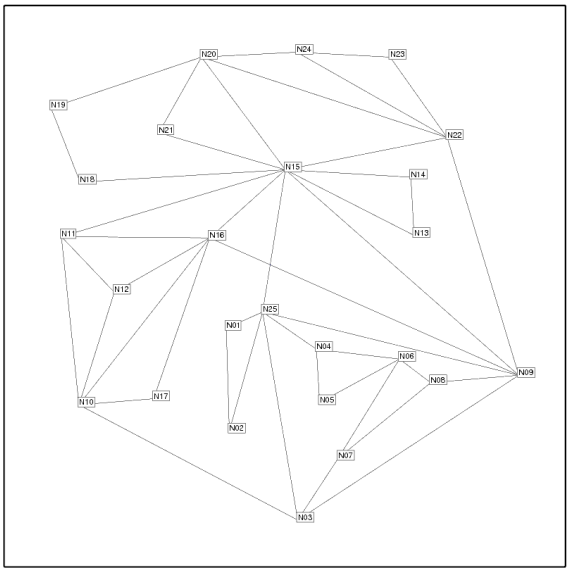


Figure A.4. A Network of France with 25 Nodes and 45 Links.

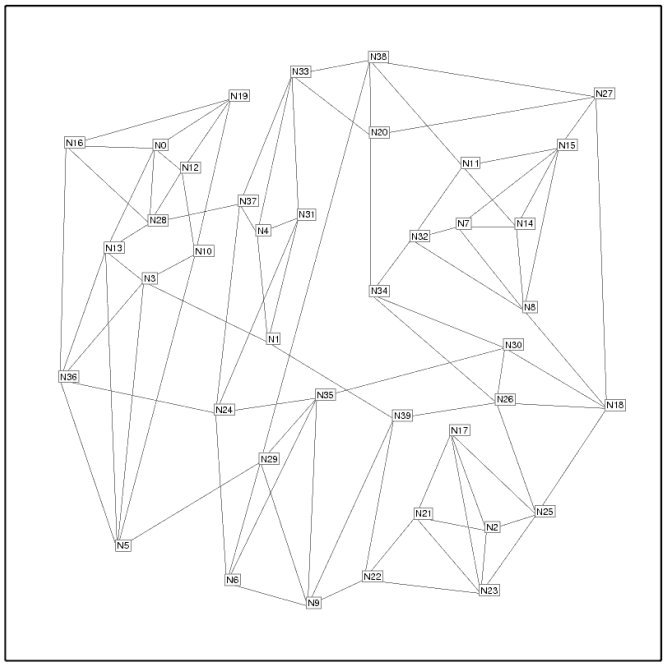


Figure A.5. A Network of Pioro with 40 Nodes and 89 Links.

Appendix B

Is Link Protection Cheaper?

Fig. B.2 - fig. B.7 illustrate that link protection is usually more economical than 1+1 protection. The relevant demand matrix for this example is given:

	B	C	D
A	1	2	1
B		2	1
C			2

We assume that the setup costs on all links are same, and so are the cable prices of unit capacity. The network is:

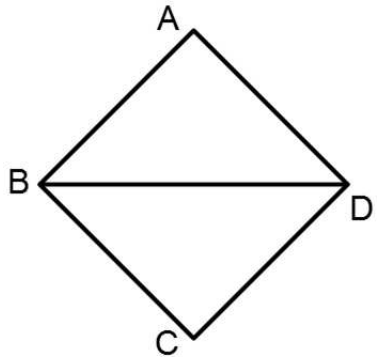


Figure B.1. An Example Network.

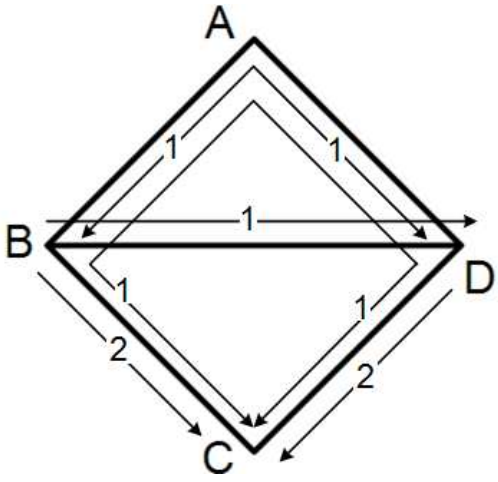


Figure B.2. Primary Link Flows by Shortest Path Algorithm.

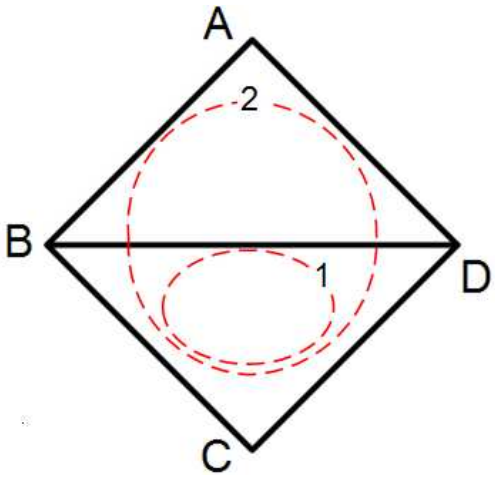


Figure B.3. Two Protection Sub-rings.

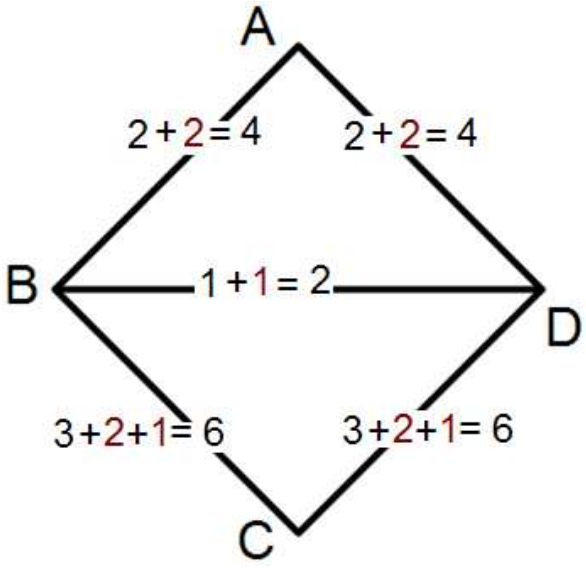


Figure B.4. Solution of Link Protection. Black Numbers for Primary Flows and Red Numbers for Protection Flows.

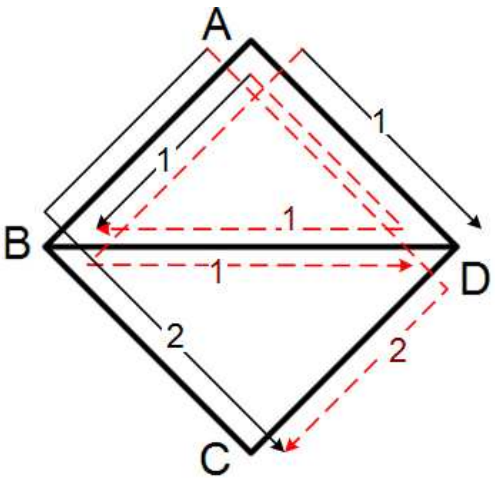


Figure B.5. Primary (Black) and Backup (Red) Paths for DEMAND_{AB}, DEMAND_{AC} and DEMAND_{AD}.

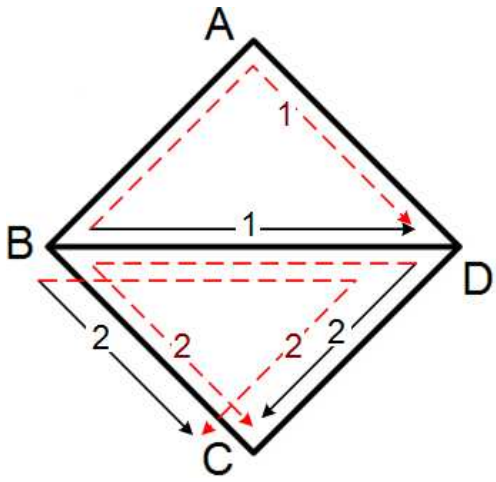


Figure B.6. Primary (Black) and Backup (Red) Paths for DEMAND_{BC}, DEMAND_{BD} and DEMAND_{CD}.

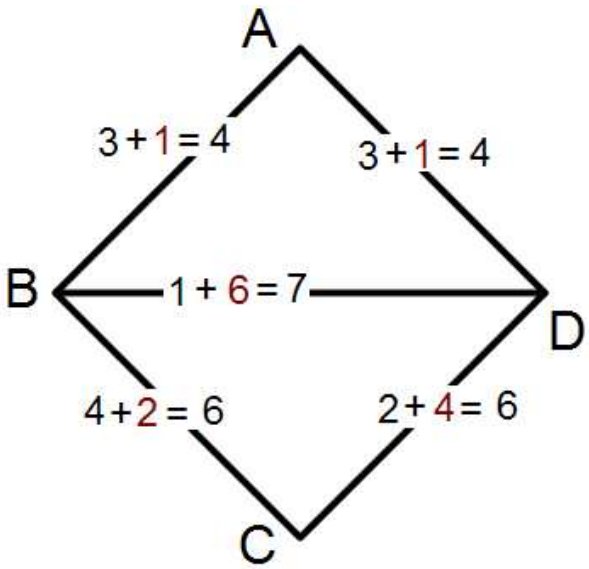


Figure B.7. Solution of 1+1 Protection. Black Numbers for Primary Flows and Red Numbers for Protection Flows.

Appendix C

GAMS Programs Code

The GAMS code for all the 11 models are listed below. The network instance which is included in the code is Germany network.

C.1 Basic Fixed-Charge Network Design Model

```
* Network Design Model.gms
* Feb. 16, 2006
* 14th Program: Included Network Data
*
* Fixed Cost, Cable Prices And Cable Capacities And Etc. From SNDLib Instances
* Cables In Fact Are Optical Carriers; Cable Prices Are Leased Fee
* A Mesh Network Model; An Undirected Demand Mesh
* Switches, MUXs And De-MUXs Are Equipped At All Nodes
* Equipments' Costs Are Not Considered
* No Protection
*
* x Is Between [0, 1]
* Non-used z And y Is Removed
*
$eolcom //
option iterlim=999999999; // avoid limit on iterations
option reslim=270000;     // timelimit for solver in sec.
option optcr=0.0;         // gap tolerance
option solprint=OFF;       // include solution print in .lst file
option limrow=0;          // limit number of rows in .lst file
option limcol=0;          // limit number of columns in .lst file
//-----------------------------------------------------------------------
```

```
  Sets
    Node                name of cities

    ALIAS(Node,i)
    ALIAS(Node,j)
    ALIAS(Node,k)
    ALIAS(Node,l);

  Set
    Cbl                 types of cables with different capacities;

  Parameters
    XX(Node)            x-bar of a city
    YY(Node)            y-bar of a city

    Cbl_Cpy(Cbl)        cable capacities
    Cbl_Prc(i,j,Cbl)    cable prices on link #ij#

    SPAN(i,j)           whether the link between #ij# exsits
    Pre_Cap(i,j)        pre-installed capacity already exsits on #ij#
    Pre_Cst(i,j)        how much have to pay on the pre-installed capacity
    Unt_Rt_Cst(i,j)     unit routing cost on #ij#
    Stp_Cst(i,j)        if not set up, how much should pay to set up

    D(k,l)              how much is the demand of flow between #kl#
    Rt_Unt(k,l)         the size of a routing unit on demand #kl#
    Dmd_Amt(k,l)        how much is the demand of unit between #kl#
    TOTAL_DEMAND        sum of demands of the entire network;

$INCLUDE "deutsch.dat";

  // build a symmetric network
  SPAN(i,j)$(ord(i)<ord(j)) = SPAN(i,j) + SPAN(j,i);
  SPAN(i,j)$(ord(i)>ord(j)) = SPAN(j,i);

  Pre_Cap(i,j)$(ord(i)<ord(j)) = Pre_Cap(i,j) + Pre_Cap(j,i);
  Pre_Cap(i,j)$(ord(i)>ord(j)) = Pre_Cap(j,i);

  Pre_Cst(i,j)$(ord(i)<ord(j)) = Pre_Cst(i,j) + Pre_Cst(j,i);
  Pre_Cst(i,j)$(ord(i)>ord(j)) = Pre_Cst(j,i);

  Unt_Rt_Cst(i,j)$(ord(i)<ord(j)) = Unt_Rt_Cst(i,j) + Unt_Rt_Cst(j,i);
```

```
Unt_Rt_Cst(i,j)$(ord(i)>ord(j)) = Unt_Rt_Cst(j,i);

Stp_Cst(i,j)$(ord(i)<ord(j)) = Stp_Cst(i,j) + Stp_Cst(j,i);
Stp_Cst(i,j)$(ord(i)>ord(j)) = Stp_Cst(j,i);

Cbl_Prc(i,j,Cbl)$(ord(i)<ord(j)) = Cbl_Prc(i,j,Cbl) + Cbl_Prc(j,i,Cbl);
Cbl_Prc(i,j,Cbl)$(ord(i)>ord(j)) = Cbl_Prc(j,i,Cbl);

// transfer data to demand matrix D, and make it upper triangular
// so here the demands are considered as undirected
D(k,l) = Dmd_Amt(k,l)*Rt_Unt(k,l)$(Rt_Unt(k,l)>0) + Dmd_Amt(k,l)$(Rt_Unt(k,l)=0);
TOTAL_DEMAND = sum((k,l), D(k,l));

D(k,l)$(ord(k)<ord(l)) = D(k,l) + D(l,k);
D(k,l)$(ord(k)>=ord(l)) = 0;

BINARY Variables
     y(i,j)     whether to build the arc #ij#;
// set all y to be #0#
     y.fx(i,j) = 0;
// set all y on half of the exsited links possible to be #1#
     y.up(i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = 1;

INTEGER Variables
     z(Cbl,i,j) whether to lease carrier #Cbl# in arc #ij#, and how many;
// set all z to be #0#
     z.fx(Cbl,i,j) = 0;
// set all z on half of the exsited links possible to be non-zero integer
     z.up(Cbl,i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = ceil (TOTAL_DEMAND/Cbl_Cpy(Cbl));

Variables
     cost         total cost
     x(i,j,k,l)   how much percentage of flow through arc #ij# for demand #kl#;
// set all x to be #0#
     x.fx(i,j,k,l) = 0;
// set all x on exsited links to have upper bound as #1#
     x.up(i,j,k,l)$(SPAN(i,j) and (D(k,l) <> 0)) = 1;

Equations
     obj               define objective function
     c1(j,k,l)         demand constraints
     c2(i,j)           link capacity constraints
     c3(i,j,k,l)
```

```
       c4(i,j,k,l)       ;

obj..       cost =E= sum((Cbl,i,j), z(Cbl,i,j)*Cbl_Prc(i,j,Cbl))
                       + sum((i,j), Stp_Cst(i,j)*y(i,j));

c1(j,k,l)$(D(k,l) <> 0)..
            sum(i, x(i,j,k,l)) - sum(i, x(j,i,k,l))
                     =E= -1$(ord(j)=ord(k)) + 0 + 1$(ord(j)=ord(l));

c2(i,j)$(SPAN(i,j) and (ord(i) < ord(j)))..
            sum((k,l), x(i,j,k,l)*D(k,l)) + sum((k,l), x(j,i,k,l)*D(k,l))
                     =L= sum(Cbl, z(Cbl,i,j)*Cbl_Cpy(Cbl));

c3(i,j,k,l)$(SPAN(i,j) and (ord(i) < ord (j)))..
            x(i,j,k,l) =L= y(i,j);

c4(i,j,k,l)$(SPAN(i,j) and (ord(i) > ord (j)))..
            x(i,j,k,l) =L= y(j,i);


  Model NetworkDesign /all/ ;

  NetworkDesign.holdfixed=1;

  Solve NetworkDesign using MIP minimizing cost;

  DISPLAY SPAN, D, Stp_Cst, Cbl_Prc, cost.l, x.l, y.l, z.l, TOTAL_DEMAND;
```

C.2 1+1 Protection Mesh Model

```
* Network Design Model.gms
* Feb. 16, 2006
* 14th Program: Included Network Data
*
* Fixed Cost, Cable Prices And Cable Capacities And Etc. From SNDLib Instances
* Cables In Fact Are Optical Carriers; Cable Prices Are Leased Fee
* A Mesh Network Model; An Undirected Demand Mesh
* Switches, MUXs And De-MUXs Are Equipped At All Nodes
* Equipments' Costs Are Not Considered
* Protection; But Use Only One Fiber
*
* x Is Between [0, 1]
```

```
* Non-used z And y Is Removed
*
$eolcom //
option iterlim=999999999; // avoid limit on iterations
option reslim=270000;     // timelimit for solver in sec.
option optcr=0.0;         // gap tolerance
option solprint=OFF;      // include solution print in .lst file
option limrow=0;          // limit number of rows in .lst file
option limcol=0;          // limit number of columns in .lst file
//-------------------------------------------------------------------------


  Sets
    Node                name of cities

    ALIAS(Node,i)
    ALIAS(Node,j)
    ALIAS(Node,k)
    ALIAS(Node,l);

  Set
    Cbl                 types of cables with different capacities;

  Parameters
    XX(Node)            x-bar of a city
    YY(Node)            y-bar of a city

    Cbl_Cpy(Cbl)        cable capacities
    Cbl_Prc(i,j,Cbl)    cable prices on link #ij#

    SPAN(i,j)           whether the link between #ij# exsits
    Pre_Cap(i,j)        pre-installed capacity already exsits on #ij#
    Pre_Cst(i,j)        how much have to pay on the pre-installed capacity
    Unt_Rt_Cst(i,j)     unit routing cost on #ij#
    Stp_Cst(i,j)        if not set up, how much should pay to set up

    D(k,l)              how much is the demand of flow between #kl#
    Rt_Unt(k,l)         the size of a routing unit on demand #kl#
    Dmd_Amt(k,l)        how much is the demand of unit between #kl#
    TOTAL_DEMAND        sum of demands of the entire network;

$INCLUDE "deutsch.dat";
```

```
// build a symmetric network
SPAN(i,j)$(ord(i)<ord(j)) = SPAN(i,j) + SPAN(j,i);
SPAN(i,j)$(ord(i)>ord(j)) = SPAN(j,i);

Pre_Cap(i,j)$(ord(i)<ord(j)) = Pre_Cap(i,j) + Pre_Cap(j,i);
Pre_Cap(i,j)$(ord(i)>ord(j)) = Pre_Cap(j,i);

Pre_Cst(i,j)$(ord(i)<ord(j)) = Pre_Cst(i,j) + Pre_Cst(j,i);
Pre_Cst(i,j)$(ord(i)>ord(j)) = Pre_Cst(j,i);

Unt_Rt_Cst(i,j)$(ord(i)<ord(j)) = Unt_Rt_Cst(i,j) + Unt_Rt_Cst(j,i);
Unt_Rt_Cst(i,j)$(ord(i)>ord(j)) = Unt_Rt_Cst(j,i);

Stp_Cst(i,j)$(ord(i)<ord(j)) = Stp_Cst(i,j) + Stp_Cst(j,i);
Stp_Cst(i,j)$(ord(i)>ord(j)) = Stp_Cst(j,i);

Cbl_Prc(i,j,Cbl)$(ord(i)<ord(j)) = Cbl_Prc(i,j,Cbl) + Cbl_Prc(j,i,Cbl);
Cbl_Prc(i,j,Cbl)$(ord(i)>ord(j)) = Cbl_Prc(j,i,Cbl);

// transfer data to demand matrix D, and make it upper triangular
// so here the demands are considered as undirected
D(k,l) = Dmd_Amt(k,l)*Rt_Unt(k,l)$(Rt_Unt(k,l)>0) + Dmd_Amt(k,l)$(Rt_Unt(k,l)=0);
TOTAL_DEMAND = sum((k,l), D(k,l));

D(k,l)$(ord(k)<ord(l)) = D(k,l) + D(l,k);
D(k,l)$(ord(k)>=ord(l)) = 0;

BINARY Variables
     y(i,j)     whether to build the arc #ij#;
// set all y to be #0#
     y.fx(i,j) = 0;
// set all y on half of the exsited links possible to be #1#
     y.up(i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = 1;

INTEGER Variables
     z(Cbl,i,j) whether to lease carrier #Cbl# in arc #ij#, and how many;
// set all z to be #0#
     z.fx(Cbl,i,j) = 0;
// set all z on half of the exsited links possible to be non-zero integer
     z.up(Cbl,i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = ceil (TOTAL_DEMAND/Cbl_Cpy(Cbl));

Variables
     cost         total cost
```

```
      x1(i,j,k,l)  how much percentage of flow through arc #ij# for demand #kl#, path 1
      x2(i,j,k,l)  how much percentage of flow through arc #ij# for demand #kl#, path 2;
  // set all x to be #0#
      x1.fx(i,j,k,l) = 0;
      x2.fx(i,j,k,l) = 0;
  // set all x on exsited links to have upper bound as #1#
      x1.up(i,j,k,l)$(SPAN(i,j) and (D(k,l) <> 0)) = 1;
      x2.up(i,j,k,l)$(SPAN(i,j) and (D(k,l) <> 0)) = 1;

  Equations
      obj               define objective function
      c1(j,k,l)         demand constraints
      c2(j,k,l)         demand constraints
      c3(i,j)           link capacity constraints
      c4(i,j,k,l)       different path constraints
      c5(i,j,k,l)
      c6(i,j,k,l)
      c7(i,j,k,l)
      c8(i,j,k,l)       ;

obj..       cost =E= sum((Cbl,i,j), z(Cbl,i,j)*Cbl_Prc(i,j,Cbl))
                     + sum((i,j), Stp_Cst(i,j)*y(i,j));

c1(j,k,l)$(D(k,l) <> 0)..
            sum(i, x1(i,j,k,l)) - sum(i, x1(j,i,k,l))
                     =E= -1$(ord(j)=ord(k)) + 0 + 1$(ord(j)=ord(l));

c2(j,k,l)$(D(k,l) <> 0)..
            sum(i, x2(i,j,k,l)) - sum(i, x2(j,i,k,l))
                     =E= -1$(ord(j)=ord(k)) + 0 + 1$(ord(j)=ord(l));

c3(i,j)$(SPAN(i,j) and (ord(i) < ord(j)))..
            sum((k,l), x1(i,j,k,l)*D(k,l)) + sum((k,l), x1(j,i,k,l)*D(k,l)) +
            sum((k,l), x2(i,j,k,l)*D(k,l)) + sum((k,l), x2(j,i,k,l)*D(k,l))
                     =L= sum(Cbl, z(Cbl,i,j)*Cbl_Cpy(Cbl));

c4(i,j,k,l)$SPAN(i,j)..
            x1(i,j,k,l) + x1(j,i,k,l) + x2(i,j,k,l) + x2(j,i,k,l) =L= 1;

c5(i,j,k,l)$(SPAN(i,j) and (ord(i) < ord (j)))..
            x1(i,j,k,l) =L= y(i,j);

c6(i,j,k,l)$(SPAN(i,j) and (ord(i) > ord (j)))..
```

```
            x1(i,j,k,l) =L= y(j,i);

c7(i,j,k,l)$(SPAN(i,j) and (ord(i) < ord (j)))..
            x2(i,j,k,l) =L= y(i,j);

c8(i,j,k,l)$(SPAN(i,j) and (ord(i) > ord (j)))..

            x2(i,j,k,l) =L= y(j,i);


  Model NetworkDesign /all/ ;

  NetworkDesign.holdfixed=1;

  Solve NetworkDesign using MIP minimizing cost;

  DISPLAY SPAN, D, Stp_Cst, Cbl_Prc, cost.l, x1.l, x2.l, y.l, z.l, TOTAL_DEMAND;
```

C.3 Link Protection Mesh Model

```
* Network Design Model.gms
* Mar. 15, 2006
* 15th Program: Included Network Data
*
* Cable Prices And Demands From SNDLib Instances
* Consider Cable Prices As A Kind Of Cost, Unit Flow Cost, On Links
* A Mesh Network Adds Rings Protection Model; An Undirected Demand Mesh
* Switches, MUXs And De-MUXs Are Equipped At All Nodes
* Equipments' Costs Are Not Considered
* Advanced 1+1 Protection
*
* Give Solutions Only For Protection
*
$eolcom //
option iterlim=999999999;// avoid limit on iterations
option reslim=270000;    // time limit for solver in sec.
option optcr=0.0;        // gap tolerance
option solprint=OFF;     // include solution print in .lst file
option limrow=0;         // limit number of rows in .lst file
option limcol=0;         // limit number of columns in .lst file
//-----------------------------------------------------------------------
```

```
Set
  Node                  name of cities
  ALIAS(Node,i)
  ALIAS(Node,j)
  ALIAS(Node,k)
  ALIAS(Node,l);

Set
  Pattern /P1*P10000/;
  ALIAS(Pattern, p);

Set
  sub_p(p);

Set                     // which is unused
  Cbl                   types of cables with different capacities;

Parameters
  SPAN(i,j)             whether the link between #ij# exsits
  DD(i,j)               how much flow goes between #ij#, given by shortest path
  C(i,j)                unit flow cost on link #ij#

  A(i,j,p)              column for the program
  Phi(i,j)              dual values for each link #ij#
  PATTERN_COST(p)       cost of pattern #p#, which is one or several rings
  MAS_O(p)              objective value of master programs
  SUB_O(p)              objective value of sub-programs;

Parameters              // which is unused
  XX(Node)              x-bar of a city
  YY(Node)              y-bar of a city

  Cbl_Cpy(Cbl)          cable capacities
  Cbl_Prc(i,j,Cbl)      cable prices on link #ij#

  Pre_Cap(i,j)          pre-installed capacity already exsits on #ij#
  Pre_Cst(i,j)          how much have to pay on the pre-installed capacity
  Unt_Rt_Cst(i,j)       unit routing cost on #ij#
  Stp_Cst(i,j)          if not set up, how much should pay to set up

  D(k,l)                how much is the demand of flow between #kl#
  Rt_Unt(k,l)           the size of a routing unit on demand #kl#
  Dmd_Amt(k,l)          how much is the demand of unit between #kl#
```

```
    TOTAL_DEMAND         sum of demands of the entire network;

$INCLUDE "Arne.dat";

  C(i,j) = 0;
  DD(i,j) = 0;

  // build a symmetric network
  SPAN(i,j)$(ord(i)<ord(j)) = SPAN(i,j) + SPAN(j,i);
  SPAN(i,j)$(ord(i)>ord(j)) = SPAN(j,i);

  Cbl_Prc(i,j,Cbl)$(ord(i)<ord(j)) = Cbl_Prc(i,j,Cbl) + Cbl_Prc(j,i,Cbl);
  Cbl_Prc(i,j,Cbl)$(ord(i)>ord(j)) = Cbl_Prc(j,i,Cbl);

  C(i,j) = sum(Cbl$(ord(Cbl) = 1), Cbl_Prc(i,j,Cbl)) / sum(Cbl$(ord(Cbl) = 1), Cbl_Cpy(Cbl));

  D(k,l) = Dmd_Amt(k,l)*Rt_Unt(k,l)$(Rt_Unt(k,l)>0) + Dmd_Amt(k,l)$(Rt_Unt(k,l)=0);
  TOTAL_DEMAND = sum((k,l), D(k,l));

  DD(i,j)$((ord(i)<ord(j)) and SPAN(i,j)) = round(uniform(0.6, 1.4)*TOTAL_DEMAND/card(Node));
  DD(i,j)$(ord(i)>ord(j)) = DD(j,i);

  // give the basis, which is a kind of dummy one
  A(i,j,p) = 0;
  PATTERN_COST(p) = 0;

  A(i,j,'P1')$SPAN(i,j) = 1;
  sub_p('P1') = YES;
  PATTERN_COST('P1') = 100*sum((i,j), C(i,j));

  // master problems
  //---------------------------------------------------------------------------
  Variable
       mas_obj;

  Positive Variable
       x(p);

  Equations
       MAS_OBJECTIVE
       MAS_CONSTRAINTS(i,j);

  MAS_OBJECTIVE..
```

```
     mas_obj =E= sum(sub_p, PATTERN_COST(sub_p)*x(sub_p));

MAS_CONSTRAINTS(i,j)$SPAN(i,j)..
     sum(sub_p, A(i,j,sub_p)*x(sub_p)) =G= DD(i,j);

Model MASTER /MAS_OBJECTIVE, MAS_CONSTRAINTS/;
//---------------------------------------------------------------------------

// sub-problems
//---------------------------------------------------------------------------
BINARY Variable
     y(i,j);
     y.fx(i,j) = 0;
     y.up(i,j)$SPAN(i,j) = 1;

Variable
     sub_obj;

Equation
     SUB_OBJECTIVE
     SUB_CONSTRAINTS1(i)
     SUB_CONSTRAINTS2(i,j);

SUB_OBJECTIVE..
     sub_obj =E= sum((i,j)$SPAN(i,j), (C(i,j) - Phi(i,j))*y(i,j));

SUB_CONSTRAINTS1(i)..
     sum(j, y(i,j)) - sum(j,y(j,i)) =E= 0;

SUB_CONSTRAINTS2(i,j)$SPAN(i,j)..
     y(i,j) + y(j,i) =L= 1;

Model SUB /SUB_OBJECTIVE, SUB_CONSTRAINTS1, SUB_CONSTRAINTS2/;
//---------------------------------------------------------------------------

File data /_Arn_Columns.res/;
Put data;

// column generation loop
//---------------------------------------------------------------------------
Parameter
  STOP
  Iteration;
```

```
STOP = 0;
Iteration = 0;

loop(p$(ord(p) > 1 and STOP = 0),

  solve MASTER using LP minimizing mas_obj;

  Phi(i,j)$SPAN(i,j) = MAS_CONSTRAINTS.m(i,j)$SPAN(i,j);
  Phi(i,j)$(ord(i)<ord(j)) = Phi(i,j)$(ord(i)<ord(j)) + Phi(j,i)$(ord(i)<ord(j));
  Phi(i,j)$(ord(i)>ord(j)) = Phi(j,i)$(ord(i)>ord(j));
  MAS_O(p) = mas_obj.l;
  display mas_obj.l, x.l;

  solve SUB using MIP minimizing sub_obj;

  SUB_O(p) = sub_obj.l;
  display sub_obj.l

  if( sub_obj.l >= -0.01,      // iteration stops
      STOP = 1;
  else                         // record the new-generated column, and extend the sub-set
      A(i,j,p)$SPAN(i,j) = y.l(i,j)$SPAN(i,j);
      A(i,j,p)$(ord(i)<ord(j)) = A(i,j,p)$(ord(i)<ord(j)) + A(j,i,p)$(ord(i)<ord(j));
      A(i,j,p)$(ord(i)>ord(j)) = A(j,i,p)$(ord(i)>ord(j));
      sub_p(p) = YES;
      PATTERN_COST(p) = sum((i,j)$SPAN(i,j), C(i,j)*A(i,j,p))/2;
      display A, PATTERN_COST;
    );

  iteration = iteration + 1;

// record the master and sub objective values into another file
  put 'iteration: ', iteration:<12:0, 'mas: ',  mas_obj.l:<12:2, '  sub: ', sub_obj.l:<12:2 /;

    );

putclose;

//---------------------------------------------------------------------

display x.l;
```

C.4 Non-Protective Ring Model

```
* Network Design Model.gms
* Feb. 14, 2006
* 14th Program: Included Network Data
*
* Fixed Cost, Cable Prices And Cable Capacities And Etc. From SNDLib Instances
* Cables In Fact Are Optical Carriers; Cable Prices Are Leased Fee
* A Ring Network Model; Model 1; An Undirected Demand Ring
* Switches, MUXs AND DE-MUXs Are Equipped At All Nodes
* Equipments' Costs Are Not Considered
* No Protection
*
* x Is Between [0, 1]
* Non-used z And y Is Removed
*
$eolcom //
option iterlim=999999999;  // avoid limit on iterations
option reslim=270000;      // timelimit for solver in sec.
option optcr=0.0;          // gap tolerance
option solprint=OFF;       // include solution print in .lst file
option limrow=0;           // limit number of rows in .lst file
option limcol=0;           // limit number of columns in .lst file
//-----------------------------------------------------------------------

  Sets
    Node                name of cities

    ALIAS(Node,i)
    ALIAS(Node,j)
    ALIAS(Node,k)
    ALIAS(Node,l);

  Set
    Cbl                 types of cables with different capacities;

  Parameters
    XX(Node)            x-bar of a city
    YY(Node)            y-bar of a city

    Cbl_Cpy(Cbl)        cable capacities
    Cbl_Prc(i,j,Cbl)    cable prices on link #ij#
```

```
    SPAN(i,j)            whether the link between #ij# exsits
    Pre_Cap(i,j)         pre-installed capacity already exsits on #ij#
    Pre_Cst(i,j)         how much have to pay on the pre-installed capacity
    Unt_Rt_Cst(i,j)      unit routing cost on #ij#
    Stp_Cst(i,j)         if not set up, how much should pay to set up

    D(k,l)               how much is the demand of flow between #kl#
    Rt_Unt(k,l)          the size of a routing unit on demand #kl#
    Dmd_Amt(k,l)         how much is the demand of unit between #kl#
    TOTAL_DEMAND         sum of demands of the entire network;

$INCLUDE "deutsch.dat";

  // build a symmetric network
  SPAN(i,j)$(ord(i)<ord(j)) = SPAN(i,j) + SPAN(j,i);
  SPAN(i,j)$(ord(i)>ord(j)) = SPAN(j,i);

  Pre_Cap(i,j)$(ord(i)<ord(j)) = Pre_Cap(i,j) + Pre_Cap(j,i);
  Pre_Cap(i,j)$(ord(i)>ord(j)) = Pre_Cap(j,i);

  Pre_Cst(i,j)$(ord(i)<ord(j)) = Pre_Cst(i,j) + Pre_Cst(j,i);
  Pre_Cst(i,j)$(ord(i)>ord(j)) = Pre_Cst(j,i);

  Unt_Rt_Cst(i,j)$(ord(i)<ord(j)) = Unt_Rt_Cst(i,j) + Unt_Rt_Cst(j,i);
  Unt_Rt_Cst(i,j)$(ord(i)>ord(j)) = Unt_Rt_Cst(j,i);

  Stp_Cst(i,j)$(ord(i)<ord(j)) = Stp_Cst(i,j) + Stp_Cst(j,i);
  Stp_Cst(i,j)$(ord(i)>ord(j)) = Stp_Cst(j,i);

  Cbl_Prc(i,j,Cbl)$(ord(i)<ord(j)) = Cbl_Prc(i,j,Cbl) + Cbl_Prc(j,i,Cbl);
  Cbl_Prc(i,j,Cbl)$(ord(i)>ord(j)) = Cbl_Prc(j,i,Cbl);

  // transfer data to demand matrix D, and make it upper triangular
  // so here the demands are considered as undirected
  D(k,l) = Dmd_Amt(k,l)*Rt_Unt(k,l)$(Rt_Unt(k,l)>0) + Dmd_Amt(k,l)$(Rt_Unt(k,l)=0);
  TOTAL_DEMAND = sum((k,l), D(k,l));

  D(k,l)$(ord(k)<ord(l)) = D(k,l) + D(l,k);
  D(k,l)$(ord(k)>=ord(l)) = 0;

  BINARY Variables
       y(i,j)     whether to build the arc #ij#;
  // set all y to be #0#
```

```
      y.fx(i,j) = 0;
  // set all y on half of the exsited links possible to be #1#
      y.up(i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = 1;

  INTEGER Variables
      z(Cbl,i,j) whether to lease carrier #Cbl# in arc #ij#, and how many;
  // set all z to be #0#
      z.fx(Cbl,i,j) = 0;
  // set all z on half of the exsited links possible to be non-zero integer
      z.up(Cbl,i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = ceil (TOTAL_DEMAND/Cbl_Cpy(Cbl));

  Variables
      cost         total cost
      x(i,j,k,l)   how much percentage of flow through arc #ij# for demand #kl#;
  // set all x to be #0#
      x.fx(i,j,k,l) = 0;
  // set all x on exsited links to have upper bound as #1#
      x.up(i,j,k,l)$(SPAN(i,j) and (D(k,l) <> 0)) = 1;

  Equations
      obj               define objective function
      c1(j,k,l)         demand constraints
      c2(i,j)
      c3(i,j,k,l)
      c4(i,j,k,l)
      c5(i)             ring constraints;

obj..       cost =E= sum((Cbl,i,j), z(Cbl,i,j)*Cbl_Prc(i,j,Cbl))
                       + sum((i,j), Stp_Cst(i,j)*y(i,j));

c1(j,k,l)$(D(k,l) <> 0)..
            sum(i, x(i,j,k,l)) - sum(i, x(j,i,k,l))
                     =E= -1$(ord(j)=ord(k)) + 0 + 1$(ord(j)=ord(l));

c2(i,j)$(SPAN(i,j) and (ord(i) < ord(j)))..
            sum((k,l), x(i,j,k,l)*D(k,l)) + sum((k,l), x(j,i,k,l)*D(k,l))
                     =L= sum(Cbl, z(Cbl,i,j)*Cbl_Cpy(Cbl));

c3(i,j,k,l)$(SPAN(i,j) and (ord(i) < ord (j)))..
            x(i,j,k,l) =L= y(i,j);

c4(i,j,k,l)$(SPAN(i,j) and (ord(i) > ord (j)))..
            x(i,j,k,l) =L= y(j,i);
```

```
c5(i)..     sum(j, y(i,j)) + sum(j, y(j,i)) =E= 2;


  Model NetworkDesign /all/ ;

  NetworkDesign.holdfixed=1;

  Solve NetworkDesign using MIP minimizing cost;

  DISPLAY SPAN, D, Stp_Cst, Cbl_Prc, cost.l, x.l, y.l, z.l;
```

C.5 1+1 Protection Ring Model

```
* Network Design Model.gms
* Feb. 14, 2006
* 14th Program: Included Network Data
*
* Fixed Cost, Cable Prices And Cable Capacities And Etc. From SNDLib Instances
* Cables In Fact Are Optical Carriers; Cable Prices Are Leased Fee
* A Ring Network Model; Model 2; An Undirected Demand Ring
* Switches, MUXs And De-MUXs Are Equipped At All Nodes
* Equipments' Costs Are Not Considered
* Protection; But Use Only One Fiber
*
* x Is Between [0, 1]
* Non-used z And y Is Removed
*
$eolcom //
option iterlim=999999999;// avoid limit on iterations
option reslim=270000;    // timelimit for solver in sec.
option optcr=0.0;        // gap tolerance
option solprint=OFF;     // include solution print in .lst file
option limrow=0;         // limit number of rows in .lst file
option limcol=0;         // limit number of columns in .lst file
//-------------------------------------------------------------------

  Sets
    Node                 name of cities

    ALIAS(Node,i)
    ALIAS(Node,j)
```

```
    ALIAS(Node,k)
    ALIAS(Node,l);

  Set
    Cbl                 types of cables with different capacities;

  Parameters
    XX(Node)            x-bar of a city
    YY(Node)            y-bar of a city

    Cbl_Cpy(Cbl)        cable capacities
    Cbl_Prc(i,j,Cbl)    cable prices on link #ij#

    SPAN(i,j)           whether the link between #ij# exsits
    Pre_Cap(i,j)        pre-installed capacity already exsits on #ij#
    Pre_Cst(i,j)        how much have to pay on the pre-installed capacity
    Unt_Rt_Cst(i,j)     unit routing cost on #ij#
    Stp_Cst(i,j)        if not set up, how much should pay to set up

    D(k,l)              how much is the demand of flow between #kl#
    Rt_Unt(k,l)         the size of a routing unit on demand #kl#
    Dmd_Amt(k,l)        how much is the demand of unit between #kl#
    TOTAL_DEMAND        sum of demands of the entire network;

$INCLUDE "deutsch.dat";

  // build a symmetric network
  SPAN(i,j)$(ord(i)<ord(j)) = SPAN(i,j) + SPAN(j,i);
  SPAN(i,j)$(ord(i)>ord(j)) = SPAN(j,i);

  Pre_Cap(i,j)$(ord(i)<ord(j)) = Pre_Cap(i,j) + Pre_Cap(j,i);
  Pre_Cap(i,j)$(ord(i)>ord(j)) = Pre_Cap(j,i);

  Pre_Cst(i,j)$(ord(i)<ord(j)) = Pre_Cst(i,j) + Pre_Cst(j,i);
  Pre_Cst(i,j)$(ord(i)>ord(j)) = Pre_Cst(j,i);

  Unt_Rt_Cst(i,j)$(ord(i)<ord(j)) = Unt_Rt_Cst(i,j) + Unt_Rt_Cst(j,i);
  Unt_Rt_Cst(i,j)$(ord(i)>ord(j)) = Unt_Rt_Cst(j,i);

  Stp_Cst(i,j)$(ord(i)<ord(j)) = Stp_Cst(i,j) + Stp_Cst(j,i);
  Stp_Cst(i,j)$(ord(i)>ord(j)) = Stp_Cst(j,i);

  Cbl_Prc(i,j,Cbl)$(ord(i)<ord(j)) = Cbl_Prc(i,j,Cbl) + Cbl_Prc(j,i,Cbl);
```

```
Cbl_Prc(i,j,Cbl)$(ord(i)>ord(j)) = Cbl_Prc(j,i,Cbl);

// transfer data to demand matrix D, and make it upper triangular
// so here the demands are considered as undirected
D(k,l) = Dmd_Amt(k,l)*Rt_Unt(k,l)$(Rt_Unt(k,l)>0) + Dmd_Amt(k,l)$(Rt_Unt(k,l)=0);
TOTAL_DEMAND = sum((k,l), D(k,l));

D(k,l)$(ord(k)<ord(l)) = D(k,l) + D(l,k);
D(k,l)$(ord(k)>=ord(l)) = 0;

BINARY Variables
     y(i,j)     whether to build the arc #ij#;
// set all y to be #0#
     y.fx(i,j) = 0;
// set all y on half of the exsited links possible to be #1#
     y.up(i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = 1;

INTEGER Variables
     z(Cbl,i,j) whether to lease carrier #Cbl# in arc #ij#, and how many;
// set all z to be #0#
     z.fx(Cbl,i,j) = 0;
// set all z on half of the exsited links possible to be non-zero integer
     z.up(Cbl,i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = ceil (TOTAL_DEMAND/Cbl_Cpy(Cbl));

Variables
     cost         total cost
     x1(i,j,k,l)  how much percentage of flow through arc #ij# for demand #kl#, path 1
     x2(i,j,k,l)  how much percentage of flow through arc #ij# for demand #kl#, path 2;
// set all x to be #0#
     x1.fx(i,j,k,l) = 0;
     x2.fx(i,j,k,l) = 0;
// set all x on exsited links to have upper bound as #1#
     x1.up(i,j,k,l)$(SPAN(i,j) and (D(k,l) <> 0)) = 1;
     x2.up(i,j,k,l)$(SPAN(i,j) and (D(k,l) <> 0)) = 1;

Equations
     obj               define objective function
     c1(j,k,l)         demand constraints
     c2(j,k,l)         demand constraints
     c3(i,j)
     c4(i,j,k,l)       different path constraints
     c5(i,j,k,l)
     c6(i,j,k,l)
```

```
       c7(i,j,k,l)
       c8(i,j,k,l)
       c9(i)             ring constraints;

obj..       cost =E= sum((Cbl,i,j), z(Cbl,i,j)*Cbl_Prc(i,j,Cbl))
                       + sum((i,j), Stp_Cst(i,j)*y(i,j));

c1(j,k,l)$(D(k,l) <> 0)..
            sum(i, x1(i,j,k,l)) - sum(i, x1(j,i,k,l))
                     =E= -1$(ord(j)=ord(k)) + 0 + 1$(ord(j)=ord(l));

c2(j,k,l)$(D(k,l) <> 0)..
            sum(i, x2(i,j,k,l)) - sum(i, x2(j,i,k,l))
                     =E= -1$(ord(j)=ord(k)) + 0 + 1$(ord(j)=ord(l));

c3(i,j)$(SPAN(i,j) and (ord(i) < ord(j)))..
            sum((k,l), x1(i,j,k,l)*D(k,l)) + sum((k,l), x1(j,i,k,l)*D(k,l)) +
            sum((k,l), x2(i,j,k,l)*D(k,l)) + sum((k,l), x2(j,i,k,l)*D(k,l))
                     =L= sum(Cbl, z(Cbl,i,j)*Cbl_Cpy(Cbl));

c4(i,j,k,l)$SPAN(i,j)..
            x1(i,j,k,l) + x1(j,i,k,l) + x2(i,j,k,l) + x2(j,i,k,l) =L= 1;

c5(i,j,k,l)$(SPAN(i,j) and (ord(i) < ord (j)))..
            x1(i,j,k,l) =L= y(i,j);

c6(i,j,k,l)$(SPAN(i,j) and (ord(i) > ord (j)))..
            x1(i,j,k,l) =L= y(j,i);

c7(i,j,k,l)$(SPAN(i,j) and (ord(i) < ord (j)))..
            x2(i,j,k,l) =L= y(i,j);

c8(i,j,k,l)$(SPAN(i,j) and (ord(i) > ord (j)))..
            x2(i,j,k,l) =L= y(j,i);

c9(i)..     sum(j, y(i,j)) + sum(j, y(j,i)) =E= 2;


  Model NetworkDesign /all/ ;

  NetworkDesign.holdfixed=1;

  Solve NetworkDesign using MIP minimizing cost;
```

```
  DISPLAY SPAN, D, Stp_Cst, Cbl_Prc, cost.l, x1.l, x2.l, y.l, z.l, TOTAL_DEMAND;
```

C.6 1+1 Protection Ring(TSP) Model

```
* Network Design Model.gms
* Feb. 14, 2006
* 14th Program: Included Network Data
*
* Fixed Cost, Cable Prices And Cable Capacities And Etc. From SNDLib Instances
* Cables In Fact Are Optical Carriers; Cable Prices Are Leased Fee
* A Ring Network Model; Model 2; An Undirected Demand Ring
* Switches, MUXs And De-MUXs Are Equipped At All Nodes
* Equipments' Costs Are Not Considered
* Protection; But Use Only One Fiber
* A TSP Problem
*
* x Is Between [0, 1]
* Non-used z And y Is Removed
*
$eolcom //
option iterlim=999999999;  // avoid limit on iterations
option reslim=100000;      // timelimit for solver in sec.
option optcr=0.0;          // gap tolerance
option solprint=OFF;       // include solution print in .lst file
option limrow=0;           // limit number of rows in .lst file
option limcol=0;           // limit number of columns in .lst file
//-------------------------------------------------------------------------

  Sets
    Node                name of cities

    ALIAS(Node,i)
    ALIAS(Node,j)
    ALIAS(Node,k)
    ALIAS(Node,l);

  Set
    Cbl                 types of cables with different capacities;

  Parameters
    XX(Node)            x-bar of a city
```

```
    YY(Node)              y-bar of a city

    Cbl_Cpy(Cbl)          cable capacities
    Cbl_Prc(i,j,Cbl)      cable prices on link #ij#

    SPAN(i,j)             whether the link between #ij# exsits
    Pre_Cap(i,j)          pre-installed capacity already exsits on #ij#
    Pre_Cst(i,j)          how much have to pay on the pre-installed capacity
    Unt_Rt_Cst(i,j)       unit routing cost on #ij#
    Stp_Cst(i,j)          if not set up, how much should pay to set up

    D(k,l)                how much is the demand of flow between #kl#
    Rt_Unt(k,l)           the size of a routing unit on demand #kl#
    Dmd_Amt(k,l)          how much is the demand of unit between #kl#
    TOTAL_DEMAND          sum of demands of the entire network
    z(Cbl)                the cables used on every link;

$INCLUDE "deutsch.dat";

  // build a symmetric network
  SPAN(i,j)$(ord(i)<ord(j)) = SPAN(i,j) + SPAN(j,i);
  SPAN(i,j)$(ord(i)>ord(j)) = SPAN(j,i);

  Pre_Cap(i,j)$(ord(i)<ord(j)) = Pre_Cap(i,j) + Pre_Cap(j,i);
  Pre_Cap(i,j)$(ord(i)>ord(j)) = Pre_Cap(j,i);

  Pre_Cst(i,j)$(ord(i)<ord(j)) = Pre_Cst(i,j) + Pre_Cst(j,i);
  Pre_Cst(i,j)$(ord(i)>ord(j)) = Pre_Cst(j,i);

  Unt_Rt_Cst(i,j)$(ord(i)<ord(j)) = Unt_Rt_Cst(i,j) + Unt_Rt_Cst(j,i);
  Unt_Rt_Cst(i,j)$(ord(i)>ord(j)) = Unt_Rt_Cst(j,i);

  Stp_Cst(i,j)$(ord(i)<ord(j)) = Stp_Cst(i,j) + Stp_Cst(j,i);
  Stp_Cst(i,j)$(ord(i)>ord(j)) = Stp_Cst(j,i);

  Cbl_Prc(i,j,Cbl)$(ord(i)<ord(j)) = Cbl_Prc(i,j,Cbl) + Cbl_Prc(j,i,Cbl);
  Cbl_Prc(i,j,Cbl)$(ord(i)>ord(j)) = Cbl_Prc(j,i,Cbl);

  // calculate the bandwidth which is needed on every link
  D(k,l) = Dmd_Amt(k,l)*Rt_Unt(k,l)$(Rt_Unt(k,l)>0) + Dmd_Amt(k,l)$(Rt_Unt(k,l)=0);
  TOTAL_DEMAND = sum((k,l), D(k,l));

  z(Cbl) = 0;
```

```
scalar IT;
IT = card(Cbl);

scalar TD;
TD = TOTAL_DEMAND;

while ( IT > 0,

        loop( Cbl$(ord(Cbl) = IT),

              z(Cbl) = floor(TD/Cbl_Cpy(Cbl));
              TD = mod(TD, Cbl_Cpy(Cbl));

              if ( IT = 1,
                   z(Cbl) = z(Cbl) + 1;
                   if ( (Cbl_Cpy(Cbl+1)/(z(Cbl)*Cbl_Cpy(Cbl)) > 0.95)
                        and (Cbl_Cpy(Cbl+1)/(z(Cbl)*Cbl_Cpy(Cbl)) < 1.05),
                        z(Cbl) = 0;
                        z(Cbl+1) = z(Cbl+1) + 1;
                      );
                 );
            );

        IT = IT - 1;
      );

BINARY Variables
     y(i,j)     whether to build the arc #ij#;
// set all y to be #0#
     y.fx(i,j) = 0;
// set all y on half of the exsited links possible to be #1#
     y.up(i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = 1;

Variables
     cost         total cost
     x(i,j,k,l)   how much percentage of flow through arc #ij# for demand #kl#;
// set all x to be #0#
     x.fx(i,j,k,l) = 0;
// set all x for demand #node1 node*# possible to be 1;
     x.up(i,j,k,l)$(SPAN(i,j) and (ord(k) = 1) and (ord(l) <> 1)) = 1;

Equations
```

```
       obj               define objective function
       c1(j,k,l)         demand constraints or sub-circles elimination constraints
       c2(i,j,k,l)
       c3(i,j,k,l)
       c4(i)             ring constraints;

obj..       cost =E= sum((i,j), y(i,j)*( sum(Cbl, z(Cbl)*Cbl_Prc(i,j,Cbl)) + Stp_Cst(i,j) ));

c1(j,k,l)$((ord(k) = 1) and (ord(l) <> 1))..
            sum(i, x(i,j,k,l)) - sum(i, x(j,i,k,l))
                     =E= -1$(ord(j)=ord(k)) + 0 + 1$(ord(j)=ord(l));

c2(i,j,k,l)$(SPAN(i,j) and (ord(i) < ord (j)))..
            x(i,j,k,l) =L= y(i,j);

c3(i,j,k,l)$(SPAN(i,j) and (ord(i) > ord (j)))..
            x(i,j,k,l) =L= y(j,i);

c4(i)..     sum(j, y(i,j)) + sum(j, y(j,i)) =E= 2;


  Model NetworkDesign /all/ ;

  NetworkDesign.holdfixed=1;

  Solve NetworkDesign using MIP minimizing cost;

  DISPLAY SPAN, D, Stp_Cst, Cbl_Prc, TOTAL_DEMAND, Cbl_Cpy, z;

  DISPLAY cost.l, y.l;
```

C.7 Two-Fibre Unidirectional Self-Healing Ring Model

```
* Network Design Model.gms
* Feb. 14, 2006
* 14th Program: Included Network Data
*
* Fixed Cost, Cable Prices And Cable Capacities And Etc. From SNDLib Instances
* Cables In Fact Are Optical Carriers; Cable Prices Are Leased Fee
* A Ring Network Model; Model 3; An Directed Demand Ring
```

```
* Switches, ADMs Are Equipped At All Nodes
* Equipments' Costs Are Not Considered
* Two-Fibre Unidirectional Self-Healing Protection
*
* x Is Between [0, 1]
* Non-used z And y Is Removed
*
$eolcom //
option iterlim=999999999; // avoid limit on iterations
option reslim=720000;     // timelimit for solver in sec.
option optcr=0.0;         // gap tolerance
option solprint=OFF;      // include solution print in .lst file
option limrow=0;          // limit number of rows in .lst file
option limcol=0;          // limit number of columns in .lst file
//-------------------------------------------------------------------------

  Sets
    Node                name of cities

    ALIAS(Node,i)
    ALIAS(Node,j)
    ALIAS(Node,k)
    ALIAS(Node,l);

  Set
    Cbl                 types of cables with different capacities;

  Parameters
    XX(Node)            x-bar of a city
    YY(Node)            y-bar of a city

    Cbl_Cpy(Cbl)        cable capacities
    Cbl_Prc(i,j,Cbl)    cable prices on link #ij#

    SPAN(i,j)           whether the link between #ij# exsits
    Pre_Cap(i,j)        pre-installed capacity already exsits on #ij#
    Pre_Cst(i,j)        how much have to pay on the pre-installed capacity
    Unt_Rt_Cst(i,j)     unit routing cost on #ij#
    Stp_Cst(i,j)        if not set up, how much should pay to set up

    D(k,l)              how much is the demand of flow between #kl#
    Rt_Unt(k,l)         the size of a routing unit on demand #kl#
    Dmd_Amt(k,l)        how much is the demand of unit between #kl#
```

```
    TOTAL_DEMAND         sum of demands of the entire network;

$INCLUDE "deutsch.dat";

  // build a symmetric network
  SPAN(i,j)$(ord(i)<ord(j)) = SPAN(i,j) + SPAN(j,i);
  SPAN(i,j)$(ord(i)>ord(j)) = SPAN(j,i);

  Pre_Cap(i,j)$(ord(i)<ord(j)) = Pre_Cap(i,j) + Pre_Cap(j,i);
  Pre_Cap(i,j)$(ord(i)>ord(j)) = Pre_Cap(j,i);

  Pre_Cst(i,j)$(ord(i)<ord(j)) = Pre_Cst(i,j) + Pre_Cst(j,i);
  Pre_Cst(i,j)$(ord(i)>ord(j)) = Pre_Cst(j,i);

  Unt_Rt_Cst(i,j)$(ord(i)<ord(j)) = Unt_Rt_Cst(i,j) + Unt_Rt_Cst(j,i);
  Unt_Rt_Cst(i,j)$(ord(i)>ord(j)) = Unt_Rt_Cst(j,i);

  Stp_Cst(i,j)$(ord(i)<ord(j)) = Stp_Cst(i,j) + Stp_Cst(j,i);
  Stp_Cst(i,j)$(ord(i)>ord(j)) = Stp_Cst(j,i);

  Cbl_Prc(i,j,Cbl)$(ord(i)<ord(j)) = Cbl_Prc(i,j,Cbl) + Cbl_Prc(j,i,Cbl);
  Cbl_Prc(i,j,Cbl)$(ord(i)>ord(j)) = Cbl_Prc(j,i,Cbl);

  // transfer data to demand matrix D
  // so here the demands are considered as directed
  D(k,l) = Dmd_Amt(k,l)*Rt_Unt(k,l)$(Rt_Unt(k,l)>0) + Dmd_Amt(k,l)$(Rt_Unt(k,l)=0);
  TOTAL_DEMAND = sum((k,l), D(k,l));

  D(k,l)$(ord(k)<ord(l)) = D(k,l) + D(l,k);
  D(k,l)$(ord(k)>ord(l)) = round(uniform(0.3, 0.7)*D(l,k)$(ord(k)>ord(l)));
  D(k,l)$(ord(k)<ord(l)) = D(k,l)$(ord(k)<ord(l)) - D(l,k)$(ord(k)<ord(l));
  D(k,l)$(ord(k)=ord(l)) = 0;

  BINARY Variables
       y(i,j)      whether to build the arc #ij#;
  // set all y to be #0#
       y.fx(i,j) = 0;
  // set all y on the exsited links possible to be #1#
       y.up(i,j)$SPAN(i,j) = 1;

  INTEGER Variables
       z1(Cbl,i,j) whether to lease carrier #Cbl# in arc #ij# for active ring, and how many
       z2(Cbl,i,j) whether to lease carrier #Cbl# in arc #ij# for protect ring, and how many;
```

```
  // set all z1 to be #0#
       z1.fx(Cbl,i,j) = 0;
  // set all z1 on half of the exsited links possible to be non-zero integer
       z1.up(Cbl,i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = ceil (TOTAL_DEMAND/Cbl_Cpy(Cbl));
  // set all z2 to be #0#
       z2.fx(Cbl,i,j) = 0;
  // set all z2 on half of the exsited links possible to be non-zero integer
       z2.up(Cbl,i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = ceil (TOTAL_DEMAND/Cbl_Cpy(Cbl));

  BINARY Variables
       x(i,j,k,l)    how much percentage of active flow through arc #ij# for demand #kl#;
  // set all x to be #0#
       x.fx(i,j,k,l) = 0;
  // set all x on exsited links to have upper bound as #1#
       x.up(i,j,k,l)$(SPAN(i,j) and (D(k,l) <> 0)) = 1;

  Variables
       cost         total cost;

  Equations
       obj               define objective function
       c1(j,k,l)         demand constraints
       c2(i,j)           active capacity
       c3(i,j)           protection capacity
       c4(i,j,k,l)       unidirectional flow constraints
       c5(i)             unidirectional ring constraints
       c6(i)             unidirectional ring constraints;

obj..       cost =E= sum((Cbl,i,j), z1(Cbl,i,j)*Cbl_Prc(i,j,Cbl))
                   + sum((Cbl,i,j), z2(Cbl,i,j)*Cbl_Prc(i,j,Cbl))
                   + sum((i,j), Stp_Cst(i,j)*y(i,j));

c1(j,k,l)$(D(k,l) <> 0)..
            sum(i, x(i,j,k,l)) - sum(i, x(j,i,k,l))
                     =E= -1$(ord(j)=ord(k)) + 0 + 1$(ord(j)=ord(l));

c2(i,j)$(SPAN(i,j) and (ord(i) < ord(j)))..
            sum((k,l), x(i,j,k,l)*D(k,l)) + sum((k,l), x(j,i,k,l)*D(k,l))
                     =L= sum(Cbl, z1(Cbl,i,j)*Cbl_Cpy(Cbl));

c3(i,j)$(SPAN(i,j) and (ord(i) < ord(j)))..
            sum((k,l), x(i,j,k,l)*D(l,k)) + sum((k,l), x(j,i,k,l)*D(l,k))
                     =L= sum(Cbl, z2(Cbl,i,j)*Cbl_Cpy(Cbl));
```

```
c4(i,j,k,l)$(SPAN(i,j) and (D(k,l) <> 0))..
            x(i,j,k,l) + x(i,j,l,k) =E= y(i,j);

c5(i)..     sum(j, y(i,j)) =E= 1;

c6(i)..     sum(j, y(j,i)) =E= 1;


  Model NetworkDesign /all/ ;

  NetworkDesign.holdfixed=1;

  Solve NetworkDesign using MIP minimizing cost;

  DISPLAY SPAN, D, Stp_Cst, Cbl_Prc, cost.l, x.l, y.l, z1.l, z2.l;
```

C.8 Tree Model

```
* Network Design Model.gms
* Feb. 22, 2006
* 14th Program: Included Network Data
*
* Fixed Cost, Cable Prices And Cable Capacities And Etc. From SNDLib Instances
* A Tree Network Model; LAN Switches Are Equipped At Some Nodes
* Hub Node Is Given
* No Protection
*
* x Is Between [0, 1]
* Non-used z And y Is Removed
*
$eolcom //
option iterlim=999999999; // avoid limit on iterations
option reslim=270000;     // timelimit for solver in sec.
option optcr=0.0;         // gap tolerance
option solprint=OFF;      // include solution print in .lst file
option limrow=0;          // limit number of rows in .lst file
option limcol=0;          // limit number of columns in .lst file
//-------------------------------------------------------------------------

  Sets
    Node                 name of cities
```

```
    ALIAS(Node,i)
    ALIAS(Node,j)
    ALIAS(Node,k)
    ALIAS(Node,l);

  Set
    Cbl                 types of cables with different capacities;

  Parameters
    XX(Node)            x-bar of a city
    YY(Node)            y-bar of a city

    Cbl_Cpy(Cbl)        cable capacities
    Cbl_Prc(i,j,Cbl)    cable prices on link #ij#

    SPAN(i,j)           whether the link between #ij# exsits
    Pre_Cap(i,j)        pre-installed capacity already exsits on #ij#
    Pre_Cst(i,j)        how much have to pay on the pre-installed capacity
    Unt_Rt_Cst(i,j)     unit routing cost on #ij#
    Stp_Cst(i,j)        if not set up, how much should pay to set up

    D(k,l)              how much is the demand of flow between #kl#
    Rt_Unt(k,l)         the size of a routing unit on demand #kl#
    Dmd_Amt(k,l)        how much is the demand of unit between #kl#
    TOTAL_DEMAND        sum of demands of the entire network;

$INCLUDE "deutsch.dat";

  // build a symmetric network
  SPAN(i,j)$(ord(i)<ord(j)) = SPAN(i,j) + SPAN(j,i);
  SPAN(i,j)$(ord(i)>ord(j)) = SPAN(j,i);

  Pre_Cap(i,j)$(ord(i)<ord(j)) = Pre_Cap(i,j) + Pre_Cap(j,i);
  Pre_Cap(i,j)$(ord(i)>ord(j)) = Pre_Cap(j,i);

  Pre_Cst(i,j)$(ord(i)<ord(j)) = Pre_Cst(i,j) + Pre_Cst(j,i);
  Pre_Cst(i,j)$(ord(i)>ord(j)) = Pre_Cst(j,i);

  Unt_Rt_Cst(i,j)$(ord(i)<ord(j)) = Unt_Rt_Cst(i,j) + Unt_Rt_Cst(j,i);
  Unt_Rt_Cst(i,j)$(ord(i)>ord(j)) = Unt_Rt_Cst(j,i);

  Stp_Cst(i,j)$(ord(i)<ord(j)) = Stp_Cst(i,j) + Stp_Cst(j,i);
```

```
Stp_Cst(i,j)$(ord(i)>ord(j)) = Stp_Cst(j,i);

Cbl_Prc(i,j,Cbl)$(ord(i)<ord(j)) = Cbl_Prc(i,j,Cbl) + Cbl_Prc(j,i,Cbl);
Cbl_Prc(i,j,Cbl)$(ord(i)>ord(j)) = Cbl_Prc(j,i,Cbl);

// transfer data to demand matrix D, and make it upper triangular
// so here the demands are considered as undirected
D(k,l) = Dmd_Amt(k,l)*Rt_Unt(k,l)$(Rt_Unt(k,l)>0) + Dmd_Amt(k,l)$(Rt_Unt(k,l)=0);
TOTAL_DEMAND = sum((k,l), D(k,l));

D(k,l)$(ord(k)<ord(l)) = D(k,l) + D(l,k);
D(k,l)$(ord(k)>=ord(l)) = 0;

SCALAR LAN_S_C  LAN switch cost;
LAN_S_C = 400;

BINARY Variables
     s(i)       whether to equip a LAN switch at node #i#
     y(i,j)     whether to build the arc #ij#;
// set all y to be #0#
     y.fx(i,j) = 0;
// set all y on half of the exsited links possible to be #1#
     y.up(i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = 1;

INTEGER Variables
     z(Cbl,i,j) whether to lease carrier #Cbl# in arc #ij#, and how many;
// set all z to be #0#
     z.fx(Cbl,i,j) = 0;
// set all z on half of the exsited links possible to be non-zero integer
     z.up(Cbl,i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = ceil (TOTAL_DEMAND/Cbl_Cpy(Cbl));

Variables
     cost         total cost
     x(i,j,k,l)   how much percentage of flow through arc #ij# for demand #kl#;
// set all x to be #0#
     x.fx(i,j,k,l) = 0;
// set all x on exsited links to have upper bound as #1#
     x.up(i,j,k,l)$(SPAN(i,j) and (D(k,l) <> 0)) = 1;

Equations
     obj               define objective function
     c1(j,k,l)         demand constraints
     c2(i,j)           link capacity constraints
```

```
       c3(i,j,k,l)
       c4(i,j,k,l)
       c5                hub management constraint
       c6(i)             switch constraints
       c7                tree constraint;

obj..       cost =E= sum((Cbl,i,j), z(Cbl,i,j)*Cbl_Prc(i,j,Cbl))
                       + sum((i,j), Stp_Cst(i,j)*y(i,j))
                       + (sum(i, s(i)) - 1)*LAN_S_C;

c1(j,k,l)$(D(k,l) <> 0)..
            sum(i, x(i,j,k,l)) - sum(i, x(j,i,k,l))
                     =E= -1$(ord(j)=ord(k)) + 0 + 1$(ord(j)=ord(l));

c2(i,j)$(SPAN(i,j) and (ord(i) < ord(j))).
            sum((k,l), x(i,j,k,l)*D(k,l)) + sum((k,l), x(j,i,k,l)*D(k,l))
                     =L= sum(Cbl, z(Cbl,i,j)*Cbl_Cpy(Cbl));

c3(i,j,k,l)$(SPAN(i,j) and (ord(i) < ord (j))).
            x(i,j,k,l) =L= y(i,j);

c4(i,j,k,l)$(SPAN(i,j) and (ord(i) > ord (j))).
            x(i,j,k,l) =L= y(j,i);

c5..        sum(j, y('E',j)) + sum(j, y(j,'E')) =E= 3;

c6(i)..     sum(j, y(i,j)) + sum(j, y(j,i)) =L= s(i)*card(Node) + 1;

c7..        sum((i,j), y(i,j)) =E= card(Node) - 1;


  Model NetworkDesign /all/ ;

  NetworkDesign.holdfixed=1;

  Solve NetworkDesign using MIP minimizing cost;

  DISPLAY SPAN, D, Stp_Cst, Cbl_Prc, cost.l, x.l, y.l, s.l, z.l;
```

C.9 Star Model(No Multiplexing)

```
* Network Design Model.gms
```

```
* Feb. 22, 2006
* 14th Program: Included Network Data
*
* Fixed Cost, Cable Prices And Cable Capacities And Etc. From SNDLib Instances
* A Star Network Model; No Equipment At Any Node; No Multiplexing
* Hub Node Is Given
* No Protection
*
* x Is Between [0, 1]
* Non-used z And y Is Removed
*
$eolcom //
option iterlim=999999999; // avoid limit on iterations
option reslim=270000;     // timelimit for solver in sec.
option optcr=0.0;         // gap tolerance
option solprint=OFF;      // include solution print in .lst file
option limrow=0;          // limit number of rows in .lst file
option limcol=0;          // limit number of columns in .lst file
//-------------------------------------------------------------------------

  Sets
    Node                name of cities

    ALIAS(Node,i)
    ALIAS(Node,j)
    ALIAS(Node,k)
    ALIAS(Node,l);

  Set
    Cbl                 types of cables with different capacities;

  Parameters
    XX(Node)            x-bar of a city
    YY(Node)            y-bar of a city

    Cbl_Cpy(Cbl)        cable capacities
    Cbl_Prc(i,j,Cbl)    cable prices on link #ij#

    SPAN(i,j)           whether the link between #ij# exsits
    Pre_Cap(i,j)        pre-installed capacity already exsits on #ij#
    Pre_Cst(i,j)        how much have to pay on the pre-installed capacity
    Unt_Rt_Cst(i,j)     unit routing cost on #ij#
    Stp_Cst(i,j)        if not set up, how much should pay to set up
```

```
    D(k,l)              how much is the demand of flow between #kl#
    Rt_Unt(k,l)         the size of a routing unit on demand #kl#
    Dmd_Amt(k,l)        how much is the demand of unit between #kl#
    TOTAL_DEMAND        sum of demands of the entire network

    Hub(i)              which node is the hub
    DH(k)               the demand between node #i# and the hub
    zH(k,Cbl)           the cables used for every DH #k#;

$INCLUDE "deutsch.dat";

  // build a symmetric network
  SPAN(i,j)$(ord(i)<ord(j)) = SPAN(i,j) + SPAN(j,i);
  SPAN(i,j)$(ord(i)>ord(j)) = SPAN(j,i);

  Pre_Cap(i,j)$(ord(i)<ord(j)) = Pre_Cap(i,j) + Pre_Cap(j,i);
  Pre_Cap(i,j)$(ord(i)>ord(j)) = Pre_Cap(j,i);

  Pre_Cst(i,j)$(ord(i)<ord(j)) = Pre_Cst(i,j) + Pre_Cst(j,i);
  Pre_Cst(i,j)$(ord(i)>ord(j)) = Pre_Cst(j,i);

  Unt_Rt_Cst(i,j)$(ord(i)<ord(j)) = Unt_Rt_Cst(i,j) + Unt_Rt_Cst(j,i);
  Unt_Rt_Cst(i,j)$(ord(i)>ord(j)) = Unt_Rt_Cst(j,i);

  Stp_Cst(i,j)$(ord(i)<ord(j)) = Stp_Cst(i,j) + Stp_Cst(j,i);
  Stp_Cst(i,j)$(ord(i)>ord(j)) = Stp_Cst(j,i);

  Cbl_Prc(i,j,Cbl)$(ord(i)<ord(j)) = Cbl_Prc(i,j,Cbl) + Cbl_Prc(j,i,Cbl);
  Cbl_Prc(i,j,Cbl)$(ord(i)>ord(j)) = Cbl_Prc(j,i,Cbl);

  // transfer data to demand matrix D, and make it upper triangular
  // so here the demands are considered as undirected
  D(k,l) = Dmd_Amt(k,l)*Rt_Unt(k,l)$(Rt_Unt(k,l)>0) + Dmd_Amt(k,l)$(Rt_Unt(k,l)=0);
  TOTAL_DEMAND = sum((k,l), D(k,l));

  D(k,l)$(ord(k)<ord(l)) = D(k,l) + D(l,k);
  D(k,l)$(ord(k)>=ord(l)) = 0;

  // preparation for a star
  Hub(i) = 0;
  Hub('E') = 1;
  DH(k) = sum(l, D(k,l)) + sum(l, D(l,k));
```

```
DH('E') = 0;

zH(k,Cbl) = 0;

scalar IT;

parameter DHH(k);
DHH(k) = DH(k);

loop  ( k$(ord(k) <> 5),

IT = card(Cbl);

while ( IT > 0,

        loop( Cbl$(ord(Cbl) = IT),

              zH(k,Cbl) = floor(DHH(k)/Cbl_Cpy(Cbl));
              DHH(k) = mod(DHH(k), Cbl_Cpy(Cbl));

              if ( IT = 1,
                   zH(k,Cbl) = zH(k,Cbl) + 1;
                   if ( (Cbl_Cpy(Cbl+1)/(zH(k,Cbl)*Cbl_Cpy(Cbl)) > 0.95)
                        and (Cbl_Cpy(Cbl+1)/(zH(k,Cbl)*Cbl_Cpy(Cbl)) < 1.05),
                        zH(k,Cbl) = 0;
                        zH(k,Cbl+1) = zH(k,Cbl+1) + 1;
                      );
                 );
            );

        IT = IT - 1;
      );

      );

BINARY Variables
     y(i,j)       whether to build the arc #ij#;
// set all y to be #0#
     y.fx(i,j) = 0;
// set all y on half of the exsited links possible to be #1#
     y.up(i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = 1;

Variables
```

```
       cost         total cost
       x(i,j,k)     how much percentage of flow through arc #ij# for DH #k#;

  // set all x to be #0#
       x.fx(i,j,k) = 0;
  // set all x on exsited links to have upper bound as #1#
       x.up(i,j,k)$(SPAN(i,j) and (DH(k) <> 0)) = 1;

  Equations
       obj               define objective function
       c1(j,k)           demand constraints
       c2(i,j,k)
       c3(i,j,k)
       c4                tree constraint;

obj..       cost =E= sum((k,Cbl,i,j), zH(k,Cbl)*x(i,j,k)*Cbl_Prc(i,j,Cbl))
                   + sum((i,j), Stp_Cst(i,j)*y(i,j));

c1(j,k)$(DH(k) <> 0)..
            sum(i, x(i,j,k)) - sum(i, x(j,i,k))
                     =E= -1$(ord(j)=ord(k)) + 0 + 1$(ord(j)=5);

c2(i,j,k)$(SPAN(i,j) and (ord(i) < ord (j)))..
            x(i,j,k) =L= y(i,j);

c3(i,j,k)$(SPAN(i,j) and (ord(i) > ord (j)))..
            x(i,j,k) =L= y(j,i);

c4..        sum((i,j), y(i,j)) =E= card(Node) - 1;


  Model NetworkDesign /all/ ;

  NetworkDesign.holdfixed=1;

  Solve NetworkDesign using MIP minimizing cost;

  DISPLAY SPAN, D, DH, zH, cost.l, x.l, y.l;
```

C.10 Star Model(Multiplexing and Continuous Flow Decision)

```
* Network Design Model.gms
* Feb. 22, 2006
* 14th Program: Included Network Data
*
* Fixed Cost, Cable Prices And Cable Capacities And Etc. From SNDLib Instances
* A Star Network Model; MUXs At Some Nodes; Multiplexing
* Hub Node Is Given
* No Protection
*
* x Is Between [0, 1]
* Non-used z And y Is Removed
*
$eolcom //
option iterlim=999999999; // avoid limit on iterations
option reslim=270000;     // timelimit for solver in sec.
option optcr=0.0;         // gap tolerance
option solprint=OFF;      // include solution print in .lst file
option limrow=0;          // limit number of rows in .lst file
option limcol=0;          // limit number of columns in .lst file
//-----------------------------------------------------------------------

  Sets
    Node                name of cities

    ALIAS(Node,i)
    ALIAS(Node,j)
    ALIAS(Node,k)
    ALIAS(Node,l);

  Set
    Cbl                 types of cables with different capacities;

  Parameters
    XX(Node)            x-bar of a city
    YY(Node)            y-bar of a city

    Cbl_Cpy(Cbl)        cable capacities
    Cbl_Prc(i,j,Cbl)    cable prices on link #ij#
```

```
    SPAN(i,j)            whether the link between #ij# exsits
    Pre_Cap(i,j)         pre-installed capacity already exsits on #ij#
    Pre_Cst(i,j)         how much have to pay on the pre-installed capacity
    Unt_Rt_Cst(i,j)      unit routing cost on #ij#
    Stp_Cst(i,j)         if not set up, how much should pay to set up

    D(k,l)               how much is the demand of flow between #kl#
    Rt_Unt(k,l)          the size of a routing unit on demand #kl#
    Dmd_Amt(k,l)         how much is the demand of unit between #kl#
    TOTAL_DEMAND         sum of demands of the entire network

    Hub(i)               which node is the hub
    DH(k)                the demand between node #i# and the hub;

$INCLUDE "deutsch.dat";

  // build a symmetric network
  SPAN(i,j)$(ord(i)<ord(j)) = SPAN(i,j) + SPAN(j,i);
  SPAN(i,j)$(ord(i)>ord(j)) = SPAN(j,i);

  Pre_Cap(i,j)$(ord(i)<ord(j)) = Pre_Cap(i,j) + Pre_Cap(j,i);
  Pre_Cap(i,j)$(ord(i)>ord(j)) = Pre_Cap(j,i);

  Pre_Cst(i,j)$(ord(i)<ord(j)) = Pre_Cst(i,j) + Pre_Cst(j,i);
  Pre_Cst(i,j)$(ord(i)>ord(j)) = Pre_Cst(j,i);

  Unt_Rt_Cst(i,j)$(ord(i)<ord(j)) = Unt_Rt_Cst(i,j) + Unt_Rt_Cst(j,i);
  Unt_Rt_Cst(i,j)$(ord(i)>ord(j)) = Unt_Rt_Cst(j,i);

  Stp_Cst(i,j)$(ord(i)<ord(j)) = Stp_Cst(i,j) + Stp_Cst(j,i);
  Stp_Cst(i,j)$(ord(i)>ord(j)) = Stp_Cst(j,i);

  Cbl_Prc(i,j,Cbl)$(ord(i)<ord(j)) = Cbl_Prc(i,j,Cbl) + Cbl_Prc(j,i,Cbl);
  Cbl_Prc(i,j,Cbl)$(ord(i)>ord(j)) = Cbl_Prc(j,i,Cbl);

  // transfer data to demand matrix D, and make it upper triangular
  // so here the demands are considered as undirected
  D(k,l) = Dmd_Amt(k,l)*Rt_Unt(k,l)$(Rt_Unt(k,l)>0) + Dmd_Amt(k,l)$(Rt_Unt(k,l)=0);
  TOTAL_DEMAND = sum((k,l), D(k,l));

  D(k,l)$(ord(k)<ord(l)) = D(k,l) + D(l,k);
  D(k,l)$(ord(k)>=ord(l)) = 0;
```

```
  // preparation for a star
  Hub(i) = 0;
  Hub('E') = 1;
  DH(k) = sum(l, D(k,l)) + sum(l, D(l,k));
  DH('E') = 0;

  SCALAR MUX_C      cost of a MUX;
  MUX_C = 100;

  BINARY Variables
       m(i)         whether to equip a MUX at node #i#
       y(i,j)       whether to build the arc #ij#;
  // set all y to be #0#
       y.fx(i,j) = 0;
  // set all y on half of the exsited links possible to be #1#
       y.up(i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = 1;

  INTEGER Variables
       z(Cbl,i,j) whether to lease carrier #Cbl# in arc #ij#, and how many;
  // set all z to be #0#
       z.fx(Cbl,i,j) = 0;
  // set all z on half of the exsited links possible to be non-zero integer
       z.up(Cbl,i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = ceil (TOTAL_DEMAND/Cbl_Cpy(Cbl));

  Variables
       cost         total cost
       x(i,j,k)     whether a flow for DH #k# passes through arc #ij#;

  // set all x to be #0#
       x.fx(i,j,k) = 0;
  // set all x on exsited links to have upper bound as #1#
       x.up(i,j,k)$(SPAN(i,j) and (DH(k) <> 0)) = 1;

  Equations
       obj               define objective function
       c1(j,k)           demand constraints
       c2(i,j)           capacity constraints
       c3(i,j,k)
       c4(i,j,k)
       c5(i)
       c6                tree constraint;

obj..       cost =E= sum((Cbl,i,j), z(Cbl,i,j)*Cbl_Prc(i,j,Cbl))
```

```
                  + sum((i,j), Stp_Cst(i,j)*y(i,j))
                  + (sum(i, m(i)) - 1)*MUX_C;

c1(j,k)$(DH(k) <> 0)..
            sum(i, x(i,j,k)) - sum(i, x(j,i,k))
                    =E= -1$(ord(j)=ord(k)) + 0 + 1$(ord(j)=5);

c2(i,j)$(SPAN(i,j) and (ord(i) < ord(j)))..
            sum(k, x(i,j,k)*DH(k)) + sum(k, x(j,i,k)*DH(k))
                    =L= sum(Cbl, z(Cbl,i,j)*Cbl_Cpy(Cbl));

c3(i,j,k)$(SPAN(i,j) and (ord(i) < ord (j)))..
            x(i,j,k) =L= y(i,j);

c4(i,j,k)$(SPAN(i,j) and (ord(i) > ord (j)))..
            x(i,j,k) =L= y(j,i);

c5(i)..     sum(j, y(i,j)) + sum(j, y(j,i)) =L= m(i)*card(Node) + 1;

c6..        sum((i,j), y(i,j)) =E= card(Node) - 1;


  Model NetworkDesign /all/ ;

  NetworkDesign.holdfixed=1;

  Solve NetworkDesign using MIP minimizing cost;

  DISPLAY SPAN, D, DH, cost.l, x.l, y.l, m.l, z.l;
```

C.11 Star Model(Multiplexing and Binary Flow Decision)

```
* Network Design Model.gms
* Feb. 22, 2006
* 14th Program: Included Network Data
*
* Fixed Cost, Cable Prices And Cable Capacities And Etc. From SNDLib Instances
* A Star Network Model; MUXs At Some Nodes; Multiplexing
* Hub Node Is Given
* No Protection
```

```
*
* x Is Between [0, 1]
* Non-used z And y Is Removed
*
$eolcom //
option iterlim=999999999; // avoid limit on iterations
option reslim=270000;     // timelimit for solver in sec.
option optcr=0.0;         // gap tolerance
option solprint=OFF;      // include solution print in .lst file
option limrow=0;          // limit number of rows in .lst file
option limcol=0;          // limit number of columns in .lst file
//-------------------------------------------------------------------------

  Sets
    Node                 name of cities

    ALIAS(Node,i)
    ALIAS(Node,j)
    ALIAS(Node,k)
    ALIAS(Node,l);

  Set
    Cbl                  types of cables with different capacities;

  Parameters
    XX(Node)             x-bar of a city
    YY(Node)             y-bar of a city

    Cbl_Cpy(Cbl)         cable capacities
    Cbl_Prc(i,j,Cbl)     cable prices on link #ij#

    SPAN(i,j)            whether the link between #ij# exsits
    Pre_Cap(i,j)         pre-installed capacity already exsits on #ij#
    Pre_Cst(i,j)         how much have to pay on the pre-installed capacity
    Unt_Rt_Cst(i,j)      unit routing cost on #ij#
    Stp_Cst(i,j)         if not set up, how much should pay to set up

    D(k,l)               how much is the demand of flow between #kl#
    Rt_Unt(k,l)          the size of a routing unit on demand #kl#
    Dmd_Amt(k,l)         how much is the demand of unit between #kl#
    TOTAL_DEMAND         sum of demands of the entire network

    Hub(i)               which node is the hub
```

```
    DH(k)                the demand between node #i# and the hub;

$INCLUDE "deutsch.dat";

  // build a symmetric network
  SPAN(i,j)$(ord(i)<ord(j)) = SPAN(i,j) + SPAN(j,i);
  SPAN(i,j)$(ord(i)>ord(j)) = SPAN(j,i);

  Pre_Cap(i,j)$(ord(i)<ord(j)) = Pre_Cap(i,j) + Pre_Cap(j,i);
  Pre_Cap(i,j)$(ord(i)>ord(j)) = Pre_Cap(j,i);

  Pre_Cst(i,j)$(ord(i)<ord(j)) = Pre_Cst(i,j) + Pre_Cst(j,i);
  Pre_Cst(i,j)$(ord(i)>ord(j)) = Pre_Cst(j,i);

  Unt_Rt_Cst(i,j)$(ord(i)<ord(j)) = Unt_Rt_Cst(i,j) + Unt_Rt_Cst(j,i);
  Unt_Rt_Cst(i,j)$(ord(i)>ord(j)) = Unt_Rt_Cst(j,i);

  Stp_Cst(i,j)$(ord(i)<ord(j)) = Stp_Cst(i,j) + Stp_Cst(j,i);
  Stp_Cst(i,j)$(ord(i)>ord(j)) = Stp_Cst(j,i);

  Cbl_Prc(i,j,Cbl)$(ord(i)<ord(j)) = Cbl_Prc(i,j,Cbl) + Cbl_Prc(j,i,Cbl);
  Cbl_Prc(i,j,Cbl)$(ord(i)>ord(j)) = Cbl_Prc(j,i,Cbl);

  // transfer data to demand matrix D, and make it upper triangular
  // so here the demands are considered as undirected
  D(k,l) = Dmd_Amt(k,l)*Rt_Unt(k,l)$(Rt_Unt(k,l)>0) + Dmd_Amt(k,l)$(Rt_Unt(k,l)=0);
  TOTAL_DEMAND = sum((k,l), D(k,l));

  D(k,l)$(ord(k)<ord(l)) = D(k,l) + D(l,k);
  D(k,l)$(ord(k)>=ord(l)) = 0;

  // preparation for a star
  Hub(i) = 0;
  Hub('E') = 1;
  DH(k) = sum(l, D(k,l)) + sum(l, D(l,k));
  DH('E') = 0;

  SCALAR MUX_C      cost of a MUX;
  MUX_C = 100;

  BINARY Variables
       m(i)         whether to equip a MUX at node #i#
       y(i,j)       whether to build the arc #ij#;
```

```
  // set all y to be #0#
       y.fx(i,j) = 0;
  // set all y on half of the exsited links possible to be #1#
       y.up(i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = 1;

  INTEGER Variables
       z(Cbl,i,j) whether to lease carrier #Cbl# in arc #ij#, and how many;
  // set all z to be #0#
       z.fx(Cbl,i,j) = 0;
  // set all z on half of the exsited links possible to be non-zero integer
       z.up(Cbl,i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = ceil (TOTAL_DEMAND/Cbl_Cpy(Cbl));

  Variables
       cost         total cost;

  BINARY Variable
       x(i,j,k)     whether a flow for DH #k# passes through arc #ij#;

  // set all x to be #0#
       x.fx(i,j,k) = 0;
  // set all x on exsited links to have upper bound as #1#
       x.up(i,j,k)$(SPAN(i,j) and (DH(k) <> 0)) = 1;

  Equations
       obj               define objective function
       c1(j,k)           demand constraints
       c2(i,j)           capacity constraints
       c3(i,j,k)
       c4(i,j,k)
       c5(i)             ;

obj..       cost =E= sum((Cbl,i,j), z(Cbl,i,j)*Cbl_Prc(i,j,Cbl))
                   + sum((i,j), Stp_Cst(i,j)*y(i,j))
                   + (sum(i, m(i)) - 1)*MUX_C;

c1(j,k)$(DH(k) <> 0)..
            sum(i, x(i,j,k)) - sum(i, x(j,i,k))
                     =E= -1$(ord(j)=ord(k)) + 0 + 1$(ord(j)=5);

c2(i,j)$(SPAN(i,j) and (ord(i) < ord(j)))..
            sum(k, x(i,j,k)*DH(k)) + sum(k, x(j,i,k)*DH(k))
                     =L= sum(Cbl, z(Cbl,i,j)*Cbl_Cpy(Cbl));
```

```
c3(i,j,k)$(SPAN(i,j) and (ord(i) < ord (j)))..
            x(i,j,k) =L= y(i,j);

c4(i,j,k)$(SPAN(i,j) and (ord(i) > ord (j)))..
            x(i,j,k) =L= y(j,i);

c5(i)..     sum(j, y(i,j)) + sum(j, y(j,i)) =L= m(i)*card(Node) + 1;


  Model NetworkDesign /all/ ;

  NetworkDesign.holdfixed=1;

  Solve NetworkDesign using MIP minimizing cost;

  DISPLAY SPAN, D, DH, cost.l, x.l, y.l, m.l, z.l;
```

C.12 Bus Model

```
* Network Design Model.gms
* Feb. 22, 2006
* 14th Program: Included Network Data
*
* Fixed Cost, Cable Prices And Cable Capacities And Etc. From SNDLib Instances
* A Bus Network Model; No Equipment At Any Node; No Multiplex
* Hub Node Is Given
* Do Not Consider Demands
*
* x Is Between [0, 1]
* Non-used z And y Is Removed
*
$eolcom //
option iterlim=999999999;  // avoid limit on iterations
option reslim=270000;      // timelimit for solver in sec.
option optcr=0.0;          // gap tolerance
option solprint=OFF;       // include solution print in .lst file
option limrow=0;           // limit number of rows in .lst file
option limcol=0;           // limit number of columns in .lst file
//-------------------------------------------------------------------------

  Sets
    Node                name of cities
```

```
    ALIAS(Node,i)
    ALIAS(Node,j)
    ALIAS(Node,k)
    ALIAS(Node,l);

  Set
    Cbl                 types of cables with different capacities;

  Parameters
    XX(Node)            x-bar of a city
    YY(Node)            y-bar of a city

    Cbl_Cpy(Cbl)        cable capacities
    Cbl_Prc(i,j,Cbl)    cable prices on link #ij#

    SPAN(i,j)           whether the link between #ij# exsits
    Pre_Cap(i,j)        pre-installed capacity already exsits on #ij#
    Pre_Cst(i,j)        how much have to pay on the pre-installed capacity
    Unt_Rt_Cst(i,j)     unit routing cost on #ij#
    Stp_Cst(i,j)        if not set up, how much should pay to set up

    D(k,l)              how much is the demand of flow between #kl#
    Rt_Unt(k,l)         the size of a routing unit on demand #kl#
    Dmd_Amt(k,l)        how much is the demand of unit between #kl#
    TOTAL_DEMAND        sum of demands of the entire network

    Hub(i)              which node is the hub;

$INCLUDE "deutsch.dat";

  // choose a hub node
  Hub(i) = 0;
  Hub('E') = 1;

  // build a symmetric network
  SPAN(i,j)$(ord(i)<ord(j)) = SPAN(i,j) + SPAN(j,i);
  SPAN(i,j)$(ord(i)>ord(j)) = SPAN(j,i);

  Pre_Cap(i,j)$(ord(i)<ord(j)) = Pre_Cap(i,j) + Pre_Cap(j,i);
  Pre_Cap(i,j)$(ord(i)>ord(j)) = Pre_Cap(j,i);

  Pre_Cst(i,j)$(ord(i)<ord(j)) = Pre_Cst(i,j) + Pre_Cst(j,i);
```

```
  Pre_Cst(i,j)$(ord(i)>ord(j)) = Pre_Cst(j,i);

  Unt_Rt_Cst(i,j)$(ord(i)<ord(j)) = Unt_Rt_Cst(i,j) + Unt_Rt_Cst(j,i);
  Unt_Rt_Cst(i,j)$(ord(i)>ord(j)) = Unt_Rt_Cst(j,i);

  Stp_Cst(i,j)$(ord(i)<ord(j)) = Stp_Cst(i,j) + Stp_Cst(j,i);
  Stp_Cst(i,j)$(ord(i)>ord(j)) = Stp_Cst(j,i);

  Cbl_Prc(i,j,Cbl)$(ord(i)<ord(j)) = Cbl_Prc(i,j,Cbl) + Cbl_Prc(j,i,Cbl);
  Cbl_Prc(i,j,Cbl)$(ord(i)>ord(j)) = Cbl_Prc(j,i,Cbl);

  // transfer data to demand matrix D, and make it upper triangular
  // so here the demands are considered as undirected
  D(k,l) = Dmd_Amt(k,l)*Rt_Unt(k,l)$(Rt_Unt(k,l)>0) + Dmd_Amt(k,l)$(Rt_Unt(k,l)=0);
  TOTAL_DEMAND = sum((k,l), D(k,l));

  D(k,l)$(ord(k)<ord(l)) = D(k,l) + D(l,k);
  D(k,l)$(ord(k)>=ord(l)) = 0;

  BINARY Variables
       y(i,j)     whether to build the arc #ij#;
  // set all y to be #0#
       y.fx(i,j) = 0;
  // set all y on half of the exsited links possible to be #1#
       y.up(i,j)$(SPAN(i,j) and (ord(i) < ord(j))) = 1;

  Variables
       cost         total cost
       x(i,j,k,l)   how much percentage of flow through arc #ij# for demand #kl#;
  // set all x to be #0#
       x.fx(i,j,k,l) = 0;
  // set all x on exsited links to have upper bound as #1#
       x.up(i,j,'A',l)$(SPAN(i,j) and (ord(l) > 1)) = 1;

  Equations
       obj               define objective function
       c1(j,l)           demand constraints to eliminate sub ring
       c2(i,j,k,l)
       c3(i,j,k,l)
       c4                bus constraint
       c5(i)             bus constraints;

obj..       cost =E= sum((i,j), Stp_Cst(i,j)*y(i,j));
```

```
c1(j,l)$(ord(l) > 1)..
            sum(i, x(i,j,'A',l)) - sum(i, x(j,i,'A',l))
                     =E= -1$(ord(j)=1) + 0 + 1$(ord(j)=ord(l));

c2(i,j,k,l)$(SPAN(i,j) and (ord(i) < ord (j)))..
            x(i,j,k,l) =L= y(i,j);

c3(i,j,k,l)$(SPAN(i,j) and (ord(i) > ord (j)))..
            x(i,j,k,l) =L= y(j,i);

c4..        sum((i,j), y(i,j)) =E= card(Node) - 1;

c5(i)..     sum(j, y(i,j)) + sum(j, y(j,i)) =L= 2 - Hub(i);


  Model NetworkDesign /all/ ;

  NetworkDesign.holdfixed=1;

  Solve NetworkDesign using MIP minimizing cost;

  DISPLAY SPAN, Stp_Cst, cost.l, x.l, y.l;
```

Appendix D

C Program Code

A C program "Convert.c" is written to convert the digital maps into GAMS-readable files.

```
/* This file converts a digital map to a GAMS_READABLE map. */
/* For version 1.0 instances from SND*Lib */

#include <stdio.h>

char InFile[99999],     /* the entire input file */
     InFileName[20],    /* name of the input file */
     OutFileName[20],   /* name of the output file */
     Word[30],          /* current word or number */
     LastWord[30],      /* last word or number */
     RecordWord1[30],   /* 1st recorded word or number for link end */
     RecordWord2[30];   /* 2nd recorded word or number for link end */

int InFileSize,         /* number of characters in the input file */
    CurrInFilePlace,    /* current scanning place in the input file */
    RereadPlace,        /* record the place where is needed to be re-read */
    WordLength,         /* length of the current word */
    counter1,           /* count how many '(' there are */
    counter2;           /* count how many ')' there are */

int NodeNumber,         /* number of nodes */
    CableNumber;        /* number of cables */

FILE *InFilePtr,        /* pointer to input file */
     *OutFilePtr;       /* pointer to output file */
```

```
GetFileName(char* name)  /* get the input file name */
{
  int i;

  for (i=0; name[i]!='\0'; i++)
  {
    InFileName[i] = name[i];
    OutFileName[i] = name[i];
    if (i>20) break;
  }
  InFileName[i]='.';  OutFileName[i++]='.';
  InFileName[i]='s';  OutFileName[i++]='d';
  InFileName[i]='l';  OutFileName[i++]='a';
  InFileName[i]='n';  OutFileName[i++]='t';

  InFilePtr = fopen(InFileName, "r");
  OutFilePtr = fopen(OutFileName, "w");
  ReadInFile();
}

ReadInFile()  /* read the input file into the array InFile */
{
  char C;

  InFileSize = 0;
  while(fscanf(InFilePtr, "%c", &C) != -1) InFile[InFileSize++] = C;
}

ReadBlanksAndEOLs()  /* starting at the current position in the input file, keep
                        scanning until get a "real" character */
{
  char C;

  while (1)
  {
    C = InFile[CurrInFilePlace];
    if (C == ' ' || C == '\n') CurrInFilePlace++;
    else break;
  }
}

int GetWord()  /* scans one word until hit its end; copy the word to the array
                  Word; also, return 1 if successful in getting a word, 0 otherwise
```

```
                (failed when reaching the end of file); copy to the last word */
{
  char C;

  RecordWord(0);  /* record the last word */

  WordLength = 0;
  while (1)
  {
    /* if reach the end of the file, leave, reporting failure */
    if (CurrInFilePlace == InFileSize) return 0;
    C = InFile[CurrInFilePlace];
    /* if hit the end of the word, leave, otherwise record the
       current character in the array Word */
    if (C == ' ' || C == '\n')
    {
      Word[WordLength] = '\0';  break;
    }
    else
    {
      if (C != ',') Word[WordLength++] = C;
      CurrInFilePlace++;
      if (C == '(' || InFile[CurrInFilePlace] == ')')
      {
        Word[WordLength] = '\0'; break;
      }
    }
  }

  return 1;
}

RecordWord(int a)  /* to record a word for further use */
{
  int i;

  switch (a)
  {
    case 0: for (i=0; i<30 && Word[i] != '\0'; i++)
            LastWord[i] = Word[i];  LastWord[i] = '\0';  break;
    case 1: for (i=0; i<30 && Word[i] != '\0'; i++)
            RecordWord1[i] = Word[i];  RecordWord1[i] = '\0';  break;
    case 2: for (i=0; i<30 && Word[i] != '\0'; i++)
```

```
            RecordWord2[i] = Word[i];  RecordWord2[i] = '\0';  break;
  }
}

CountAndCheckEmpty()  /* check whether there is a situation like "()" */
{
  if (Word[0] == '(') counter1++;
  if (Word[0] == ')' && LastWord[0] != '(') counter2++;
  if (Word[0] == ')' && LastWord[0] == '(') counter1--;
}

WriteWord(char* W)  /* write a word into the file */
{
  fprintf(OutFilePtr, "%s", W);
}

NewCount()  /* start a new count of '(' and ')' */
{
  counter1 = 0;  counter2 = 0;
}

SetNodes()  /* set the SET Node */
{
  fprintf(OutFilePtr, "SET \n");
  fprintf(OutFilePtr, "    Node    / ");
  /* record the place for reread for XX && YY */
  RereadPlace = CurrInFilePlace;
  NewCount();

  while(1)  /* begin to scan which word is a node */
  {
    ReadBlanksAndEOLs();

    if (GetWord())
    {
      CountAndCheckEmpty();

      if (Word[0] != '(' && Word[0] != ')' && (counter1-counter2) == 1)
      {
        if (counter1>1) fprintf(OutFilePtr, ", ");
        NodeNumber++;
        WriteWord(Word);
      }
```

```
      if ((counter1-counter2) == 0 && counter1 > 0)
      {
        fprintf(OutFilePtr, " /; \n\n");
        CurrInFilePlace = RereadPlace;
        break;
      }
    }
  }
}

SetXXYY()  /* set node X and Y position */
{
  int i, XXorYY = 0;
  NewCount();

  while(1)
  {
    ReadBlanksAndEOLs();

    if (GetWord())
    {
      CountAndCheckEmpty();

      if (Word[0] != '(' && Word[0] != ')' && (counter1-counter2) == 1)
        RecordWord(1);
      if (Word[0] != '(' && Word[0] != ')' && (counter1-counter2) == 2)
      {
        if (XXorYY == 0) fprintf(OutFilePtr, "XX('");
        else fprintf(OutFilePtr, "YY('");
          XXorYY = XXorYY ^ 1;
        WriteWord(RecordWord1);
        fprintf(OutFilePtr, "')=");
        WriteWord(Word);
        fprintf(OutFilePtr, ";\n");
      }
    }

    if ((counter1-counter2) == 0 && counter1 > 0) break;
  }
}

SetCableCapacitys()  /* set the SET Cbl and the Parameter Cbl_Cpy(Cbl) */
```

```
{
  int i, counter3 = 0, j = 0;  /* count how many kinds of calbes there are */
  /* record the place for reread for Link Parameters */
  RereadPlace = CurrInFilePlace;
  NewCount();

  while(1)  /* write the SET Cbl */
  {
    ReadBlanksAndEOLs();

    if (GetWord())
    {
      CountAndCheckEmpty();

      if (counter1 == 3 && counter2 == 1) counter3++;
      if (counter3 == 2 && Word[0] == 'U' && Word[1] == 'N') break;
      if (counter3 == 2 && Word[0] != 'U' && counter1 < 4)
      {
        fprintf(OutFilePtr, "\nSET \n");
        fprintf(OutFilePtr, "    Cbl     / ");
      }
      if (counter1 == 3 && Word[0] == ')')
      {
    counter3 = (counter3-1)*0.5;
        CableNumber = counter3;
        fprintf(OutFilePtr, "Cbl1*Cbl%d /;\n\n", counter3);
        counter3 = 0;
      }
      if (counter1 == 5 && counter2 == 3)
      {
        counter3++;  j = j ^ 1;  i = (counter3+1)*0.5;
        if (j == 0)  /* write the Parameter Cbl_Cpy(Cbl*)*/
    {
          fprintf(OutFilePtr, "Cbl_Cpy('Cbl%d')=", i);
          WriteWord(Word);
          fprintf(OutFilePtr, ";\n");
        }
      }
      if (counter2 > 3) break;
    }
  }
}
```

```
SetLinks()
{
  int counter3 = 0;   /* mark1 the position in a link line */
  int k = 0;          /* mark2 which '(' 1st or 2nd, in this line */
  int i, j = 0;
  fprintf(OutFilePtr, "\nSPAN(i,j)=0;\n");
  CurrInFilePlace = RereadPlace;
  NewCount();

  while(1)
  {
    ReadBlanksAndEOLs();

    if (GetWord())
    {
      CountAndCheckEmpty();

      counter3++;
      if (Word[0] == '(')
      {
        counter3 = 0;
        k = k ^ 1;   /* k=0 when in the 1st () and k=1 when in the 2nd () */
      }

      /* record the two ends of this link */
      if (k == 0 && (counter1 - counter2) == 2) RecordWord(counter3);

      /* set Parameters SPAN, Pre_Cap, Pre_Cst, Unt_Rt_Cst, Stp_Cst of a link */
      if (k == 0 && counter1 > 1 && (counter1 - counter2) == 1)
      {
        switch (counter3)
      {
          case 3: fprintf(OutFilePtr, "\nSPAN('%s','%s')=1;\n", RecordWord1, RecordWord2);
                  break;
          case 4: fprintf(OutFilePtr, "Pre_Cap('%s','%s')=", RecordWord1, RecordWord2);
                  break;
          case 5: fprintf(OutFilePtr, "Pre_Cst('%s','%s')=", RecordWord1, RecordWord2);
                  break;
          case 6: fprintf(OutFilePtr, "Unt_Rt_Cst('%s','%s')=", RecordWord1, RecordWord2);
                  break;
          case 7: fprintf(OutFilePtr, "Stp_Cst('%s','%s')=", RecordWord1, RecordWord2);
                  break;
        }
```

```
        if (Word[0] != ')')
        {
          WriteWord(Word);
          fprintf(OutFilePtr, ";\n");
        }
      }

      /* set Parameter Cbl_Prc of a link */
      if (k == 1 && counter1 > 1 && Word[0] != '(' && (counter1 - counter2) == 2)
      {
        j = j ^ 1;  i = counter3*0.5;
        if (j == 0)
    {
          fprintf(OutFilePtr, "Cbl_Prc('%s','%s','Cbl%d')=", RecordWord1, RecordWord2, i);
          if (Word[0] != ')')
          {
            WriteWord(Word);
            fprintf(OutFilePtr, ";\n");
          }
        }
      }

      if ((counter1 - counter2) == 0 && counter1 > 1) break;
    }
  }
}

SetDemands()
{
  int counter3 = 0;   /* mark the position in a link line */
  int i, j = 0;
  NewCount();

  while(1)
  {
    ReadBlanksAndEOLs();

    if (GetWord())
    {
      CountAndCheckEmpty();

      counter3++;
      if (Word[0] == '(') counter3 = 0;
```

```
      /* record the two ends of this link */
      if ((counter1 - counter2) == 2) RecordWord(counter3);

      /* set Parameters Rt_Unt, Dmd_Amt, Max_PLgth of a demand */
      if (counter1 > 1 && (counter1 - counter2) == 1 && counter3 == 4)
        fprintf(OutFilePtr, "\nRt_Unt('%s','%s')=", RecordWord1, RecordWord2);
      if (counter1 > 1 && (counter1 - counter2) == 1 && counter3 == 5)
        fprintf(OutFilePtr, "Dmd_Amt('%s','%s')=", RecordWord1, RecordWord2);
      if (counter1 > 1 && (counter1 - counter2) == 1 && counter3 == 6 && Word[0] != 'U')
        fprintf(OutFilePtr, "Max_PLgth('%s','%s')=", RecordWord1, RecordWord2);
      if (counter1 > 1 && (counter1 - counter2) == 1 && counter3 != 3 && counter3 < 6
                      || counter3 == 6 && Word[0] != 'U')
      {
          WriteWord(Word);
          fprintf(OutFilePtr, ";\n");
      }

      if ((counter1 - counter2) == 0 && counter1 > 1) break;
    }
  }
}

main(int argc, char** argv)
{
  GetFileName(argv[1]);
  CurrInFilePlace = 0;
  NodeNumber = 0;
  CableNumber = 0;

  /* keep alternating this cycle: scan through blanks and end-of-line
     characters, copying them to the output file */
  while(CurrInFilePlace < InFileSize)
  {
    ReadBlanksAndEOLs();

    if (GetWord())
    {
      if (Word[0] == 'N' && Word[1] == 'O' && Word[2] == 'D'
                         && Word[3] == 'E' && Word[4] == 'S')
      {
        SetNodes();
        SetXXYY();
```

```
      }

      if (Word[0] == 'L' && Word[1] == 'I' && Word[2] == 'N'
                         && Word[3] == 'K' && Word[4] == 'S')
      {
        SetCableCapacitys();
    SetLinks();
      }

      if (Word[0] == 'D' && Word[1] == 'E' && Word[2] == 'M' && Word[3] == 'A'
                         && Word[4] == 'N' && Word[5] == 'D' && Word[6] == 'S')
        SetDemands();
    }
  }

  fprintf(OutFilePtr, "\nThe End!");
  printf("\nNetwork %s is converted to a GAMS readable file.\n\n", argv[1]);
}
```