NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

OPTIMAL CONFIGURATION OF DIGITAL COM-
MUNICATION NETWORK

by

Hwang, Yong Goo

December 1990

Thesis Advisor   Myung W. Suh

Approved for public release; distribution is unlimited.

91-17022

91 12 4 018

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified		1b Restrictive Markings			
2a Security Classification Authority		3 Distribution Availability of Report			
2b Declassification Downgrading Schedule		Approved for public release: distribution is unlimited.			
4 Performing Organization Report Number(s)		5 Monitoring Organization Report Number(s)			
6a Name of Performing Organization Naval Postgraduate School	6b Office Symbol (if applicable) CS	7a Name of Monitoring Organization Naval Postgraduate School			
6c Address (city, state, and ZIP code) Monterey. CA 93943-5000		7b Address (city, state, and ZIP code) Monterey, CA 93943-5000			
8a Name of Funding/Sponsoring Organization	8b Office Symbol (if applicable)	9 Procurement Instrument Identification Number			
8c Address (city, state, and ZIP code)		10 Source of Funding Numbers			
		Program Element No	Project No	Task No	Work Unit Accession No

11 Title (include security classification)
OPTIMAL CONFIGURATION OF DIGITAL COMMUNICATION NETWORK

12 Personal Author(s) Hwang, Yong Goo

13a Type of Report Master's Thesis	13b Time Covered From To	14 Date of Report (year, month, day) December 1990	15 Page Count 76

16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms (continue on reverse if necessary and identify by block number)
Field	Group	Subgroup	Network, Lagrangian relaxation, Subgradient optimization

19 Abstract (continue on reverse if necessary and identify by block number)

As the costs for maintaining computer communication networks are rapidly rising, it is particularly important to design the network efficiently. The objective of this thesis is to model the minimum cost design of digital communication networks and propose a heuristical solution approach to the formulated model.

The minimum cost design has been modeled as a zero-one integer programming problem. The Lagrangian relaxation method and subgradient optimization procedure have been used to find reasonably good feasible solutions.

Although the reliability requirement for computer communication networks is as important as the cost factor, only the cost factor is considered in the context of this thesis.

20 Distribution Availability of Abstract ☒ unclassified unlimited ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified	
22a Name of Responsible Individual Myung W. Suh	22b Telephone (include Area code) (408) 646-2637	22c Office Symbol ASSU

DD FORM 1473,84 MAR   83 APR edition may be used until exhausted   security classification of this page
All other editions are obsolete

Unclassified

i

Optimal Configuration of Digital Communication Network

by

Hwang, Yong Goo
Major, Republic of Korean Army
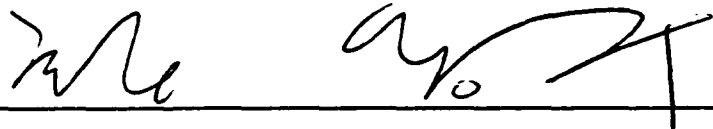B.S., Dong Kook university, 1987

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF COMPUTER SCIENCE

from the

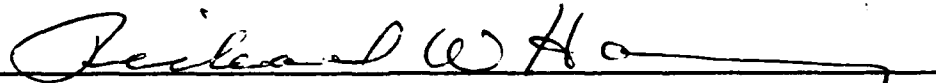NAVAL POSTGRADUATE SCHOOL
December 1990

Author: _____

Hwang, Yong Goo

Approved by: _____

Myung W. Suh, Thesis Advisor

_____

Richard W. Hamming, Second Reader

_____

Robert B. Mcghee, Chairman,
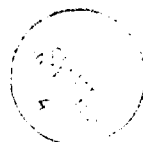Department of Computer Science

ii

ABSTRACT

As the costs for maintaining computer communication networks are rapidly rising, it is particularly important to design the network efficiently. The objective of this thesis is to model the minimum cost design of digital communication networks and propose a heuristical solution approach to the formulated model.

The minimum cost design has been modeled as a zero-one integer programming problem. The Lagrangian relaxation method and subgradient optimization procedure have been used to find reasonably good feasible solutions.

Although the reliability requirement for computer communication networks is as important as the cost factor, only the cost factor is considered in the context of this thesis.

iii

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I. INTRODUCTION

Computer communication networks are becomming a vital part of today's industrial, govermental, financial, and service organizations. The efficiency and quality of their internal operations and external services depends upon how effective their computer communication networks are. Design of computer communication networks requires the consideration of a number of factors such as performance constraints, reliability constraints, and costs. As the costs for maintaining computer communication networks are rapidly rising, it is particularly important to design the network efficiently; i.e., the emphasis on keeping the costs low in the design of networks is becoming evident.

Due to the inherent complexity of network design problems, the comprehensive optimal design of efficient networks is usually infeasible except for very simple networks. The objective of this thesis is to model the minimum cost design of digital communication networks and propose a heuristical solution approach to the formulated model.

It is assumed that node locations are fixed by the required placement of terminals and network facilities. Therefore network design is predominately a question of where to put links and what their data carrying capacities should be. For a given set of n nodes, there are n(n-1)/2 possible links. It is seldom necessary for them to be fully connected. Consequently, there is an inherent design tradeoff between equipment simplicity and low cost with fewer links. Although the reliability requirement for computer communication networks is as important as the cost factor, only the cost factor is considered in the context of this thesis.

The minimum cost design has been modeled as a zero-one integer programming problem. The Lagrangian relaxation method and subgradient optimization procedure have been used to find reasonably good feasible solutions.

The organization of this thesis is as follows: Chapter II is an overview of digital transmission facilities and describes T1, T3 circuits and digital cross-connect systems.

1

Chapter III formulates a model of the minimum cost design problem and introduces a Lagrangian relaxation method along with a subgradient optimization procedure. How to implement this technique in network design is also described. Chapter IV presents the results of computational tests. Finally, conclusion and recommendations for further research are offered in Chapter V.

II. DIGITAL TRANSMISSION FACILITIES

A. Overview of Digital Transmission

Digital transmission techniques were introduced in the Bell System in the early 1960s for efficient transport of voice signals. At that time, data for computer communications accounted for a small percentage of network traffic. Therefore, the basic digital transmission structure was centered around human voice communication.

Although a plot of amplitude versus time for a speech sample shows significant high-frequency components, a successful telephone conversation requires less than 4KHz of audio bandwidth. After being processed by a low-pass filter, a speech sample loses its high-frequency components and is ready to be digitized.

Since the maximum analog frequency to be reproduced is 4KHz, an 8Kbit/s sampling rate was adopted according to the Nyquist theorem. By sampling the voice signal every 125 microseconds, its essential information is extracted from the analog format and is ready for digital encoding. This pulse-amplitude modulated(PAM) signal contains all the information in the original signal up to approximately 4KHz.

The modulation scheme that was chosen for the early digital transmission standards is the easiest to implement. The pulse code modulation(PCM) is currently used worldwide for a standard voice communication. Figure 1 shows how the voice signal is encoded to digital signal. The pulse amplitude modulation(PAM) signal is quantized by mapping into discrete amplitude levels, each with a unique binary code. Naturally, additional discrete mappings reduce the quantization error and improve the fidelity of the coded signal. The analog voice signal is sampled 8000 times per second with each sample measuring the amplitude of the signal, and sampled amplitude is encoded into an 8-bit digital code. By creating 8-bit codes 8000 times per second, the rate of a digitized voice signal is 8 multiplied by 8000, or 64Kbit/s. This forms the basic 64Kbit/s channel that is the foundation of the digital network. Such a digital channel is often referred to as a DS-0(digital signal level 0).

3
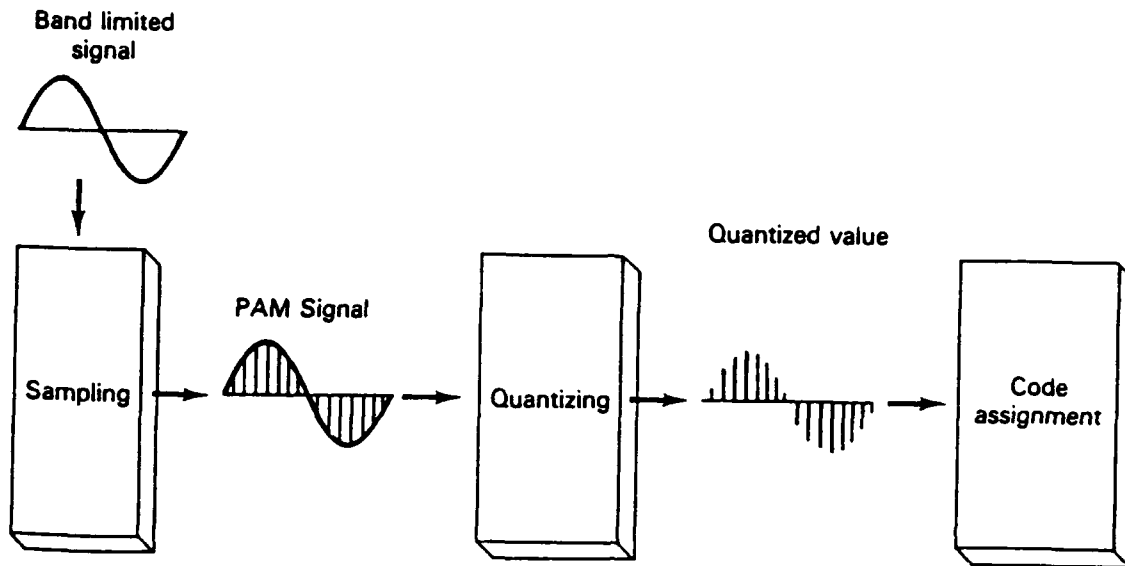
Figure 1: Pulse Code Modulation

The original deploymant of digital transmission was driven by a desire to conserve copper pairs outside telephone offices. Each analog voice signal in the existing telephone network consumed a physical pair of copper wires from a subscriber location to a telephone central office. Once the basic block was digital, it became feasible to multiplex the digital signals together into a higher-order digital signal. The time-division multiplexing led to defining the next level of the digital hierarchy as being equivalent to 24 DS-0 signals. This level is referred to as DS-1 or T1. [Ref.13]

The multiplexing function that created a DS-1 signal was originally handled by a network element known as a digital channel bank or D-bank. Numerous devices, of course, offer T1 interfaces at these days.

B. T1 System

1. What is T1?

The T1 system has become one of the most widely used high-capacity systems for transmission of voice and data. Originally conceived as voice transmission

4

technology in the early 1960s, it has evolved into a cost-effective and highly flexible means of transmitting both voice and data.

T1 is based on time-division multiplexing (TDM) 24 users onto one physical circuit. It usually involves a digital service unit(DSU) and a channel service unit(CSU) to accomplish the analog-to-digital conversion and the multiplexing. At the inception of T1, it was known that solid copper cable was capable of providing frequencies about 100KHz. Consequently, the earlier frequency division multiplexed(FDM) telephone system multiplexed 24 users onto one copper wire. Each user was assigned a different 4-KHz frequency band. The composite of 24-voice channels totalled 96KHz, which was within the capacity of twisted pair wires. [Ref.11]

The T1 system is designed around a 1.544Mbit/s rate, which was about the highest rate that could be supported across twisted pair for a distance of approximately one mile.

T1 has several advantages. The economic incentive to use T1 can be enormous despite the apparent high initial cost of equipment and the continuing costs for leased lines. An investment in a new network based on T1, however, can break even within one year. The end user organization operating a private T1 network enjoys the diagnostic power to identify and isolate faults and the ability to set up and change connections between points on the network at will. Another advantage is that T1 technology simplifies, and thereby improves corporate communications. Combining multiple voice and data channels over a single high-speed digital circuit eliminates many separate networks and many forms of network management. Therefore, T1 makes network control more practical and improves reliability. With the availability of large amounts of bandwidth, under immediate control, the network operator is able to offer the corporation new services that were not practical in the past. Additional circuits are readily available through reconfiguration. [Ref.11]

2. T1 Frame Format

The majority of T1 offerings digitize the voice signal through PCM or adaptive differential pulse code modulation(ADPCM). Whatever the encoding technique used,
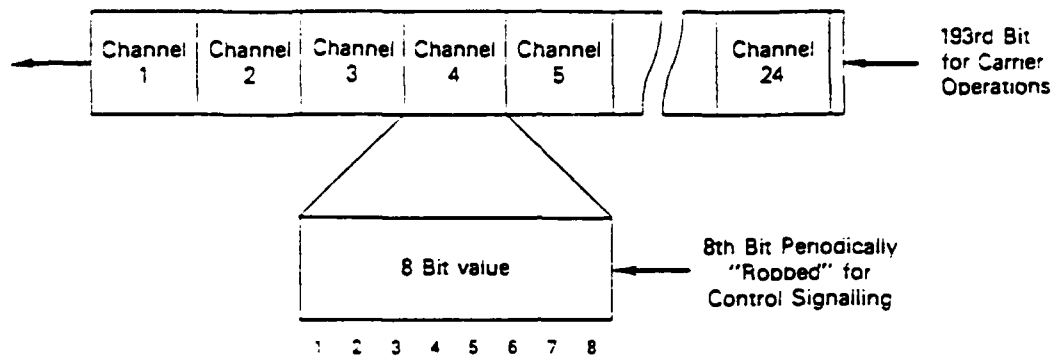
Figure 2: The T1 Based Frame

once the analog images are translated to digital bit streams, many T1 systems then
time-division multiplex voice and data together in 24 or 44 user slots within each
frame.

This section introduces the concept that the carriers transmit 24 voice or data
channels together with TDM frames. The T1 carrier system provides multiplexing
by sampling the 24 channels at a combined rate of 192,000 times per second. Figure
2 shows how the 24 channels are multiplexed into a frame. The frame contains one
sample from each channel, plus an additional bit for frame synchronization. Thus,
the complete frame contains 193 bits (8 bits per channel × 24 channels + 1 sync.
bit = 193bits). In addition, each of the 24 channels requires a 64Kbit/s rate (8000
samples/s× 8 bit/sample = 64Kbit/s).

The basic T1 frame consist of 24 eight-bit slots and a framing bit. The details
of T1 frame format are characterized by the following concepts:

- Alternate Mark Inversion (AMI)

- 1s density requirements

- Bit robbing

- Binary 7 zero substitution (B7ZS)

6

(a) BMI Bipolar Signal

(b) Odd Omission

(c) Even Omission

(d) Odd Commission

(e) Even Commission (1) or
Even Commission |
Omission (2)

Normal Pulse   Omitted Pulse   Committed Pulse
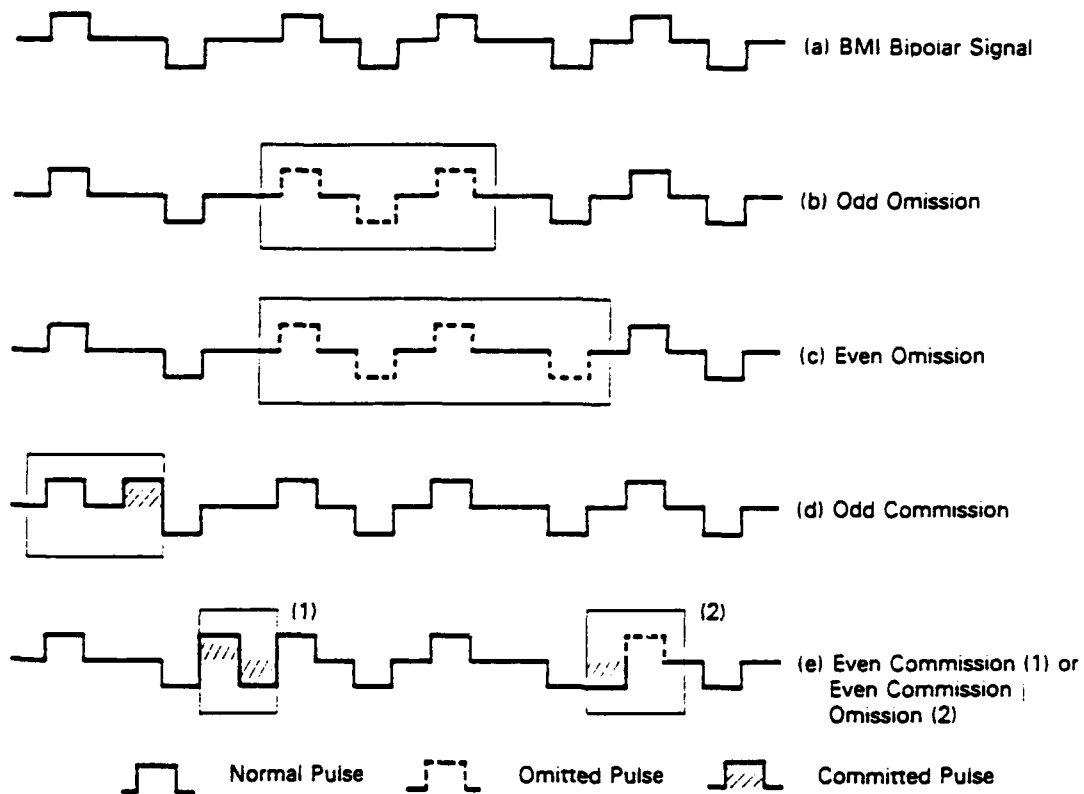
(b) Odd Omission: Bipolar Code Violation

(c) Even Omission: No Bipolar Code Violation

(d) Odd Commission: Bipolar Code Violation

(e) Even Commission | Omission

Figure 3: Format errors Due to Bipolar Code Violation

- Framing bit

Alternate Mark Inversion

The AMI concept is shown in Figure 3(a). The user may recognize this code as bipolar coding. The T1 system requires that a 1 pulse must be sent as an opposite polarity from the preceding 1 pulse, regardless of the number of 0s in between the two 1s. This bipolar code performs well because it has no direct current component and therefore can be coupled with transformers. The scheme also makes efficient use of bandwidth.

1s Density Requirements

T1 provides no separate clocking signal. The timing or clocking information is embedded in the data stream. At the receiving end, the clock is recovered from the data stream by the detection of 1 pulse. If the T1 data stream has insufficient 1 pulses embedded, the receiver can no longer produce reliable timing out.

To overcome this problem, a certain number of 1s must be present to ensure proper timing. This concept is called 1s density. The T1 facilities require that no more than 15 consecutive 0s shall be present in the frame. An additional requirement is that there must be at least three 1s in every 24 bits.

Due to noise, mechanical failures, etc., the bits may become distorted, which can cause a *violation* of the AMI rule. Bipolar violations or excessive errors are known as format errors because the errors are not in conformance with the required T1 format. A bit distortion may not cause a format error. As Figure 3(b) through 3(e) show, the nature of the error determines if the bipolar violation is detected. The bipolar signal is altered with errors of omission or errors of commission.

Bit Robbing

The least significant bit of every time slot in the 'signalling frames' is overwritten to encode the control signals. This concept is called bit robbing.

For the transmission of data, the 8th bit is unavailable. Consequently, the majority of T1 and related systems use a 56Kbit/s transmission rate instead of the 64Kbit/s rate actually available.

Binary 7 Zero Substitution

To establish 1s density in the T1 signal, a technique called B7 zero code suppression is used. As previously discussed, a 1 is substituted in a T1 frame to prevent an occurrence of more than 15 consecutive 0s. It is possible that this substitution can occur with any bit in the frame. A voice channel can tolerate the loss of the 7th bit to B7 substitution. However, a data channel cannot operate with this arrangement.

It is not necessary to rob bits in a data channel since there is no telephone signal. However, it is quite possible to have all 0s in a data channel. Therefore, the data could be corrupted by the B7 zero code suppression technique. The insertion does

T-1 Facility Termination
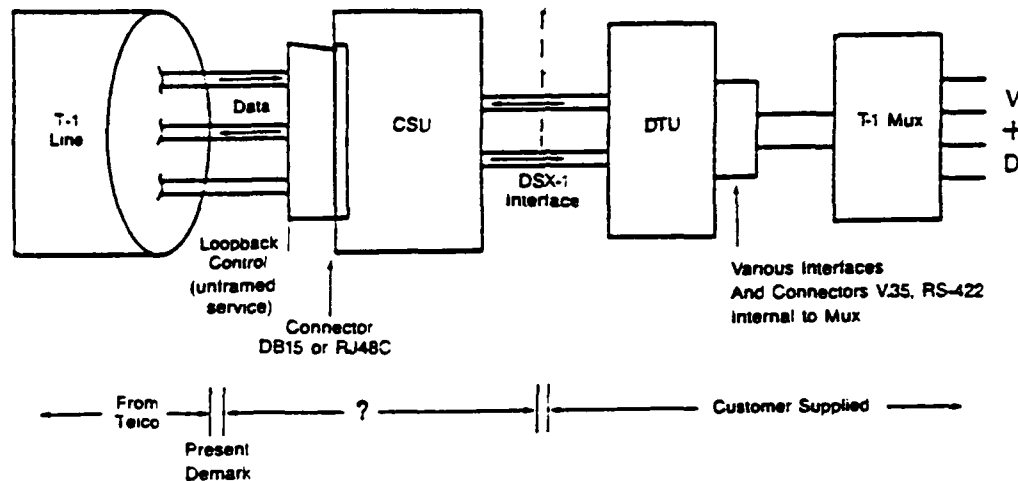
Figure 4: T1 Facility Termination

not affect the voice channel, but it certainly affects a data channel. To prevent this problem, a data channel generally only uses 7 bits and any 1 substitution can be placed into the 8th bit. This prevents the B7 zero code suppression from corrupting the data channel.

Framing Bit

To decode the incoming data stream, a receiver must be able to associate each sample with the proper channel. At a minimum, the begining and ending of the frame must be recognized. The function of the framing bit provides this requirement. This bit is located in the 193rd bit of each frame. It is not part of user's information, but added by the system for framing. The use of framing bit varies significantly depending on the type of T1 and of the technology. [Ref.17]

3. T1 Networks

(A) T1 Interface

Figure 4 shows T1 facility termination. The T1 facility termination enters the users premises as two wire pairs for data. There may be additional wires for loopback control and testing of unframed lines. The channel service unit (CSU)

9
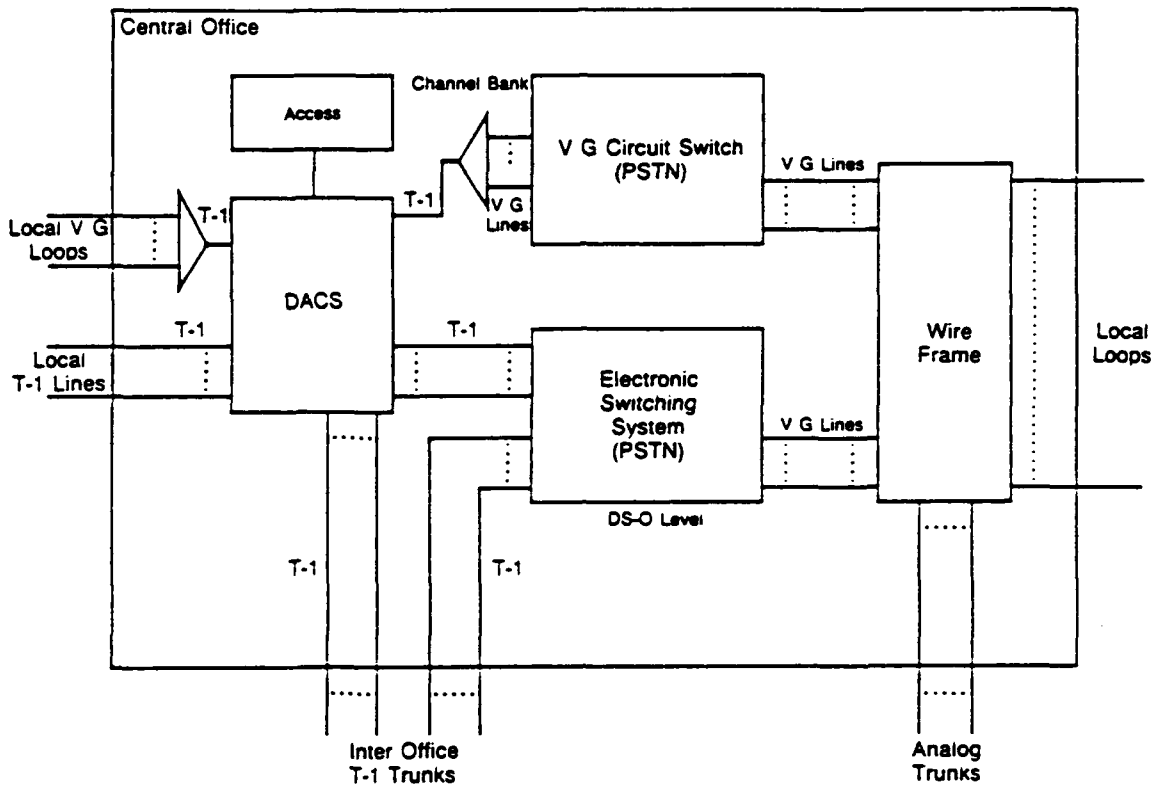
Figure 5: Digital Cross-Connect System

provides loopbacks to the network and is the last regeneration point for incoming signals. DSX-1 as an interface is well defined in terms of electrical and mechanical features. A digital service unit (DSU) matches various terminal interfaces to the DSX-1 standard and usually is part of the user's terminal equipment: multiplexer, bandwidth manager, PBX, computer, etc.[Ref.11]

(B) Digital Cross-Connect Systems

Digital cross-connect systems (DCS) are electronic switches able to take bit streams out of the T1-carrier and route them to a given output line. DCS allow remote access to and cross-connection of large bandwidth circuits. Figure 5 shows DCS. DCS can be divided into three primary subsystems.

At the core of the DCS is the Matrix Subsystem where the actual channel level (DS-0) cross-connections take place. DCS systems are controlled by the

Data Bypass
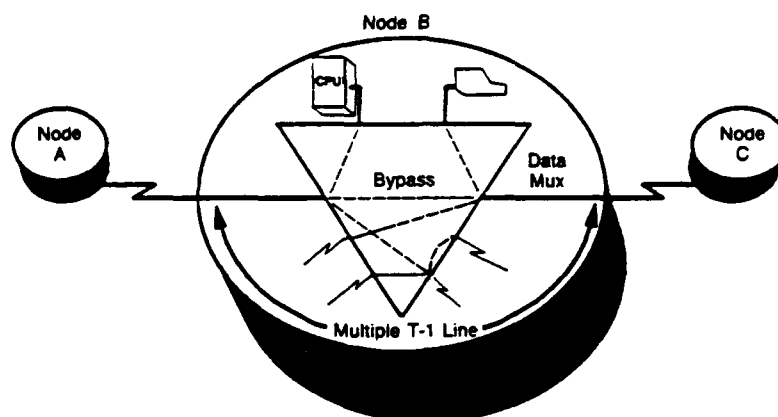
Figure 6: Data Bypass

Administrative Subsystem, where high-level processors manage the overall system functions. Other common elements such as system synchronization and user interface/control port access reside within this subsystem. Finally, there is the Interface Subsystem, which terminates DS-1 or higher facilities. Interface subsystems are organized into growth increments, or units, which accommodate anywhere from 28 to 32 slot positions each. Aside from facility terminations, these slot positions can be used to accommodate features such as sub-rate date multiplexing and bridging of digital services.

Figure 6 shows data bypass based on DCS. Data bypass resembles the drop and insert function of the voice channel bank. What distinguishes the two is that data bypass offers variable bandwidth allocation and additional flexibility in routing connection among many more T1 links.[Ref.11]

(C) T1 Networking

Lower cost for digital lines and for customer premises equipment have made all-digital networking not only possible, but attractive. Every location is on the digital *backbone*, through small sites may be served by 56Kbit/s or other data
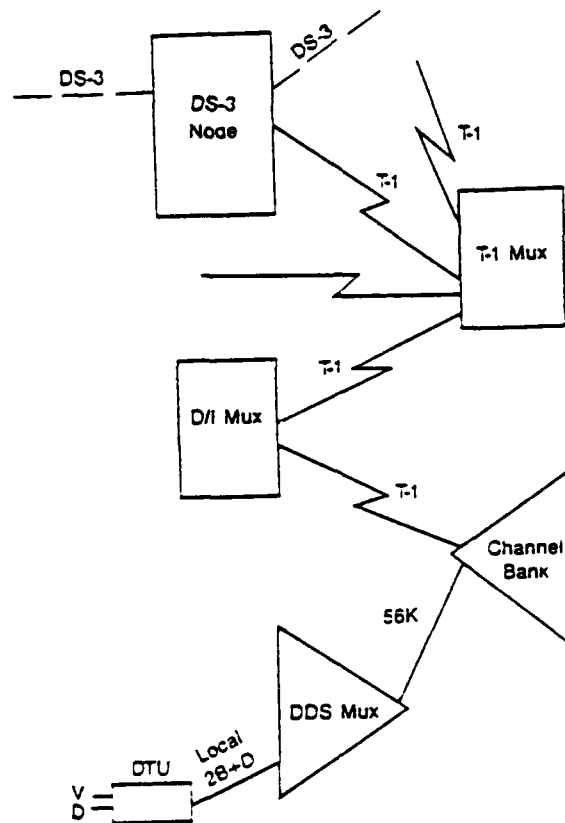
Figure 7: T1 Networking

link speed less than a full T1.

The T1 networking are shown in Figure 7. There are two main benefits in T1 networking. Trends in tariffed costs are up for analog facilities and down for digital facilities. As time goes on, it takes fewer analog lines to justify a digital line. Currently, as few as 3 or 4 leased voice grade lines cost as much as a 56kbit/s digital service. As few as two or three 56K lines cost as much as a T1. The result is that we can cost-justify digital service in more and more locations. Although there exist various types of nodes with a common network management system, everything can be brought under one network management system because T1 networking is all-digital. [Ref.11]

C. T3 SYSTEM

1. What is T3?

Many corporate managers are administering T1 networks that are growing rapidly with new users asking for more and more bandwidth. Increasingly, data communications managers are eyeing T3 network as a solution to the problems raised by this expansion.

T3 refers to a digital transmission facility running at DS-3 data rate of 44.736Mbit/s. It is equivalent of 28 T1s, and is more economical than a much less number of T1s. For a distance of less than 50 miles, only four T1s are required to break even.

T3 circuits usually use microwave and optical fiber as transmission media. Optical fiber has a much higher transmission capacity than digital microwave. Digital microwave has some problems. First, the electromagnetic spectrum is strictly licensed by the Federal Communications Commission. Second, strict spectrum allocations imply that digital microwave cannot be readily upgraded in bandwidth. While a 45Mbit/s optical fiber can be readily converted to 565Mbit/s, a 45Mbit/s digital microwave setup is probably destined to remain at 45Mbit/s. Finally, digital microwave devices are inherently limited to line-of-sight distance.

Many T3 networks mix optical fiber and digital microwave. The optical fiber can be used in high-density areas, with microwave hops to lower-density remote locations. Alternatively, the main network can be placed entirely on fiber, with T3 microwave used for emergency restoration in the event of a cable cut. Dual-media transmission is especially attractive to service-critical networks such as banks, airlines, and other on-line transaction-processing companies.

T3 was originally designed to be used as interexchange trunk within the public telephone network.[Ref.13]

2. T3 Format

By the end of the 1970s, a way around the limitations of the copper pair was on the horizon. Transmission technologies based on the nearly limitless band-

width of optical fiber were emerging from the laboratory, and a new layer of the digital hierarchy seemed appropriate. Based on technology available at the time, an asynchronous DS-3 rate was defined as the combination of seven DS-2 signals of 6.176Mbit/s. Framing and stuffing bits, as well as rudimentary error checking and internal communications, were added to the DS-2 tributaries to generate an aggregate bit rate of 44.736Mbit/s.

Notice that the information content, or payload, of a DS-3 is equal to 672 times 64Kbit/s = 43.008Mbit/s. The additional 1.728Mbit/s of overhead represents the sum of the DS-1 framing bit, the DS-2 framing and stuffing bits, and the variety of DS-3 overhead bits. In terms of network efficieny, the DS-3 format devotes 96 percent of the transmission bandwidth to payload, with approximately 4 percent overhead. [Ref.13]

Although the DS-3 was designed to be created from multiple DS-2 signals, there was an obvious inefficiency in using two separate devices for the DS-1-to-DS-2 and DS-2-to-DS-3 multiplexing functions.

There are few choices when it comes to T3 formats. The long established standard in the public network is now called M13. But for technical reasons there are new proposals for formats called Syntran and SONET.

(A) M13 Format

The name M13 comes from the fact that this multiplexer connects the DS-1 to the DS-3 level. One DS-3 contains 28 DS-1s. They are combined in two steps. The M13 process first joins four DS-1 lines into DS-2 bit stream at 6.312Mbit/s. This step is more than a straight time division multiplexer function. M12 bit-interleaves the four inputs, but it also adds 136Kbit/s of overhead and justification or bit stuffing.

The second step within M13 combines seven DS-2 streams into one DS-3. Like M12, the M23 process is bit-interleaved, with justification. That is, the DS-3 clock need not be synchronized with any DS-2 or DS-1 line speed. M13 overruns the DS-2 clock, and bit stuffs or adds extra bits to fill out the DS-3 rate of 44.736Mbit/s.

Whatever the original reasons for the two-step multiplexing, it creats a

14

problem for us today. There is no reliable way to tell where a given DS-0 is located within the DS-3 bit stream. This makes it very difficult to extract just one DS-0 because it will have no definite location within the DS-3 frame.

To get at an individual channel you must typically reverse the two step and demux the DS-3 down to individual DS-1s. The need for a DS-3 digital access and cross-connect system(DACS) is clear. To get simple switching capability among DS-3 lines, a better format is required. [Ref.11]

(B) Syntran

The Syntran has been proposed by Bellcore. Syntran describes how to multiplex DS-0 and DS-1 signals into a DS-3, in a fully synchronous format.

The resulting bit stream has the same rate as the standard. Syntran offers two important features.

- Synchronous format allows easy DACS switching of DS-0 and DS-1 channels among DS-3 ports.

- The aggregate bit rate is compatible with the large number of existing DS-3 transmission facilities installed in North America.

Unfortunately for the global network, DS-3 is seldom found outside of North America. And while M13 and Syntran are defined as electrical interfaces, the new installations of facilities near that speed are mostly optical fiber.[Ref.13]

(C) SONET

SONET(Synchronous Optical Network) is much more than DS-3. The goal is a standard for interconnection of different national networks at speeds from about 150Mbit/s up to several gigabits/s. The DS-3 level is only one portion of the standard, intended for network access from customer premises.

SONET operates at multiples of T3 bandwidth. Initially, products will be offered at the following bit rates (OC stands for Optical Carrier): OC-1, 51.84Mbit/s; OC-3, 155.52Mbit/s; OC-12, 622.08Mbit/s; OC-48, 2.49Gbit/s.

15

As a synchronous standard, SONET is well-suited for switching tributary signals within a higher bandwidth pipe. Initially, switching will be limited to T1 and DS-0 signals, but other service offerings will be defined in the future.

SONET is fully backward-compatible with the ANSI synchronous T3 format; it is partially backward-compatible with order asynchronous T3 equipment such as M13s. It does not affect users of digital microwave radio, since there is little need for a synchronous standard higher than T3 for these applications. [Ref.13]

3. T3 Networks

Implementing a T3 corporate backbone provides users with a measure of flexibility and control over the network. At the low-speed end, circuits may be either 56Kbit/s or 64Kbit/s[Ref.13]. These may be used for voice or data services.

The capability to switch the variety of circuits allows the user to put up and take down circuits of differing rates on the T3 backbone as requirements change, without having to work through an external network provider. This can take the form of reducing lead time for a circuit order, time-of-day circuit changes, or quick response to a temporary overload condition in some part of the network.

T3 networks are based on one or more of the following types of DCS's.

1. Point to point terminals: M13 multiplexers with 28 T1s in and one DS-3 out.

2. Multiple M13 multiplexers in a single chassis: something like a 48-port D4 channel bank that followed the 24-port D3.

3. Networking of multiple DS-3 lines into one hub with the ability to cross-connect DS-1s and even DS-0s within that hub.

Level 3 generally is possible only with one of the synchronous formats, not with classic M13. Devices with this performance are installed at operating companies.

Network topologies for T3 backbones will evolve from point to point up to the partial mesh or interlocking rings familiar from T1 nets. There will be many

fewer sites that can cost-justify multiple DS-3s than there will be T1 hubs. The combination likely to be most common is a sparse DS-3 backbone surrounded by a T1 network. [*Ref*.11]

III. MODEL FORMULATION AND SOLUTION

This chapter discusses how to formulate the minimum cost design of T1/T3 networks, and explains the heuristic procedure for optimal solutions based on the Lagrangian relaxation method.

A. MATHEMATICAL FORMULATION

In order to model the minimum cost design of a network, we consider several things. The amount of traffic between each node and every other node is given and we define the set of circuit types each of which is a mix of T1, T3 and fractional T1. If a network contains n nodes, there would be n(n-2)/2 communicating source-destination pairs that select routes from sets containing from a few thousand to hundreds of thousands of candidate routes, leading to very large problem sizes. Therefore, to simplify the computation, we assume that the maximum length of the path for each pair is less than a specified number of hops. For example, the network designer may specify that routes between two nodes should pass through no more than two additional nodes.

We introduce the following notations to present the mathematical formulation of the problem.

P : The index set of the communicating source-destination pairs

R_p : The index set of candidate routes for the source-destination pair $p \in P$. The set of candidate routes is generated by a route generation algorithm. A route is characterized by the ordered set of links from source to destination node.

L : The index set of candidate links in the T1/T3 network.

T : The index set of circuit types.

X_{pr} : A decision variable which is one if the source-destination pair p takes the route r and zero otherwise.

18

Z_{lt} : A decision variable which is one if the link l is supported by the circuit type t and zero otherwise.

Q_t : The number of DS-0 channels that can be carried over a circuit type t.

a_p : The number of DS-0 channels demanded by the source-destination pair p.

C_{lt} : The costs for leasing a circuit type t on link l.

I_{rl} : An indicator function which is one if link l is used in route r and zero otherwise.

Using the above notations, we formulate the minimum cost design problem as follows:

Problem P:

$$Z_P = \min \sum_{l \in L} \sum_{t \in T} C_{lt} Z_{lt} \qquad (1)$$

subject to

$$\sum_{r \in R_p} X_{pr} = 1 \qquad \forall p \in P \qquad (2)$$

$$\sum_{p \in P} \sum_{r \in R_p} a_p I_{rl} X_{pr} \leq \sum_{t \in T} Q_t Z_{lt} \qquad \forall l \in L \qquad (3)$$

$$\sum_{t \in T} Z_{lt} \leq 1 \qquad \forall l \in L \qquad (4)$$

$$0 \leq X_{pr} \leq 1, \qquad 0 \leq Z_{lt} \leq 1 \qquad integer \qquad (5)$$

The constraint (2) ensures that each source-destination pair selects exactly one route for transmission. The constraint (3) ensures that the number of DS-0 channels on each link does not exceed its capacity. The selection of at most one circuit type for each link is ensured by constraint (4). *Problem P* is a linear zero-one integer programming problem which can be shown to be NP-hard.

Based on the current status of the computational ability, the idea of solving *Problem P* optimally is rejected. The idea of using piecewise linearization of the objective function and solving the linear programming problem is also rejected for

the reason that the linear relaxation of *Problem P* is very large and its optimal solution requires a significant computational effort and may generate a fractional solution.

B. LAGRANGIAN RELAXATION

One of the most computationally useful ideas of the 1970s is the observation that many hard problems can be viewed as easy problems complicated by a relatively small set of side constraints. Dualizing the side constraints produces a Lagrangian problem that is easy to solve and whose optimal value is a lower bound for minimization problems on the optimal value of the original problem. The Lagrangian problem can thus be used in place of a linear programming relaxation to provide bounds in a branch and bound algorithm. The Lagrangian approach offers a number of important advantages over linear relaxation.

Lagrangian methods had gained considerable currency by the 1974 when Geoffrion coined the perfect name for this approach - *Lagrangian relaxation*. Since then the list of applications of Lagrangian relaxation has grown to include over a dozen of the most infamous combinatorial optimization problems. For most of these problems, Lagrangian relaxation has provided the best existing algorithm and has enabled the solution of problems of practical size. [*Ref*.3]

In the previous section, we already made a mathematical formulation for minimum cost design problem. We know that there are two natural Lagrangian relaxation for *Problem P*. The first is obtained by multipling the constraint (3) by a vector of positive Lagrange multipliers, U_l, $l \in L$, and adding them to the objective function.

Problem D_1:

$$
\begin{aligned}
Z_{D1}(U) &= \min \sum_{l \in L} \sum_{t \in T} C_{lt} Z_{lt} + \sum_{l \in L} U_l \left[\sum_{p \in P} \sum_{r \in R_p} a_p I_{rl} X_{pr} - \sum_{t \in T} Q_t Z_{lt} \right] \\
&= \min \sum_{l \in L} \sum_{t \in T} (C_{lt} - U_l Q_t) Z_{lt} + \sum_{l \in L} \sum_{p \in P} \sum_{r \in R_p} U_l a_p I_{rl} X_{pr} \qquad (6)
\end{aligned}
$$

subject to

20

$$\sum_{r \in R_p} X_{pr} = 1 \qquad \forall p \in P$$

$$\sum_{t \in T} Z_{lt} \le 1 \qquad \forall l \in L$$

$$0 \le X_{pr} \le 1, \qquad 0 \le Z_{lt} \le 1 \qquad integer$$

The set of feasible solutions for *Problem P* is a subset of the set of feasible solutions for the Lagrangian relaxation D_1. The positivity of U ensures that in any feasible solution for *Problem P*, the expression

$$\sum_{l \in L} U_l \left[\sum_{p \in P} \sum_{r \in R_p} a_p I_{rl} X_{pr} - \sum_{t \in T} Q_t Z_{lt} \right]$$

is nonpositive, and thus, the value of the objective function in (6) is never greater than the value of the objective function in *Problem P*. Thus, whenever *Problem P* has a feasible solution, $Z_{D1}(U)$ is less than or equal to Z_P. For each vector of multipliers U, $Z_{D1}(U)$ is a lower bound for Z_P.

The second relaxation is obtained by dualizing constraint (2) with nonpositive Lagrange multipliers, V_p, $p \in P$.

Problem D_2:

$$Z_{D2}(V) = \min \sum_{l \in L} \sum_{t \in T} C_{lt} Z_{lt} + \sum_{p \in P} V_p \left[\sum_{r \in R_p} X_{pr} - 1 \right] \qquad (7)$$

subject to

$$\sum_{p \in P} \sum_{r \in R_p} a_p I_{rl} X_{pr} \le \sum_{t \in T} Q_t Z_{lt} \qquad \forall l \in L$$

$$\sum_{t \in T} Z_{lt} \le 1 \qquad \forall l \in L$$

$$0 \le X_{pr} \le 1, \qquad 0 \le Z_{lt} \le 1 \qquad integer$$

Comparing above two Lagrangian relaxations, we see that the second relaxation is harder to solve but might provide better bounds. This thesis uses the first Lagrangian

21

relaxation for implementation and computational results.

C. SUBGRADIENT OPTIMIZATION PROCEDURE

Several methods have been suggested in the literature for computing a vector of optimal Lagrange multipliers. These include various versions of the simplex method implemented using column generation procedures, dual ascent and multiplier adjustment procedures, and a subgradient optimization procedure[Ref.3]. Subgradient optimization methods have been shown to be very effective in a variety of combinatorial optimization problems such as the traveling salesman problem, topological design of computer communication systems, and lot-sizing algorithms for complex product structures[Ref.1]. Therefore, we decided to employ subgradient optimization techniques.

The subgradient method is a brazen adaptation of the gradient method in which gradients are replaced by subgradients.

Let $(X_{pr}(U), Z_{lt}(U))$ be the optimal solution to the Lagrangian problem D_1 for a fixed vector U. A subgradient is the vector with coordinates

$$\gamma_l(U) = \sum_{p \in P} \sum_{r \in R_p} a_p I_{rl} X_{pr}(U) - \sum_{t \in T} Q_t Z_{lt}(U) \qquad \forall l \in L.$$

Given an initial value U^0, a sequence U^k is generated by the iterative formula

$$U_l^{k+1} = U_l^k + t_k \gamma_l^k$$

where t_k is a positive scalar stepsize. It is known that U_l^k converges to the optimal Lagrange multiplier U^* provided that t_k converges to 0 and $\sum t_k$ diverges. The stepsize most commonly used in practice is

$$t_k = \frac{\lambda_k \left[\hat{Z}_P - Z_{D1}(U^k) \right]}{\| \gamma_l^k \|^2}$$

where λ_k is a scalar satisfying $0 < \lambda_k \le 2$ and $\hat{Z}_P$ is an upper bound on $Z_{D1}(U)$, frequently obtained by applying a heuristic to *Problem P*. Often the sequence λ_k is determined by setting $\lambda_0 = 2$ and halving λ_k whenever $Z_{D1}(U)$ has failed to increase

22

in some fixed number of iterations. This rule has performed well empirically, even though it is not guaranteed to satisfy the sufficient condition given above for optimal convergence.

The steps involved in the subgradient optimization procedure are as follows.

1. Initialization:

   (a) Using a heuristic, compute an overestimate $\hat{Z}_P$ of Z_P or set $\hat{Z}_P$ to an arbitrarily large value.

   (b) Select an initial set of multipliers U^0; and set K, the iteration counter, to 0, and the improvement counter to 0; and λ a parameter for adjusting the stepsize to λ^0 (an arbitrary positive initial value, e.g., 2).

2. Solving the Lagrangian problem:

   (a) Increment the improvement and iteration counters by 1.

   (b) Solve the Lagrangian problem using U^k as the Lagrange multipliers. Thus, obtain $Z_{D1}(U^k)$ and X_{pr}^K, Z_{lt}^k.

3. Testing and updating parameters:

   (a) If $Z_{D1}(U^k)$ is greater than the current best value of $Z_{D1}(U)$ then replace the current best value of $Z_{D1}(U)$ by $Z_{D1}(U^k)$ and set U^* to U^k. Also reset the improvement counter to -1.

   (b) If X_{pr}^k and Z_{lt}^k are feasible for *Problem P*, compute the value of its associated objective function for *Problem P*. If this value is less than the current value of $\hat{Z}_P$, then set $\hat{Z}_P$ to this value.

   (c) If the improvement counter has reached a prespecified upper limit, then set λ to $\lambda/2$, U^k to U^*, the improvement counter to 0 and go to 2(a).

   (d) If the iteration counter has exceeded a prespecified limit, if λ is less than a prespecified limit, if t_k is less than a prespecified limit, or if $(\hat{Z}_P - Z_{D1}(U^*))/Z_{D1}(U^*)$ is less than a prespecified error tolerance, then stop.

23

4. Updating the multipliers :

(a) Compute a new subgradient.

$$\gamma_l^k = \sum_{p \in P} \sum_{r \in R_p} a_p I_{rl} X_{pr}^k - \sum_{t \in T} Q_t Z_{lt}^k \qquad \forall l \in L$$

(b) Compute the new stepsize.

$$t_k = \frac{\lambda \left[\hat{Z}_P - Z_{D1}(U^k) \right]}{\| \gamma_l^k \|^2}.$$

(c) Compute the new multipliers.

$$U_l^{k+1} = U_l^k + t_k \gamma_l^k \qquad \forall l \in L$$

(d) set k to $k + 1$.

5. Go to 2.

D. SOLVING THE LAGRANGIAN PROBLEM

The computational efficiency of the subgradient optimization procedure depends on our ability to solve efficiently the Lagrangian problem which is generated in step 2(b) of the subgradient optimization procedure. Fortunately, for a fixed set of multipliers, the Lagrangian problem is separable into subproblems which are readily solved.

Since there are no coupling constraints between the X_{pr}, and Z_{lt} variables, the Lagrangian problem D_1 can be decomposed into two subproblems as $Z_{D1}(U) = Z_{D11}(U) + Z_{D12}(U)$ where

Subproblem D_{11}:

$$Z_{D11}(U) = \min \sum_{l \in L} \sum_{p \in P} \sum_{r \in R_p} U_l a_p I_{rl} X_{pr} \qquad (8)$$

subject to

$$\sum_{r \in R_p} X_{pr} = 1 \qquad \forall p \in P$$

$$0 \le X_{pr} \le 1 \qquad integer$$

24

and

Subproblem D_{12}:

$$Z_{D12}(U) = \min \sum_{l \in L} \sum_{t \in T} (C_{lt} - U_l Q_t) Z_{lt} \qquad (9)$$

subject to

$$\sum_{t \in T} Z_{lt} \leq 1 \qquad \forall l \in L$$

$$0 \leq Z_{lt} \leq 1 \qquad integer$$

Subproblem D_{11} can be separated into $\mid P \mid$ subproblems, one for each pair, where the subproblem associated with the pth pair is

$$Z_{D11}^p(U) = \min a_p \sum_{r \in R_p} \left[\sum_{l \in L} U_l I_{rl} \right] X_{pr}$$

subject to

$$\sum_{r \in R_p} X_{pr} = 1$$

$$0 \leq X_{pr} \leq 1 \, integer$$

and then $Z_{D11}(U) = \sum_{p \in P} Z_{D11}^p(U)$.

Subproblem D_{12} can also be separated into $\mid L \mid$ subproblems, one for each link, where the subproblem associated with the lth link is

$$Z_{D12}^l(U) = \min \sum_{t \in T} (C_{lt} - U_l Q_t) Z_{lt}$$

subject to

$$\sum_{t \in T} Z_{lt} \leq 1$$

$$0 \leq Z_{lt} \leq 1 \qquad integer$$

25

and then $Z_{D12}(U) = \sum_{t \in T} Z_{D12}^t(U)$.

E. IMPLEMENTATION PROCEDURE

To test the applicability of the subgradient optimization procedure to network design problems and to obtain estimates on the quality of bounds and solutions generated, this thesis implemented the procedure proposed in previous section in turbo Pascal on IBM PC computer.

1. System Flow

The program consists of two main parts. The first is data generation, and the second is problem solving. Figure 8 shows the system flow. The program is initiated by entering the number of nodes. It then creates data file with source-destination node pairs, and their distance and amount of data traffic. The user is then prompted to enter a number that he/she wantsa to generate a set of candidate routes. Next, the user is prompted to specify a number of subgradient parameters.

Two types of iterations, major and minor, are defined in the program. A major iteration consists of many minor iterations, each of which is an evaluation of the current value of the objective function, a choice of subgradient direction and modification of the value of U. The minor iterations terminate when one of the stopping conditions is satisfied such as the difference between the best feasible solution and the lower bound based on the Lagrangian dual is small, the subgradient is small or an upper limit on the number of iterations has been reached.

Throughout the optimization process, the program updates the Lagrangian value, the multipliers generated in each major iterations and the best feasible solution yet generated.

Based on the output generated after a major iteration, the user may terminate or reenter the subgradient optimization procedure by changing the number of routes, iteration counter or improvement counter. This iterative process continues until the solution does not get improved.

26
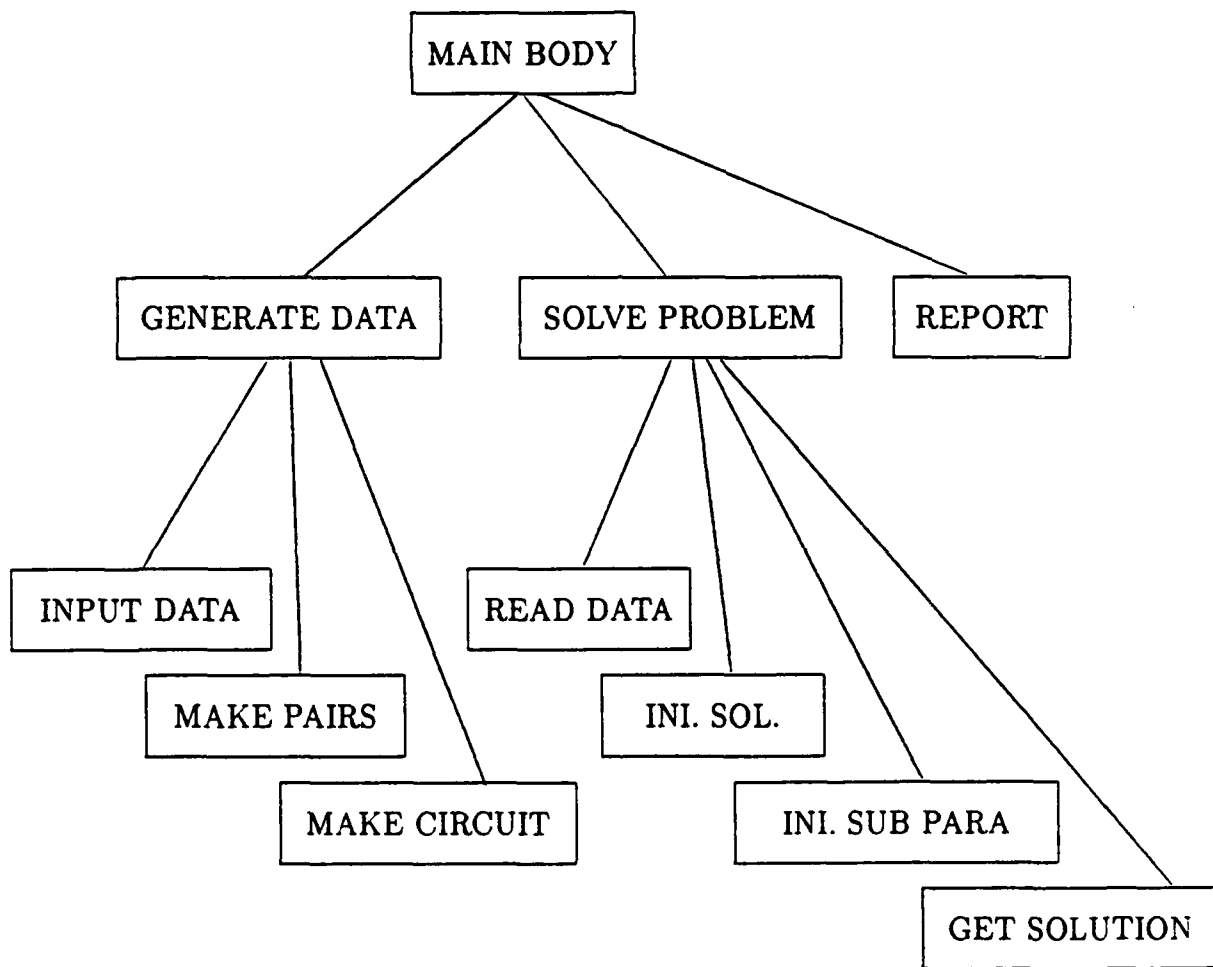
Figure 8: System Chart

2. Heuristic

The subgradient optimization procedure requires an overestimate $\hat{Z}_{IP}$ on $Z_{D1}(U^*)$ used to update the Lagrange multipliers. This overestimate can be a feasible solution generated by a heuristic. Figure 9 shows the heuristic that has been used in this thesis to generate good feasible solutions.

In each minor iteration of the subgradient optimization procedure, dual solution X_{pr}^k and Z_{lt}^k are obtained. These solutions might be infeasible because a set of capacity constraint (3) was relaxed. This constraint, therefore, may be violated by the dual solution X_{pr}^k and Z_{lt}^k,

$$\sum_{p \in P} \sum_{r \in R_p} a_p I_{rl} X_{pr} > \sum_{t \in T} Q_t Z_{lt} \qquad \exists l \in L.$$

By correcting this infeasibility situation for all l which has been violated,

$$\sum_{p \in P} \sum_{r \in R_p} a_p I_{rl} X_{pr} \leq \sum_{t \in T} Q_t Z_{lt} \qquad \forall l \in L,$$

the solution becomes feasible.

If this feasible solution is smaller than previous best feasible solution which is upper bound, it replaces the previous solution and becomes the new best feasible solution.

INPUT : Infeasible Dual Solution X_{pr}^k and Z_{lt}^k
OUTPUT: Feasible Solution

BEGIN

    OBJECTIVE := 0;

    FOR each Link l

       BEGIN

          t := 0;

          WHILE $a_p I_{rl} X_{pr} > Q_t$ DO

            t := t + 1;

          Z_{lt}^k := t;

          OBJECTIVE := OBJECTIVE + $C_{lt} Z_{lt}$

       END; (FOR LOOP)

    IF OBJECTIVE < UPPER-BOUND THEN

       UPPER-BOUND := OBJECTIVE;

END. (HEURISTIC)

Figure 9: Heuristic

IV. COMPUTATIONAL RESULTS

The solution approach proposed in Chapter III has been tested using network design problems.

A. Computational Experiments

The procedure was tested under two situations: Under one situation [case A], the cost of T3 circuit was 9 times of cost of T1. The other situation [case B] was such that the cost of T3 circuit was 5 times of T1. Under each situation, we have tested several networks that have different number of nodes. To find better feasible solutions, we have tested this procedure several times changing the number of candidate routes — 8, 12, 16 and 20 candidate routes.

The distance and amount of traffic for each of source-destination node pairs and circuits information were randomly generated by the random number generation procedure. Figure 10 shows one of sample data used.

The procedure has the number of subgradient parameters. For the test, we initially set λ^0 to 0.5 and U^0 to 0. The stopping criteria parameters are set as follows: stepsize limit to 0.0001, error limit to 0.005, and iteration limit to 100.

B. Summary of Computational Results

The results of the computational tests on the five randomly generated problems are summarized in the Table 1 and 2. The values shown in the tables represent the feasible solutions for network with 6, 10, 15, 20 and 25 nodes. The asterisk means that the previous feasible solution is not improved.

While testing the procedure, we had some problems with the size of main memory and the processing times. One of the problems was memory overflow error when we were trying to enlarge the number of node, the number of candidate route of each pair, and/or the number of circuit. Since it takes $O(n^{n-2})$ times to locate all candidate

30

```
           node pair information
      source dest.   distance   traffic
      ------------------------------------
         1      2       443          0
         1      3       592         60
         1      4       488         71
         1      5       516          0
         1      6       648          4
         2      3       149          0
         2      4       607          0
         2      5       319         53
         2      6       539          0
         3      4       695          0
         3      5       351         61
         3      6       562         70
         4      5       371         64
         4      6       298         46
         5      6       220         97

      circuits info.
      capa. cost
      -----------------
         6     10
        12     15
        18     19
        24     21
        30     31
              . . .

      1554    625
      1560    627
      1566    637
      1572    642
      1578    646
```

Figure 10: Sample Input Data
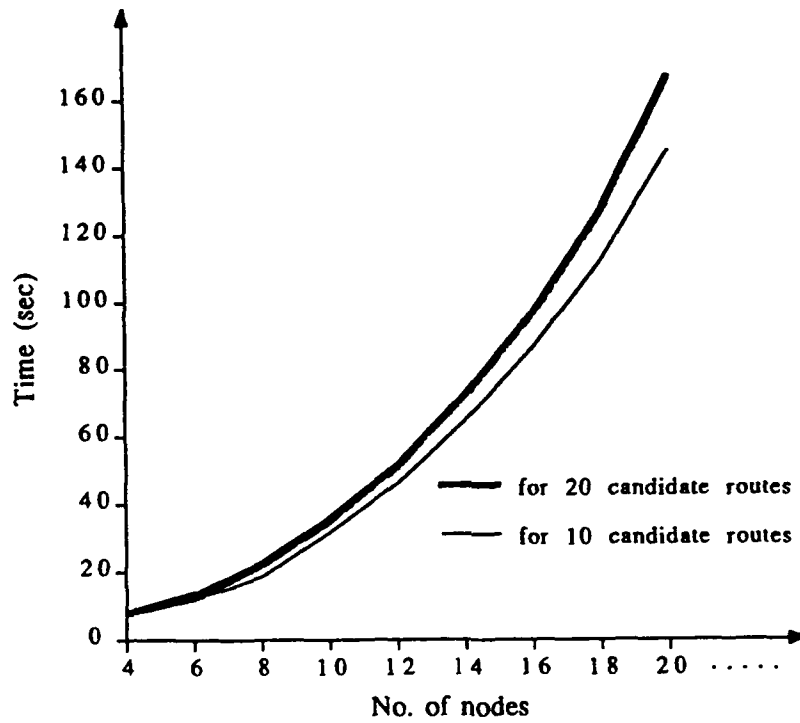
Figure 11: Graph of Time Complexity

routes, we limited the maximum number of hops of candidate routes to 4 and the maximum number of nodes to 25. Figure 11 shows how much time is required as the number of nodes increases.

Table 1 and 2 summarized the results obtained for networks with the different candidate routes. In most of cases, the feasible solution improve when the number of candidate routes increases.

Ni	Nc	1	2	3	4	5
6	8	180175	253761	242297	185416	221353
	12	178883	*	238662	185119	*
	16	*	*	*	*	*
	20	178001	253413	*	183249	*
10	8	629591	505213	530912	613584	569267
	12	*	486110	530421	*	*
	16	622310	449811	522941	577069	568054
	20	621037	*	*	*	547983
15	8	1415404	1678792	1110691	1453550	1373344
	12	1414896	1605410	1060431	1398207	1291821
	16	1388078	1474272	1048101	1382763	*
	20	1343018	*	*	1355496	1240806
20	8	2262668	2278370	2501282	2645749	2226695
	12	2162049	2227108	2468560	2578408	2148869
	16	2114736	2176106	*	2542664	2098762
	20	*	*	2460715	2509654	*
25	8	3701524	3482153	3497099	3724747	4357936
	12	3595955	3474732	3302242	3712604	4258339
	16	*	3382492	*	3685839	*
	20	*	*	*	*	4176299

N_i : Number of nodes
N_c : Number of candidate routes
      for each node pair
* : Indication for nonimprovement

Table 1: Summary of Feasible Solutions : Case A

Ni	Nc	1	2	3	4	5
6	8	127895	185711	138250	171362	129688
	12	*	*	*	*	*
	16	125755	*	136375	*	*
	20	*	178148	134125	171363	*
10	8	436919	378817	378229	438411	437988
	12	*	348660	*	417009	410846
	16	428966	*	*	377508	*
	20	*	326292	361724	*	*
15	8	1015946	1184250	770354	1000964	954370
	12	979128	1156759	*	*	909834
	16	*	*	751964	973137	*
	20	944654	1055800	*	947506	894031
20	8	1610162	1682611	1859701	1888310	1744943
	12	1500080	1496639	1771325	1749449	1507031
	16	*	*	1682985	*	*
	20	*	*	*	1708854	1495353
25	8	2730969	2678297	2645483	2628391	3166161
	12	2613587	2578349	2408956	*	3048471
	16	2438277	2508159	2391261	*	2985384
	20	*	*	*	2522723	*

Table 2: Summary of Feasible Solutions : Case B

V. CONCLUSION

We have made a mathematical formulation for minimum cost design of digital communication network. This thesis used the Lagrangian relaxation and the subgradient optimization procedure to obtain lower bounds and feasible solutions in minimizing the cost of network design with given node location and traffic of each node pairs. Computational experience indicates that this procedure is effective in most instances. The quality of bound generated depends to a large extent on the limits set on the improvement and iteration counters. Low settings terminate the procedure before it reaches a good feasible solution, while high settings consume excessive computing resources. Good settings of these parameters require careful balancing between these two consideration.

The difference between upper bound and lower bound was fifty to seventy percent in our experiments. Our conjecture is that the other constraint relaxation is required to get a better bound. This thesis did not explore that constraint relaxation.

Our model can be extended by considering reliability constraints, which is left for future research.

APPENDIX A: SOURCE CODE

A. DATA STRUCTURE DESCRIPTION

```
{***************************************************}
{*    TITLE     : Minimum Cost Design of Network    *}
{*    AUTHOR    : Hwang, Yong Goo                    *}
{*    DATE      : 13 June 1990                       *}
{*    REVISE    : 15 Nov. 1990                       *}
{*    DESCRIPTION : This program is to get best      *}
{*       feasible solution for minimum cost design of *}
{*       network. It has been implemented in Turbo   *}
{*       Pascal and run on IBM PC computer.          *}
{*       System flow is as follows:                  *}
{*          1.   Generate data and store it to 'net.dat' *}
{*               file.                                *}
{*               -  node pairs information            *}
{*               -  circuit information               *}
{*          2.   Get data from 'net.dat' file         *}
{*               -  make route table                  *}
{*          3.   Solve the problem                    *}
{*          4.   Out report report                    *}
{***************************************************}

PROGRAM NETWORK_DESIGN;

USES CRT, GRAPH;

CONST
   VLARGE = 1e20;
   MAX_I =    20;     ( maximum No. of nodes )
   MAX_P =   190;
   MAX_R = 6000;
   MAX_T =   200;     ( maximum size of circuit )
   max_nc = 20;       ( maximum No. of candidate routes )
   F_NAME = 'net.dat';
   FILE_NAME = 'out.dat';

TYPE
   NODE = 1..MAX_I;
   PAIR = 0..MAX_P;
   CNFG = 0..MAX_T;
   POINT = ARRAY[1..100] OF INTEGER;
```

```
CIRCUTE = RECORD
            FT_1 : INTEGER;
            T_1  : INTEGER;
            T_3  : INTEGER;
            CAPACITY: INTEGER;
            COST : INTEGER;
          END;

CIRCUTE_TABLE = ARRAY[0..MAX_T] OF CIRCUTE;

SD_PAIR = RECORD
            S : INTEGER;
            D : INTEGER
          END;

PAIR_TABLE = ARRAY[1..MAX_P] OF SD_PAIR;

VAR
  i: NODE;
  p: PAIR;
  t: CNFG;
  r,ni,np,nr,nt,nc : INTEGER;

  ORGN,DSTN: array[1..MAX_P] of NODE;
  DISTANCE,TRAFFIC: array[1..MAX_P] of REAL;
  CAPA,RATE: array[0..MAX_T] of REAL;
  ROUTE: array[0..MAX_R,1..4] of PAIR;
  px,dx: array[1..MAX_P] of 0..MAX_R;
  pz,dz: array[0..MAX_P] of CNFG;

  ITRCTR,ITRLIM,IMPCTR,IMPLIM: INTEGER;
  UPPBND,LOWBND,pobj,dobj: REAL;
  NORM,ERRORLIM,LAMBDA,LAMBDALIM,STEPSIZE: REAL;
  ALPHA,BEST_ALPHA,SUBGRAD: array[0..MAX_P] of REAL;
  OPTIMAL,TERMINATED,FRAG1: BOOLEAN;
  DATA_FILE : TEXT;
  POINT_X, POINT_Y : POINT;
  T_TABLE : CIRCUTE_TABLE;
  S_D_PAIR : PAIR_TABLE;
  ANS1 : CHAR;
```

B. DATA GENERATION PROCEDURE

```
(**********************************************************)
(*    TITLE    : Generate Data                         *)
(*    AUTHOR   : Hwang, Yong Goo                        *)
(*    DATE     : 15 June 1990                           *)
(*    REVISE   : 15 July 1990                           *)
(*    DESCRIPTION : This procedure will make a data     *)
(*        file that consists of node pair informations  *)
(*        and circuit informations with random genera-  *)
(*        tion procedure                                *)
(**********************************************************)

PROCEDURE GENERATE_DATA;

VAR
  t : INTEGER;


  (********************************************************)
  PROCEDURE T_TABLE_GENERATION;

  VAR
    i, j, k : INTEGER;

  BEGIN

    nt := 0;
    FOR i := 0 TO 2 DO
      FOR j := 0 TO 9 DO
        FOR k := 0 TO 3 DO
          BEGIN
            T_TABLE[nt].FT_1 := k;
            T_TABLE[nt].T_1 := j;
            T_TABLE[nt].T_3 := i;
            T_TABLE[nt].CAPACITY := k*6 + j*24 + i*672;
            CASE k OF
              1 : T_TABLE[nt].COST := 10 + j*21 + i*219;
              2 : T_TABLE[nt].COST := 15 + j*21 + i*219;
              3 : T_TABLE[nt].COST := 19 + j*21 + i*219;
              ELSE
```

```
                    T_TABLE[nt].COST := j*21 + i*219;
           END;
           nt := nt + 1;
        END;

END;


{*******************************************}
FUNCTION RANDINT(MIN,MAX : INTEGER): INTEGER;

  {* The RANDINT function suppliers a random integer within
     the range of the specified arguments min and max.
     For example, RANDINT(10,20) gives a random integer
     between 10 and 20. *}
CONST
  MININT = -5000;

VAR
  RANDRANGE : INTEGER;

BEGIN

  IF (MIN < MININT) THEN
     MIN := MININT;
  RANDRANGE := MAX - MIN + 1;
  RANDINT := RANDOM(RANDRANGE) + MIN;

END;


{**************************}
PROCEDURE NODE_GENERATION;

VAR
  n : INTEGER;

BEGIN

  FOR n := 1 TO ni DO
    BEGIN
      POINT_X[n] := RANDINT(50,1000);
      POINT_Y[n] := RANDINT(30,800);
    END;
```

```pascal
    END;


(*************************)
PROCEDURE PAIR_GENERATION;

VAR
   j, k, m, p : INTEGER;
   DISTANCES : REAL;
   TRAFFIC_VOLUMN, TEMP_TRAFFIC_VOLUMN : INTEGER;
   X, Y : REAL;

BEGIN

   p := 0;
   FOR j := 1 TO (ni - 1) DO
     BEGIN
       m := j + 1;
       FOR k := m TO ni DO
         BEGIN
           p := p + 1;
           S_D_PAIR[p].S := j;
           S_D_PAIR[p].D := k;
           X := POINT_X[j] - POINT_X[k];
           Y := POINT_Y[j] - POINT_Y[k];
           DISTANCES := SQRT((X * X) + (Y * Y));
           TEMP_TRAFFIC_VOLUMN := RANDINT(-30,100);
           IF (TEMP_TRAFFIC_VOLUMN < 0) THEN
               TRAFFIC_VOLUMN := 0
           ELSE
               TRAFFIC_VOLUMN := TEMP_TRAFFIC_VOLUMN;
           WRITELN(DATA_FILE, j:5, k:5, DISTANCES:10:0,
                                       TRAFFIC_VOLUMN:10);
         END;
     END;

   END;

(*   main part of data generation   *)
BEGIN

   ASSIGN(DATA_FILE,F_NAME);
   REWRITE(DATA_FILE);
   RANDOMIZE;
   WRITELN('    Enter number of NODEs? ');
```

40

```
   READLN(ni);
   np := ni*(ni-1) DIV 2;
   T_TABLE_GENERATION;
   (*  WRITELN(DATA_FILE, ni:5, np:5, nt-1:5); *)
   NODE_GENERATION;
   PAIR_GENERATION;
   FOR t := 1 TO (nt-1) DO
     WRITELN(DATA_FILE, T_TABLE[t].CAPACITY:6,
                                       T_TABLE[t].COST: 5);
   CLOSE(DATA_FILE);

END;
```

C. SOLVING LAGRANGIAN PROBLEM PROCEDURE

```
(*********************************************************)
(*    TITLE       : Generate source-destination pair     *)
(*    AUTHOR      : Hwang, Yong Goo                       *)
(*    DATE        : 18 July 1990                          *)
(*    REVISE      : 18 July 1990                          *)
(*    DESCRIPTION : This function provides a node pair *)
(*********************************************************)

FUNCTION p_(i,j: NODE): PAIR;

VAR
  o,d : NODE;
  p : INTEGER;

BEGIN

  IF i < j THEN
    BEGIN
      o:=i;
      d:=j;
    END
  ELSE
    BEGIN
      o:=j;
      d:=i;
    END;
  p:=0;
  REPEAT
    p:=p+1;
  UNTIL (ORGN[p] = o) and (DSTN[p] = d);
  p_:=p;

END;


(*********************************************************)
(*    TITLE       : Get data and Make route table        *)
(*    AUTHOR      : Hwang, Yong Goo                       *)
(*    DATE        : 18 July 1990                          *)
(*    REVISE      : 12 Oct. 1990                          *)
```

42

```
(*    DESCRIPTION : This procedure make route table    *)
(*          from 'net.dat' file                        *)
(****************************************************)

PROCEDURE GET_DATA;

VAR
  p: PAIR;
  t: CNFG;
  datf: text;

  (***********************)
  PROCEDURE BUILD_ROUTE_TABLE;

  TYPE
    rte = RECORD
            l1,l2,l3,l4: PAIR;
            ds: REAL;
          END;
    ptr = ^rte;

  VAR
    i,j,k: NODE;
    p,lk1,lk2,lk3,lk4: PAIR;
    c: INTEGER;
    dists: REAL;
    ptrs: ARRAY[1..max_nc] of ptr;
    aptr: ptr;
    nset: SET OF NODE;
    LONGER: BOOLEAN;

    (********************************)
    PROCEDURE REVISE_PTRS (k: INTEGER);

    VAR
      c: INTEGER;

    BEGIN

      IF k < nc THEN
        IF dists < ptrs[k]^.ds THEN
          BEGIN
            dispose(ptrs[nc-1]);
            FOR c:= nc-1 downto k+1 do
                      ptrs[c]:=ptrs[c-1];
```

43

```
            new(aptr);
            with aptr^ do
              BEGIN
                l1:=lk1;  l2:=lk2;  l3:=lk3;  l4:=lk4;
                ds:=dists;
              END;
            ptrs[k]:=aptr;
          END
        ELSE
          BEGIN
            REVISE_PTRS (k+1);
          END
      ELSE
        LONGER:=true;

  END;


BEGIN  (* main of build route table                      *)
       (* create number of nc shortest routes            *)
       (* - with the shortest being the direct link.     *)
  WRITELN(' Wait a moment !! ');
  FOR p:=1 to np do
    BEGIN
      FOR c:=1 to nc-1 do
        BEGIN
          new(aptr);
          aptr^.ds:=VLARGE*10;
          ptrs[c]:=aptr;
        END;
      nset:=[1..ni];
      nset:=nset-[ORGN[p]];
      nset:=nset-[DSTN[p]];
      FOR i:=1 to ni do
        IF i in nset THEN
          BEGIN
            lk1:=p_(ORGN[p],i);
            lk2:=p_(i,DSTN[p]);
            lk3:=0;
            lk4:=0;
            dists:=DISTANCE[lk1]+DISTANCE[lk2];
            LONGER:=false;
            REVISE_PTRS (1);
            IF LONGER = false THEN
              FOR j:=1 to ni do
```

44

```pascal
                    IF j in nset-[i] THEN
                      BEGIN
                        lk2:=p_(i,j);
                        lk3:=p_(j,DSTN[p]);
                        lk4:=0;
                        dists:=DISTANCE[lk1]+DISTANCE[lk2]
                                            +DISTANCE[lk3];

                        REVISE_PTRS (1);
                        IF LONGER = false THEN
                          FOR k:=1 to ni do
                            IF k in (nset-[i])-[j] THEN
                              BEGIN
                                lk3:=p_(j,k);
                                lk4:=p_(k,DSTN[p]);
                                dists:=DISTANCE[lk1]+DISTANCE[lk2]
                                        +DISTANCE[lk3]+DISTANCE[lk4];
                                REVISE_PTRS (1);
                              END;
                    END;
            END;
      ROUTE[(p-1)*nc,1]:=p;
      ROUTE[(p-1)*nc,2]:=0;
      ROUTE[(p-1)*nc,3]:=0;
      ROUTE[(p-1)*nc,4]:=0;
      FOR c:=1 to nc-1 do
        BEGIN
          ROUTE[(p-1)*nc+c,1]:=ptrs[c]^.l1;
          ROUTE[(p-1)*nc+c,2]:=ptrs[c]^.l2;
          ROUTE[(p-1)*nc+c,3]:=ptrs[c]^.l3;
          ROUTE[(p-1)*nc+c,4]:=ptrs[c]^.l4;
        END;

    END; (* FOR p *)

  FOR c:=1 to nc-1 do
    BEGIN
      dispose(ptrs[c]);
      ptrs[c]:=nil;
    END;

  END;    (* end of build route table *)


BEGIN     (* main of get data *)
```

```
    ASSIGN(datf,F_NAME);

    RESET(datf);

    FOR p:=1 to np do
      READLN(datf,ORGN[p],DSTN[p],DISTANCE[p],TRAFFIC[p]);

    WRITELN(' Enter the No. of Candidate routes for each Pair :');

    READLN(nc);

    CAPA[0]:=0;

    RATE[0]:=0;

    FOR t:=1 to nt do
      READLN(datf,CAPA[t],RATE[t]);

    BUILD_ROUTE_TABLE;
END;



(***********************************************************)
(*    TITLE     : Find initial feasible solution       *)
(*    AUTHOR    : Hwang, Yong Goo                       *)
(*    DATE      : 2 Aug. 1990                           *)
(*    REVISE    : 15 Aug. 1990                          *)
(*    DESCRIPTION : This procedure gives us initial     *)
(*         feasible solution for problem. This          *)
(*         solution may be fully connected network.     *)
(***********************************************************)

PROCEDURE FIND_INITIAL_FEASIBLE_SOLUTION;

VAR
  p: PAIR;
  t: CNFG;

BEGIN

  pobj:=0;
  FOR p:=1 to np do
    BEGIN
```

```
      t:=0;
      IF TRAFFIC[p] > 0 THEN
        BEGIN
          REPEAT
            t:=t+1;
          UNTIL TRAFFIC[p] <= CAPA[t];
        END;
      px[p]:=(p-1)*nc;
      pz[p]:=t;
      pobj:=pobj+DISTANCE[p]*RATE[t];
    END;
  UPPBND:=pobj;

END;




(*****************************************************)
(*    TITLE      : Initiate subgradient parameters   *)
(*    AUTHOR     : Hwang, Yong Goo                    *)
(*    DATE       : 13 June 1990                       *)
(*    REVISE     : 15 Nov. 1990                       *)
(*    DESCRIPTION : At first processing, this         *)
(*         procedure initiates sub. parameters        *)
(*         autometically, and then when user reenter  *)
(*         the system, he can modify them.            *)
(*****************************************************)

PROCEDURE PREP_FOR_SUBGRADIENT_ITERATION;

VAR
  p : PAIR;
  ANS : CHAR;

BEGIN

  IF FRAG1 THEN
    BEGIN
      LAMBDA:= 0.5;
      IMPLIM:= 25;
      ITRLIM:=100;
      ERRORLIM:=0.0050;
      LAMBDALIM:=0.0001;
      ITRCTR:=0;
      IMPCTR:=0;
```

```
      LOWBND:=0;
      FOR p:=0 to np do
            ALPHA[p]:=0;
      FRAG1 := FALSE;
    END
  ELSE
    BEGIN
      WRITELN('Do you want to change some subgradient parameters? ');
      READLN(ANS);
      IF (ANS = 'y') OR (ANS = 'Y') THEN
        BEGIN
          WRITELN('  LAMBDA : ', LAMBDA :4:2);
          WRITELN('  Enter the LAMBDA? ');
          READLN(LAMBDA);
          WRITELN('  IMPROVEMENT COUNTER LIMIT : ', IMPLIM :4);
          WRITELN('  Enter the IMPROVEMENT COUNTER LIMIT? ');
          READLN(IMPLIM);
          WRITELN('  LAMBDA LIMIT : ', LAMBDALIM :4:6);
          WRITELN(' Enter the LAMBDA LIMIT? ');
          READLN(LAMBDALIM);
          WRITELN('  ITERATION COUNTER LIMIT : ', ITRLIM :4);
          WRITELN(' Enter the ITERATION COUNTER LIMIT? ');
          READLN(ITRLIM);
        END
      ELSE
        BEGIN
          LAMBDA:= 0.5;
          IMPLIM:= 25;
          ITRLIM:=100;
          LAMBDALIM:=0.0001;
        END;
      ERRORLIM:=0.0050;
      ITRCTR:=0;
      IMPCTR:=0;
      LOWBND:=0;
      FOR p:=0 to np do
            ALPHA[p]:=0;
    END;

END;



(*******************************************************)
(*   TITLE    : Solve the Lagrangian dual            *)
```

```
(*    AUTHOR    : Hwang, Yong Goo                       *)
(*    DATE      : 21 Sep. 1990                          *)
(*    REVISE    : 11 Nov. 1990                          *)
(*    DESCRIPTION : This procedure  computes            *)
(*        a Lagrangian dual.                            *)
(***********************************************************)

PROCEDURE SOLVE_LAGRANGIAN_DUAL;

VAR
  p: PAIR;
  t: CNFG;
  r: INTEGER;
  sum,min: REAL;

BEGIN

  dobj:=0;
  FOR p:=1 to np do
    BEGIN
      min:=VLARGE;
      FOR r:=(p-1)*nc to p*nc-1 do
        BEGIN
          sum:=ALPHA[ROUTE[r,1]]+ALPHA[ROUTE[r,2]]
                    +ALPHA[ROUTE[r,3]]+ALPHA[ROUTE[r,4]];
          IF sum < min THEN
            BEGIN
              min:=sum;
              dx[p]:=r;
            END;
        END;
      dobj:=dobj+TRAFFIC[p]*min;
    END;
  FOR p:=1 to np do
    BEGIN
      min:=0;
      dz[p]:=0;
      FOR t:=1 to nt do
        BEGIN
          IF DISTANCE[p]*RATE[t]-ALPHA[p]*CAPA[t] < min THEN
            BEGIN
              min:=DISTANCE[p]*RATE[t]-ALPHA[p]*CAPA[t];
              dz[p]:=t;
            END;
        END;
```

```
            dobj:=dobj+min;
        END;
END;


{***********************************************************}
{*     TITLE      : Make feasible solution               *}
{*     AUTHOR     : Hwang, Yong Goo                       *}
{*     DATE       : 15 Sep. 1990                          *}
{*     REVISE     : 15 Nov. 1990                          *}
{*     DESCRIPTION : This procedure  will give us         *}
{*          a feasible solution using infeasible sol-     *}
{*          ution. If this solution is lower than         *}
{*          upper bound, then it replaces upper bound.    *}
{***********************************************************}

PROCEDURE PERTURB_FOR_FEASIBLE_SOLUTIONS;

VAR
  p: PAIR;
  t: CNFG;
  obj: REAL;
  flag: BOOLEAN;

BEGIN

  obj:=0;
  FOR p:=1 to np do
    BEGIN
      flag:=false;
      t:=0;
      IF SUBGRAD[p]+CAPA[dz[p]] > 0 THEN
        BEGIN
          REPEAT
            IF t+1 <= nt THEN
              t:=t+1
            ELSE
              flag:=true;
          UNTIL flag or (SUBGRAD[p]+CAPA[dz[p]] <= CAPA[t]);
        END;
      IF flag THEN
        obj:=VLARGE
      ELSE
        BEGIN
          dz[p]:=t;
```

```
          obj:=obj+DISTANCE[p]*RATE[t];
        END;
    END;
  IF obj < UPPBND THEN
    BEGIN
      UPPBND:=obj;
      pz:=dz;
      px:=dx;
    END;

END;



(***************************************************)
(*    TITLE     : Test subgradient parameters      *)
(*    AUTHOR    : Hwang, Yong Goo                   *)
(*    DATE      : 15 Sep. 1990                      *)
(*    REVISE    : 18 Nov. 1990                      *)
(*    DESCRIPTION : This procedure checks that      *)
(*       terminated condition is satisfied.         *)
(***************************************************)

PROCEDURE CHECK_IMPRO_COUNTER;

BEGIN

  IF IMPCTR > IMPLIM THEN
    BEGIN
      LAMBDA:=LAMBDA/2.0;
      IMPCTR:=1;
    END;

END;

(***********************)
PROCEDURE TEST_TERMINATION;

BEGIN

  IF (LOWBND > 0) and (UPPBND-LOWBND > 0)
              and ((UPPBND-LOWBND)/LOWBND < ERRORLIM) THEN
    TERMINATED:=true
  ELSE
    IF LAMBDA < LAMBDALIM THEN
```

51

```pascal
          TERMINATED:=true
      ELSE
        IF  ITRCTR >=  ITRLIM  THEN
          TERMINATED:=true
        ELSE
          TERMINATED:=false;

END;



{**************************************************}
{*    TITLE     : Update Lagrange multipliers           *}
{*    AUTHOR    : Hwang, Yong Goo                       *}
{*    DATE      : 12 Aug. 1990                          *}
{*    REVISE    : 13 Nov. 1990                          *}
{*    DESCRIPTION : This procedure change Lagrange      *}
{*        multipliers.                                  *}
{**************************************************}

PROCEDURE UPDATE_LAGRANGE_MULTIPLIERS;

VAR
  p: PAIR;

BEGIN

  STEPSIZE:=LAMBDA*(UPPBND-dobj)/NORM;
  FOR p:=1 to np do
    BEGIN
      ALPHA[p]:=ALPHA[p]+STEPSIZE*SUBGRAD[p];
      IF ALPHA[p] < 1e-4 THEN
                  ALPHA[p]:=0;
      IF ALPHA[p] >= VLARGE THEN
        BEGIN
          WRITELN('WARNING: MULTIPLIER TOO LARGE');
          WRITELN(dobj:10:2,NORM:10:0,STEPSIZE:10:2,SUBGRAD[p]:10:2);
          READLN;
        END;
    END;

END;
```

```
(*************************************************)
(*    TITLE      : Solve problem                 *)
(*    AUTHOR     : Hwang, Yong Goo               *)
(*    DATE       : 7  Oct. 1990                  *)
(*    REVISE     : 16 Nov. 1990                  *)
(*    DESCRIPTION : This procedure make a best   *)
(*         feasible solution using above all subproc- *)
(*         edure.                                *)
(*************************************************)

PROCEDURE SOLVE_PROBLEM;

BEGIN

  REPEAT;

    ITRCTR:=ITRCTR+1;
    IMPCTR:=IMPCTR+1;

    CHECK_IMPRO_COUNTER;

    SOLVE_LAGRANGIAN_DUAL;

    WRITELN(ITRCTR:4,'  ',dobj:10:2,'  ',UPPBND:10:2,'  ',LAMBDA:10:4);

    IF dobj > LOWBND THEN
      BEGIN
        IMPCTR:=0;
        LOWBND:=dobj;
        BEST_ALPHA:=ALPHA;
      END;

    SUBGRAD[0]:=0;
    FOR p:=1 to np do
        SUBGRAD[p]:=CAPA[dz[p]]*(-1);
    FOR p:=1 to np do
      BEGIN
        SUBGRAD[ROUTE[dx[p],1]]:=SUBGRAD[ROUTE[dx[p],1]]+TRAFFIC[p];
        SUBGRAD[ROUTE[dx[p],2]]:=SUBGRAD[ROUTE[dx[p],2]]+TRAFFIC[p];
        SUBGRAD[ROUTE[dx[p],3]]:=SUBGRAD[ROUTE[dx[p],3]]+TRAFFIC[p];
        SUBGRAD[ROUTE[dx[p],4]]:=SUBGRAD[ROUTE[dx[p],4]]+TRAFFIC[p];
      END;
    FOR p:=1 to np do
      IF abs(SUBGRAD[p]) < 1e-4 THEN
                            SUBGRAD[p]:=0;
```

53

```
    NORM:=0;
    FOR p:=1 to np do
      NORM:=NORM+SUBGRAD[p]*SUBGRAD[p];

    OPTIMAL:=true;
    FOR p:=1 to np do
      IF SUBGRAD[p] > 0 THEN
                      OPTIMAL:=false;
    IF OPTIMAL THEN
      BEGIN
        UPPBND:=dobj;
        px:=dx;
        pz:=dz;
      END
    ELSE
      PERTURB_FOR_FEASIBLE_SOLUTIONS;

    TEST_TERMINATION;

    IF not TERMINATED THEN
          UPDATE_LAGRANGE_MULTIPLIERS;

  UNTIL TERMINATED;

END;
```

D. REPORT PROCEDURE

```
(********************************************************)
(*    TITLE       : Make report                       *)
(*    AUTHOR      : Hwang, Yong Goo                    *)
(*    DATE        : 3  Oct. 1990                       *)
(*    REVISE      : 21 Nov. 1990                       *)
(*    DESCRIPTION : This procedure provides a report.  *)
(********************************************************)

PROCEDURE DRAW_NETWORK;

CONST
  MAXLENGTH = 250;
  MAXHEIGHT = 150;

VAR
  CENTER_X,
  CENTER_Y : INTEGER;
  i, j : INTEGER;
  TEMP_FT_1, TEMP_T_1, TEMP_T_3 : INTEGER;
  inkey : INTEGER;
  ckey : char;
  DRIVERVAR, MODEVAR : INTEGER;

  (************************************)
  PROCEDURE GRAPHTITLE (INTITLE : STRING);

    BEGIN

      SETTEXTJUSTIFY(CENTERTEXT, TOPTEXT);
      SETTEXTSTYLE(TRIPLEXFONT, HORIZDIR,4);
      OUTTEXTXY(GETMAXX DIV 2, 1, INTITLE);

    END;  (*  END of graphtitle  *)


  (*********************)
  PROCEDURE GRAPHCONTINUE;

  CONST
    message1 = 'CONTINUE? (y/n)';
```

```
    MESSAGE2 = 'Enter the PAIR number!';

BEGIN

  SETTEXTJUSTIFY(CENTERTEXT, TOPTEXT);
  SETTEXTSTYLE(GOTHICFONT, HORIZDIR,3);
  OUTTEXTXY(GETMAXX DIV 2, GETMAXY - 40, MESSAGE1);
  READLN (ckey);
  IF (ckey <> 'N') AND (ckey <> 'n') THEN
    BEGIN
      SETTEXTJUSTIFY(CENTERTEXT, TOPTEXT);
      SETTEXTSTYLE(GOTHICFONT, HORIZDIR,3);
      OUTTEXTXY(GETMAXX DIV 2, GETMAXY - 50, MESSAGE2);
      READLN(inkey);
    END;
  CLEARDEVICE;

END;    (*  END of graphcontinue  *)


(*****************)
PROCEDURE DRAW_ROUTE;

VAR
  i,j : INTEGER;

BEGIN

  LINE(10,10,10,450);
  LINE(10,450,getmaxx,450);
  LINE(center_x,450,center_x,10);
  LINE(getmaxx,10,10,10);
  LINE(getmaxx,getmaxy,getmaxx,10);
  FOR i := 1 TO ni DO
       CIRCLE((POINT_X[i] DIV 3),(POINT_Y[i] DIV 3), 2);
  FOR j := 1 TO np DO
    IF (pz[j] <> 0) THEN
      BEGIN
        (* TEMP_FT_1 := T_TABLE[j].FT_1;
        TEMP_T_1  := T_TABLE[j].T_1;
        TEMP_T_3  := T_TABLE[j].T_3; *)
        LINE(POINT_X[S_D_PAIR[j].S] DIV 3,
             POINT_Y[S_D_PAIR[J].S] DIV 3,
             POINT_X[S_D_PAIR[j].D] DIV 3,
             POINT_Y[S_D_PAIR[J].D] DIV 3);
```

56

```
            END;
      LINE(POINT_X[s_d_PAIR[ROUTE[px[inkey],1]].s] div 4 + center_x,
            POINT_Y[s_d_PAIR[ROUTE[px[inkey],1]].s] div 4,
            POINT_X[s_d_PAIR[ROUTE[px[inkey],1]].d] div 4 + center_x,
            POINT_Y[s_d_PAIR[ROUTE[px[inkey],1]].d] div 4);
      IF ROUTE[px[inkey],2] > 0 THEN
         LINE(POINT_X[s_d_PAIR[ROUTE[px[inkey],2]].s] div 4 + center_x,
              POINT_Y[s_d_PAIR[ROUTE[px[inkey],2]].s] div 4,
              POINT_X[s_d_PAIR[ROUTE[px[inkey],2]].d] div 4 + center_x,
              POINT_Y[s_d_PAIR[ROUTE[px[inkey],2]].d] div 4);
      IF ROUTE[px[inkey],3] > 0 THEN
         LINE(POINT_X[s_d_PAIR[ROUTE[px[inkey],3]].s] div 4 + center_x,
              POINT_Y[s_d_PAIR[ROUTE[px[inkey],3]].s] div 4,
              POINT_X[s_d_PAIR[ROUTE[px[inkey],3]].d] div 4 + center_x,
              POINT_Y[s_d_PAIR[ROUTE[px[inkey],3]].d] div 4);
      IF ROUTE[px[inkey],4] > 0 THEN
         LINE(POINT_X[s_d_PAIR[ROUTE[px[inkey],4]].s] div 4 + center_x,
              POINT_Y[s_d_PAIR[ROUTE[px[inkey],4]].s] div 4,
              POINT_X[s_d_PAIR[ROUTE[px[inkey],4]].d] div 4 + center_x,
              POINT_Y[s_d_PAIR[ROUTE[px[inkey],4]].d] div 4);

   END;


BEGIN   (*   MAIN  PART  *)

   inkey :=0;
   DRIVERVAR := detect;
   INITGRAPH(DRIVERVAR, MODEVAR, '');
   CENTER_X := GETMAXX DIV 2;
   CENTER_Y := GETMAXY DIV 2;
   GRAPHTITLE('NETWORK DESIGN');
   LINE(10,10,10,450);
   LINE(10,450,550,450);
   LINE(550,450,550,10);
   LINE(550,10,10,10);
   FOR i := 1 TO ni DO
        CIRCLE((POINT_X[i] DIV 2),(POINT_Y[i] DIV 2), 2);
   FOR j := 1 TO np DO
     BEGIN
       IF (pz[j] <> 0) THEN
         BEGIN
          (* TEMP_FT_1 := T_TABLE[j].FT_1;
           TEMP_T_1  := T_TABLE[j].T_1;
           TEMP_T_3  := T_TABLE[j].T_3; *)
```

```
                LINE(POINT_X[S_D_PAIR[j].S] DIV 2,
                     POINT_Y[S_D_PAIR[J].S] DIV 2,
                     POINT_X[S_D_PAIR[j].D] DIV 2,
                     POINT_Y[S_D_PAIR[J].D] DIV 2);
          END;
     END;
  REPEAT
    graphcontinue;
    IF (inkey <> 0) THEN
            DRAW_ROUTE;
  UNTIL (ckey = 'N') or (ckey = 'n');
  CLOSEGRAPH;

END;


(**************************************)
PROCEDURE REPORT_BEST_FEASIBLE_SOLUTION;

VAR
  p: PAIR;
  df: text;
  i: INTEGER;

BEGIN

  ASSIGN(df,FILE_NAME);
  REWRITE(df);
  WRITELN(df);
  WRITELN (df,'relative error bound: ',
              ((UPPBND-LOWBND)/UPPBND)*100:7:4,' %');
  WRITELN (df,'LAMBDA: ',LAMBDA);
  WRITELN(df);
  WRITELN(df,'The best feasible solution:  ',UPPBND:10:2);
  FOR p:=1 to np do
    BEGIN
      WRITE(df,p:2,' [',ORGN[p]:2,'-',DSTN[p]:2,'] : '
                                     ,ROUTE[px[p],1]:4);
      IF ROUTE[px[p],2] > 0 THEN
              WRITE(df,ROUTE[px[p],2]:4);
      IF ROUTE[px[p],3] > 0 THEN
              WRITE(df,ROUTE[px[p],3]:4);
      IF ROUTE[px[p],4] > 0 THEN
              WRITE(df,ROUTE[px[p],4]:4);
      WRITELN(df);
```

58

```
      END;
    i:=0;
    FOR p:=1 to np do
      BEGIN
        IF i < 4 THEN
          BEGIN
            WRITE(df,'[',p:3,'] ',pz[p]:4);
            i:=i+1;
          END
        ELSE
          BEGIN
            WRITE(df,'[',p:3,'] ',pz[p]:4);
            WRITELN(df);
            i:=0;
          END;
      END;
    WRITELN(df);
    close(df);
    DRAW_NETWORK;

END;



(*********************************)
(*   main body                  *)
(*********************************)

BEGIN

  FRAG1 := TRUE;

  GENERATE_DATA;

  REPEAT

    GET_DATA;

    FIND_INITIAL_FEASIBLE_SOLUTION;

    PREP_FOR_SUBGRADIENT_ITERATION;

    SOLVE_PROBLEM;

    REPORT_BEST_FEASIBLE_SOLUTION;
```

```
      WRITELN(' Try again? : (y/n) ');
      READLN(ANS1);

   UNTIL (ANS1 = 'N') OR (ANS1 = 'n');


END.   (*** main PROCEDURE ***)
(**********************************************)
```

APPENDIX B: SAMPLE INPUT DATA FILE

A. PAIR INFORMATION

pairs sour.	dest.	distance	traffic
1	2	395	7
1	3	339	98
1	4	156	0
1	5	261	64
1	6	724	43
1	7	516	23
2	3	556	0
2	4	448	38
2	5	592	27
2	6	674	0
2	7	502	93
3	4	491	62
3	5	172	90
3	6	469	28
3	7	295	34
4	-	385	53
4	6	877	0
4	7	668	87
5	6	640	23
5	7	459	0
6	7	209	6

61

B. CIRCUIT INFORMATION

CIRCUIT	
capa.	cost
6	10
12	15
18	19
24	21
30	31
36	36
42	40
48	42
54	52
60	57
66	61
72	63
78	73
84	78
90	82
96	84
102	94
108	99
114	103
120	105
126	115
132	120
138	124
144	126
150	136
156	141
162	145
168	147
174	157
180	162
186	166
192	168
198	178
204	183
210	187
216	189
222	199
228	204
234	208
672	219
678	229
684	234

690	238
696	240
702	250
708	255
714	259
720	261
726	271
732	276
738	280
744	282
750	292
756	297
762	301
768	303
774	313
780	318
786	322
792	324
798	334
804	339
810	343
816	345
822	355
828	360
834	364
840	366
846	376
852	381
858	385
864	387
870	397
876	402
882	406
888	408
894	418
900	423
906	427
1344	438
1350	448
1356	453
1362	457
1368	459
1374	469
1380	474
1386	478
1392	480
1398	490

1404	495
1410	499
1416	501
1422	511
1428	516
1434	520
1440	522
1446	532
1452	537
1458	541
1464	543
1470	553
1476	558
1482	562
1488	564
1494	574
1500	579
1506	583
1512	585
1518	595
1524	600
1530	604
1536	606
1542	616
1548	621
1554	625
1560	627
1566	637
1572	642
1578	646

APPENDIX C: SAMPLE OUTPUT DATA FILE

relative error bound: 63.5411 %
LAMBDA: 2.5000000000E-01

The best feasible solution: 274700.00
```
 1 [ 1- 2] :     1
 2 [ 1- 3] :     4   13
 3 [ 1- 4] :     3
 4 [ 1- 5] :     4
 5 [ 1- 6] :     4   13   14
 6 [ 1- 7] :     4   13   15
 7 [ 2- 3] :     7
 8 [ 2- 4] :     1    3
 9 [ 2- 5] :     1    4
10 [ 2- 6] :    10
11 [ 2- 7] :    11
12 [ 3- 4] :    13    4    3
13 [ 3- 5] :    13
14 [ 3- 6] :    14
15 [ 3- 7] :    15
16 [ 4- 5] :     3    4
17 [ 4- 6] :    17
18 [ 4- 7] :     3    4   13   15
19 [ 5- 6] :    13   14
20 [ 5- 7] :    13   15
21 [ 6- 7] :    21
[  1]    12[  2]     0[  3]    40[  4]    40[  5]     0
[  6]     0[  7]     0[  8]     0[  9]     0[ 10]     0
[ 11]    16[ 12]     0[ 13]    40[ 14]    16[ 15]    24
[ 16]     0[ 17]     0[ 18]     0[ 19]     0[ 20]     0
[ 21]     1
```

LIST OF REFERENCES

1. P. Afentakis and B. Gavish, "Optimal lot-sizing algorithms for complex product structures", Grad. School Management, University Rochester, Rochester, NY, 1983.

2. D.P. Bertsekas, "A class of optimal routing algorithms for communications networks", in Proc. 1980 Int. Conf. Circuits Comput., Atlanta, GA, Nov. 1980

3. M.L. Fisher, "Lagrangean relaxation method for solving integer programming problems", Management Science, vol. 27, pp. 1-18, Jan. 1981.

4. M. Held, P. Wolfe, and H. Crowder, "Variation of subgradient optimization", Math. Program., vol. 6, pp. 62-88, 1974.

5. B. Gavish and S.L. Hantler, "An algorithm for optimal route selection in SNA networks", IEEE Trans. on Comm., vol.com-31, pp. 1154-1160, Oct. 1983.

6. M. Karnaugh, "A New Class of Algorithms for Multipoint Network Optimization", IEEE Transactions on Communications, vol. COM-24, pp. 500-505, May 1976.

7. L.G. Mitten, "Branch-and bound methods: General formulation and properties", Operations Research, vol. 18, pp 24-34, 1970.

8. E. Rosenberg, "A Nonlinear Programming Heuristic for Computing Optimal Link Capacities in a Multi-hour Alternate Routing Communications network", Operations Research, vol. 35, No. 3, pp. 354-364, May-June 1987.

9. D.N. Lee, K.T. Medhi and J.L. Strand, "Solving Large Telecommunications Network Loading Problems", AT T Technical Journal, pp. 48-56, May-June 1989.

10. W.J. Barksdale, Practical Computer Data Communications, Plenum Press, New York, 1986.

11. W.A. Flanagan, The Guide to T-1 Networking, Telecom Library Inc., 1988.

12. R. Sharma, "T-1 Network Design and Planning Made Easier", Data Communications, pp. 199-207, Sept. 1988.

13. S. Fleming, "Get Ready for T-3 Networking", Data Communications, pp. 82-94, Sept. 1989.

14. Udi Manber, Introduction to Algorithms, Addison-Wesley Publishing Co., July 1988.

15. David R. Anderson, Dennis J. Sweeney and Thomas A. Williams, An Introduction to Management Science, West Publishing Co., 1988.

16. Andrew S. Tanenbaum, Computer Networks, Prentice-Hall Inc., 1988.

17. Uyless Black, Data Networks, Prentice-Hall Inc., 1989.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2. Library, Code 52 Naval Postgraduate School Monterey, California 93943	2
3. Department Chairman, Code CS Department of Computer Science Naval Postgraduate School Monterey, California 93943	1
4. Prof. Myung W. Suh, Code ASSU Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943	1
5. Prof. Richard W. Hamming, Code CSHG Department of Computer Science Naval Postgraduate School Monterey, California 93943	1
6. Prof. Mantak Shing, Code CSSH Department of Computer Science Naval Postgraduate School Monterey, California 93943	1
7. Prof. Dong S. Kim, Code MEKM Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943	1
8. Prof. Kyung C. Kim, Code Department of Computer Science Naval Postgraduate School Monterey, California 93943	1

9. Army Central Library   1
   Army Headquarter, Bunam-Ri,
   Duma-Myun, Nonsan-Gun, Chungcheong-Nam-Do,
   Republic of Korea, 320-919

10. Hwang; Yong Goo   11
    734-3, HwangGeum-Dong, SuSeong-Gu, Teagu,
    Seoul, Republic of Korea, 706-040

11. Song, Il Youn   1
    SMC 2302 NPGS
    Monterey, California 93943

12. Chung, Jae Du   1
    SMC 1504 NPGS
    Monterey, California 93943

13. Park, Hyun Kyoo   1
    SMC 2820 NPGS
    Monterey, California 93943

14. Choi, Nag Jung   1
    SMC 2853 NPGS
    Monterey, California 93943