

# **Network Optimisation Package for the Design and Analysis of Survivable SDH Networks**

James O'Hara B. Eng.

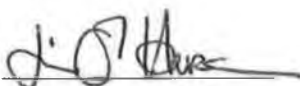
A thesis submitted as a requirement for the degree  
of Masters of Engineering in Electronic Engineering

Dublin City University  
Supervisor - Dr. T. Curran

September 1998

## Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Masters of Engineering, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of this work.

Signed:   
James O'Hara

ID Number: 93701365

Date: 11<sup>th</sup> September, 1998

## Acknowledgements

The author would like to express his gratitude and thanks to the following people who contributed to and made this work possible:

- To Dr. Tommy Curran for providing the opportunity to pursue this research, and for his technical advice and expertise throughout.
- To Broadcom Éireann Research Ltd., for its assistance and continued support.
- To my colleagues, both at DCU and Broadcom, for listening to my concepts and for their enthusiasm for offering advice. In particular, thanks is due to Mandy, for her - as she put it - "*amazing patience, incredible eye for detail, limitless generosity, ...*"
- To my parents for their constant support throughout my studies.
- And finally, to Fiona, for being there when the going got tough, and for her encouragement and unflagging desire to see this labour of love concluded.

# Abstract

---

**Title:** Network Optimisation Package for the Design and Analysis of Survivable SDH Networks

**Author:** James O'Hara B. Eng.

---

*The advancements in digital technology and the resultant economies of scale offered by high capacity fibre transmission systems have enabled network operators to provide more cost-effective telecommunications services to their increasing customer base. However, with the deployment of such systems, the emphasis on service quality and network reliability has increased considerably. To address this and many other network management issues, the SDH digital transmission standard was established. While supporting enhanced network management functionality, it was only through the subsequent development of highly intelligent Network Elements (NEs) that the inherent cost benefits of SDH have been realised. Each NE is characterised by its own topology and associated restoration schemes. As the economic viability of these survivable topologies are dependent on varying network conditions, it has been shown that the most cost-effective networks will be based on a combination of these topologies. The complexity of these highly integrated networks and the immense capital investment involved suggests that much consideration should be given to the design and optimisation of such networks. This research work concerns an investigation into this particular area of network optimisation. It involves the development of an optimisation design package that facilitates the generation and analysis of survivable SDH networks through the implementation of highly efficient optimisation techniques. Existing optimisation techniques are utilised to this end, and enhanced upon to address aspects of the SDH design model that were previously not taken into consideration. This results in the development of innovative and highly efficient algorithmic procedures for solving this particular design problem, and "NetOpt" - a prototype design tool for implementing such procedures. Based on a set of generated test case networks, a detailed analysis and evaluation of these procedures and the resultant network solutions is presented. In conclusion, it is shown that CAD-based tools are essential for the generation and analysis of survivable SDH networks.*

---

# Table of Contents

---

<b>Chapter 1 - Introduction .....</b>	<b>1</b>
1.1 Motivation .....	1
1.2 Objectives .....	2
1.3 Thesis Structure .....	4
1.4 Associated Publications .....	4
<b>Chapter 2 - SDH Network Design Concepts.....</b>	<b>5</b>
2.1 Emergence of SDH.....	5
2.1.1 SDH Standardisation Process .....	6
2.1.2 SDH Frame Structure.....	8
2.1.3 Layer Concept and SDH Overhead Channels.....	9
2.1.4 STM-1 Payload .....	10
2.1.5 Benefits of SDH.....	12
2.2 Review of SDH-Based Network Elements.....	13
2.2.1 Terminal Multiplexer (TM) .....	13
2.2.2 Add-Drop Multiplexer (ADM) .....	14
2.2.3 Digital Cross-Connect (DXC) .....	15
2.3 SDH Survivable Topologies.....	16
2.3.1 TM Point-to-Point Topologies.....	16
2.3.2 ADM Ring Topologies .....	17
2.3.2.1 Line-level restoration .....	18
2.3.2.2 Path-level restoration .....	19
2.3.3 DXC-based Mesh Topologies.....	20
2.3.4 Hybrid ADM-ring/DXC Topology .....	22
2.3.5 Comparison of Survivable Topologies .....	22
2.4 SDH Network Architectures and Deployment Strategies .....	23
2.4.1 SDH Network Architecture .....	23
2.4.1.1 National trunk network .....	24
2.4.1.2 Regional junction network .....	24
2.4.1.3 Local access network .....	24
2.4.2 SDH Deployment Strategies .....	25
2.4.3 Considerations Affecting SDH Deployment .....	26

<b>Chapter 3 - SDH Network Optimisation.....</b>	<b>27</b>
3.1 Problem Formulation and Network Representation .....	27
3.1.1 Input data and network representation .....	27
3.1.2 Output data and solution representation .....	28
3.1.3 Design constraints.....	29
3.1.4 Formulation of Cost Function.....	30
3.1.4.1 Nodal cost function .....	30
3.1.4.2 Link cost function .....	31
3.1.5 Reformulation as a Combinatorial Optimisation Problem .....	34
3.2 Joint Optimal Routing and Working Capacity Assignment Problem.....	35
3.2.1 Problem Formulation .....	35
3.2.2 Adoption of a Minimum-Hop Routing Constraint .....	35
3.2.3 Routing Algorithm and Computational Complexity .....	40
3.3 Optimal Spare Capacity Assignment Problem .....	40
3.3.1 Problem Formulation .....	40
3.3.2 Generation of Optimal Flow Constraints.....	41
3.3.3 Optimisation Phase and Problem Complexity .....	46
3.3.4 Proposed Solution Method .....	47
3.3.4.1 Initial spare capacity assignment algorithm.....	47
3.3.4.2 Global spare capacity assignment algorithm .....	50
3.3.4.3 Adaptive minimisation algorithm .....	51
3.3.5 Spare Capacity Assignment based on Path-Level Restoration .....	52
3.3.5.1 Initial spare capacity assignment algorithm.....	53
3.3.5.2 Global spare capacity assignment algorithm .....	54
3.3.5.3 Overall complexity.....	55
3.4 Network Optimisation Solution Method .....	56
3.4.1 Review of Solution Methods .....	56
3.4.1.1 Local search methodology .....	56
3.4.1.2 Greedy algorithms.....	57
3.4.1.3 Modern methods .....	57
3.4.1.4 Comparison of solution methods .....	58
3.4.2 Proposed Solution Method .....	60
3.4.2.1 Initial optimisation phase.....	60
3.4.2.2 Refinement phase.....	62
<b>Chapter 4 - Implementation of Design Procedure.....</b>	<b>63</b>
4.1 Optimal Design Procedure .....	63
4.2 Network Optimisation Algorithm.....	64

4.2.1 Initial Optimisation Algorithm .....	65
4.2.2 Dual Optimisation Algorithm.....	67
4.2.3 Network Cost Algorithms.....	70
4.2.3.1 Basic network cost algorithm.....	70
4.2.3.2 Total network cost algorithm.....	71
4.3 Joint Optimal Routing & Working Capacity Assignment Algorithm .....	71
4.3.1 Optimal Routing Algorithm.....	72
4.3.1.1 Dijkstra's shortest-path algorithm .....	72
4.3.1.2 Improvements to existing algorithm .....	74
4.3.1.3 Imposing minimum-hop routing constraint.....	75
4.3.1.4 Overall routing algorithm.....	76
4.3.2 Interpretation of Paths Matrix and Path-Trace Algorithm.....	77
4.3.3 Connectivity Test Algorithm .....	79
4.3.4 Working Capacity Assignment Algorithm .....	80
4.4 Optimal Spare Capacity Assignment Algorithm .....	80
4.4.1 Initial Spare Capacity Assignment Algorithm.....	81
4.4.2 Global Spare Capacity Assignment Algorithm.....	83
4.4.3 Adaptive Minimisation Algorithm .....	84
4.4.4 Release Phase Algorithm.....	85

## **Chapter 5 - NetOpt Design Tool .....86**

5.1 Overview of NetOpt .....	86
5.1.1 Active Design Field .....	86
5.1.1.1 Network representation.....	87
5.1.1.2 Zoom setting.....	88
5.1.1.3 Gridlines.....	88
5.1.1.4 Network coordinates .....	88
5.1.1.5 Scroll bars .....	89
5.1.1.6 Network rendering .....	90
5.1.1.7 Changing the zoom setting.....	90
5.1.2 Menu Bar .....	91
5.1.2.1 File menu.....	91
5.1.2.2 Edit menu .....	92
5.1.2.3 View menu .....	93
5.1.2.4 Optimisation menu .....	94
5.1.3 Tool Bar .....	95
5.1.4 Status Bar.....	95
5.1.5 Dialog Boxes.....	96

5.1.6	Interaction with the Network Solution.....	96
5.1.6.1	Link options .....	96
5.1.6.2	Node options .....	96
5.2	Initialising a New Input Network .....	97
5.2.1	Initialise the Active Network Domain .....	97
5.2.2	Initialise the Number of Nodes .....	97
5.2.3	Initialise the Nodal Characteristics .....	98
5.2.4	Initialise the Inter-Nodal Constraints.....	98
5.2.5	Initialise the Cost Components .....	99
5.2.6	Initialise the Network Description .....	100
5.2.7	Save the Network to File .....	100
5.3	Network Editing Utilities .....	100
5.3.1	Editing Inter-Nodal Constraints (Locally) .....	100
5.3.2	Editing Inter-Nodal Constraints (Globally) .....	102
5.3.3	Updating the Network Solution .....	102
5.4	Network Optimisation Utilities .....	102
5.4.1	<u>D</u> esign Constraints.....	102
5.4.2	<u>C</u> ost Components.....	103
5.4.3	<u>N</u> etwork Optimisation .....	103
5.4.4	<u>I</u> nitial Optimisation.....	104
5.4.5	<u>D</u> ual Optimisation.....	104
5.4.6	<u>O</u> ptimal Spare Capacity .....	104
<b>Chapter 6 - Results and Analysis.....</b>		<b>105</b>
6.1	Generation of Test Case Networks.....	105
6.2	Solution Characteristics.....	107
6.2.1	Network Connectivity.....	107
6.2.2	Working Capacity Solution Characteristics.....	108
6.2.3	Working and Spare Capacity Cost Characteristics .....	115
6.2.4	Validation of Initial Optimisation Phase .....	120
6.2.5	Optimal Solution Characteristics .....	121
6.3	Optimisation Complexity and Efficiency .....	123
6.3.1	Initial Optimisation Phase.....	124
6.3.2	Refinement Phase .....	124
6.4	Optimal Routing Strategies .....	127
6.4.1	Working Capacity Cost Characteristics.....	127
6.4.2	Spare Capacity Cost Characteristics .....	128



6.5 Spare Capacity Assignment Algorithm .....	129
6.5.1 Quality of Near-Optimal Solution .....	129
6.5.2 Path-level and Line-level Restoration.....	130
<b>Chapter 7 - Conclusions.....</b>	<b>132</b>
7.1 Overall Discussion of Results .....	132
7.1.1 Solution Characteristics .....	132
7.1.2 Optimal Routing Strategies.....	135
7.1.3 Optimal Spare Capacity Assignment Algorithm .....	135
7.1.4 Overall Optimisation Procedure .....	136
7.1.4.1 Initial optimisation phase.....	136
7.1.4.2 Refinement phase.....	136
7.1.5 NetOpt and the Optimal Design Procedure .....	138
7.2 Recommendations and Scope for Future Work.....	139
7.2.1 Further Validation of Results.....	139
7.2.2 Further Development of the Optimal Design Procedure .....	139
<b>References .....</b>	<b>140</b>
<b>Appendix A1 - DXC-based Restoration Control .....</b>	<b>I</b>
<b>Appendix A2 - Distributed Restoration Time .....</b>	<b>III</b>
<b>Appendix A3 - Optimal Routing Algorithms .....</b>	<b>VII</b>
A3.1 Implementation of Dijkstra's Shortest-Path Algorithm .....	VII
A3.2 Implementation of Modified Shortest-Path Algorithm .....	IX
A3.3 Implementation of Minimum-hop/Shortest-path Algorithm .....	X

---

# Chapter 1 - Introduction

---

## 1.1 Motivation

Synchronous Digital Hierarchy (SDH) has evolved to address the limitations of the existing Plesiochronous Digital Hierarchy (PDH) networks. It has surpassed PDH in the following aspects:

- defines a set of high-capacity hierarchical signal rates, ranging from 155 Mb/s to 2.4 Gb/s, with easy accommodation of higher order rates, such as 10 Gb/s, as technology evolves.
- represents a worldwide standard which unifies European, North American, and Japanese transmission rates, and supports interconnection of existing PDH networks.
- provides advanced network management capabilities allowing network operators to provide enhanced and highly reliable services through pre-emptive maintenance and dynamic service restoration.
- supports a simplified method of synchronous multiplexing which enables signals to be extracted/inserted directly into higher order signal rates, overcoming the inefficiency of the multi-step PDH-based process.
- precipitated the development and standardisation of advanced and highly intelligent Network Elements (NEs), i.e. Terminal Multiplexers (TMs), Add-Drop Multiplexers (ADMs), and Digital Cross-Connects (DXCs). Associated with each NE are characteristic topologies and restoration schemes, i.e. TM-based point-to-point, ADM-based ring, and DXC-based mesh topologies. These NEs and their topologies provide dynamic routing and restoration capabilities needed for distributed network management control.
- provides improved network performance and flexibility enabling network operators to meet customer's growing demands for better quality of service. The initial costs of such a new network infrastructure are offset by the consequent reduction in operating costs and return in revenue that result from the highly reliable and cost effective services that can be provided.
- offers easy accommodation of new broadband services, such as B-ISDN and ATM.

With the deployment of these high capacity fibre transmission systems, the emphasis on service quality and network reliability has increased substantially. For network operators, improved

network reliability is essential from the perspective of lost revenue and the fact that their customers, particularly business users, are now themselves demanding a higher level of service quality.

Addressing this issue, SDH has resulted in the development of enhanced self-healing techniques which are supported by the various NEs and their characteristic topologies and restoration schemes. These topologies are each dependent to a varying degree on restoration time, relative cost, network characteristics, and the required level of survivability. It has been shown that survivable SDH transmission networks are expected to be based on a combination of these topologies. Due to the difficulty of planning such complex SDH integrated restoration schemes, computer-aided design tools are essential to ensure reasonable and cost-effective network solutions that utilise the merits of each restoration topology.

## 1.2 Objectives

The overall objective of this work was to develop a network optimisation package that would facilitate the design and analysis of optimal SDH network solutions through the implementation of highly efficient and effective optimisation techniques. The realisation of this objective involved the following procedural steps:

Step 1. The formulation of the optimal design problem in terms of the:

- notation and data structures used to represent both the input network and the resultant network solution
- design constraints which must be satisfied in order to realise a feasible solution
- cost model adopted for this particular problem

Step 2. The development of an optimal design procedure for solving this problem.

Step 3. The development of a prototype design tool that would facilitate the implementation of this design procedure.

Step 4. The evaluation of SDH network solutions resulting from the application of this procedure.

The inputs to the design problem include node locations, potential links, and inter-nodal demands, fixed costs, and distances. The objective being to generate a network solution that minimises the total network cost while allocating sufficient network capacity to route all inter-nodal demands and satisfy a required level of survivability. This problem is characterised by large dimensionality even where small networks are concerned and it turns out that the overall problem is completely intractable. However, based on the network model adopted, it will be shown that the problem can be

reformulated as a combinatorial optimisation problem. This enables the natural decomposition of the problem into a number of sub-problems that can be solved sequentially.

The basic sub-problem is, given a network configuration, how should capacity be assigned in order to route all inter-nodal demands and satisfy a required level of survivability at the minimum cost. This is composed of two sequential solution steps. The first concerns the optimal routing of demands and the assignment of working capacity based on the optimal routes generated. While well-established shortest-path routing algorithms could be used to realise this solution, it was found that the routing strategy needed to be refined to take into consideration component costs specific to the SDH network model. Further modifications were also made to improve the overall performance of the algorithm. Based on the solution derived from this step, the second step involves the optimal assignment of spare capacity across the network that ensures 100 % restoration against all single link failures. While an exact solution to this problem can be found, the computational complexity of the technique used increases dramatically with the number of nodes in the network, and it turns out that this problem is itself deemed intractable for networks with more than 20 nodes. As a consequence, considerable work was devoted to this problem in an attempt to find a feasible solution method. This has resulted in the development and implementation of an innovative and highly efficient technique which, in its basic form, is capable of finding near-optimal cost solutions that are proportional representations of the optimal cost solutions.

Given that the minimum cost solution for any network configuration can be found, the objective of the overall optimisation problem is to find the configuration that represents the optimal solution. However, as the number of potential solutions grows exponentially with the number of input nodes, this problem is classified as *NP-hard*, for which no realistic time solution can be found. In recent years, a number of heuristic-based solution methods have been developed to effectively solve this class of problem. Based on the concept of a local search methodology, the most efficient of these solution methods are based on well-established *Greedy* algorithms, while, more recently, a number of more advanced algorithms have emerged. In general, while the latter are capable of finding better solutions, a considerable increase in the computational time requirement is incurred. Following a review of these algorithms, it was found that, due to the incorporation of the spare capacity assignment problem, no single algorithm could efficiently and/or effectively solve the overall optimisation problem. As a result, a two-phase solution approach was adopted. The first phase utilises a highly efficient *Greedy* algorithm to reduce the initial complexity of the problem by finding a more practical starting solution, while the second phase concerns the refinement of this solution and is based on a more advanced and computationally intensive algorithm. To accelerate the optimisation process and improve its overall effectiveness, perturbation techniques, such as user

interaction with the refinement phase, are also employed. These form part of the overall optimal design procedure facilitated by the design tool.

### **1.3 Thesis Structure**

An introduction to SDH network design concepts is presented in Chapter 2. Following a discussion on the benefits of SDH and how it will form the infrastructure for future transmission networks, the various SDH-based network elements and their characteristic topologies and restoration schemes are reviewed. Finally, the current trends in SDH deployment are reviewed.

Chapter 3 concerns the formulation of the optimal design problem. Based on the network model defined, it is shown how the problem can be decomposed into the various sub-problems identified in Section 1.2. The solution methods developed to efficiently solve these individual sub-problems are then presented.

Chapter 4 concerns the implementation of these solution methods in terms of the algorithmic procedures involved and how these form part of the overall optimal design procedure facilitated by the design tool.

An overview of the design tool and how it facilitates the generation, editing and optimisation of survivable SDH networks is presented in Chapter 5.

Chapter 6 concerns an evaluation of the optimal design procedure facilitated by the design tool. Based on the analysis of a set of test case networks that model variations in the input constraints, it will be shown how these input conditions impact on the solution characteristics for each network. It will also be shown how these characteristics have a residual effect on the computational complexity of each phase of the respective optimisation processes. Finally, a cost analysis of alternative routing strategies and restoration techniques is presented.

An overall discussion of results and scope for further work is presented in Chapter 7.

### **1.4 Associated Publications**

A summary of part of this work was published in "*Technology Ireland 1995*", a magazine supplement published by *TELTEC Ireland*.

---

## Chapter 2 - SDH Network Design Concepts

---

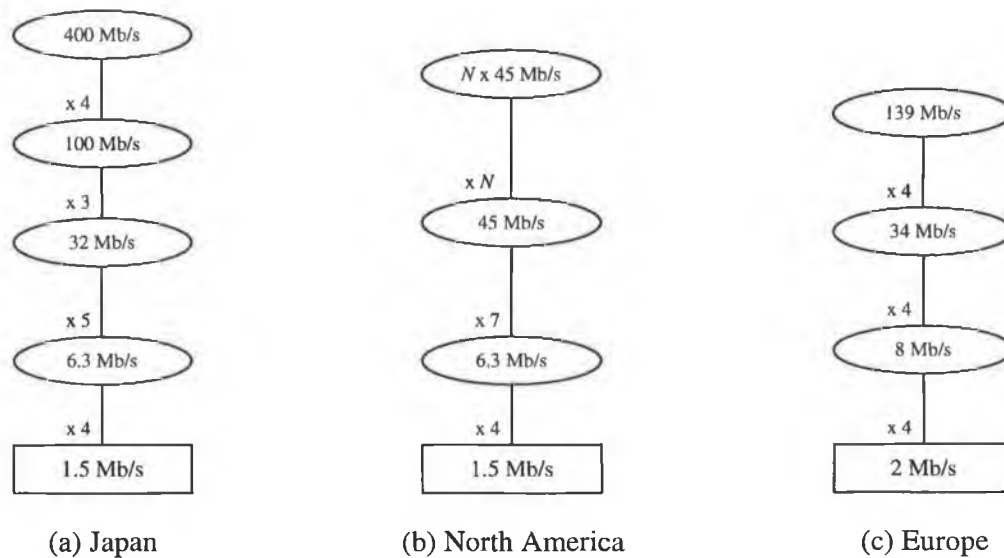
### 2.1 Emergence of SDH

As digital networks increased in size and complexity over the last decade, there was a growing need from both network operators and their customers for facilities not readily or economically supported by the existing Plesiochronous Digital Hierarchy (PDH) transmission networks. PDH was developed in the early 70s to provide increased routing capacity to meet the rapid growth in both residential and business telephony traffic. Although based on digital transmission over coaxial cables, the high cost of bandwidth and digital components at that time put constraints on the operations and management facilities PDH networks could offer. While PDH standards have evolved in a step-by-step basis to support the growing demand in voice traffic and data services, the inherent lack of facilities has left it a costly, outmoded, and difficult-to-manage network, whose inefficiencies, both in equipment and operations, have inhibited the progression toward a more flexibly controlled and fully managed broadband network.

PDH is based on a complex multiplex/demultiplex scheme which enables lower order primary rate signals, i.e. 2 Mb/s, to be multiplexed into successively higher order rates, i.e. 8 Mb/s, 34 Mb/s, 140 Mb/s and, later, up to the proprietary 565 Mb/s line rate. At each multiplexing step, the bit rate of the individual signal rates is controlled within specified limits and synchronised by the process of bit stuff justification. In order to extract/insert an individual 2 Mb/s signal at any level of the hierarchy, i.e. 140 Mb/s, the whole signal structure must be demultiplexed (i.e. from 140 to 34, 34 to 8, and 8 to 2) and then reassembled (i.e. 2 to 8, 8 to 34, and 34 to 140) for onward transmission. This highlights one of the major drawbacks of using PDH in high capacity networks.

Another serious limitation of the PDH systems is the lack of management and maintenance information passed along with the signal. The standard approach to managing these networks is to use manual distributed frames (MDFs) which provide cross-connection of channels using coaxial patching leads. This severely limits the flexibility required by network operators to respond to network failures and the rapidly changing demands of their customers. In addition, within the European PDH system, higher order line rates lack a common standard above 140 Mb/s. While proprietary signal rates do operate above this level, due to their proprietary nature, interworking between systems is impossible. At an international level, incompatibilities between line rates are

even more significant. In Japan and North America, the PDH hierarchies differ considerably from those defined in Europe. A comparison of the various PDH hierarchies is shown in Figure 2.1.



**Figure 2.1** International PDH hierarchies

Recognising these problems, the telecommunications industry has formulated a worldwide digital transmission standard, the Synchronous Digital Hierarchy.

### 2.1.1 SDH Standardisation Process

The initial standardisation process began in 1985, when the American National Standards Institute (ANSI) established the SONET (Synchronous Optical NETwork) standard [1]. This standard defined a set of hierarchical transport structures which utilise pointers to provide a more flexible and simplified technique for the multiplexing and mapping of the various PDH signal rates and SONET composite rates into payloads within a synchronised frame structure. Also defined within these frame structures are a set of overhead channels and signalling protocols that are used to support enhanced operations and management functionality. The hierarchy of frame rates, known as the Synchronous Transport Signal (STS) frame rates, is shown in Table 2.1.

SONET	SDH	Frame Rate
STS-1	-	52 Mb/s
STS-3 (STS-3c)	STM-1	155 Mb/s
STS-9	-	466 Mb/s
STS-12	STM-4	622 Mb/s
STS-18	-	933 Gb/s
STS-24	-	1244 Gb/s
STS-36	-	1866 Gb/s
STS-48	STM-16	2488 Gb/s

**Table 2.1** SONET and SDH frame rates

In 1988, the first phase of international standardisation began, when the ITU-T, formally CCITT, released its initial recommendations [2]. These recommendations, which were drawn up by the CCITT Study Group and the regional standardisation bodies (ANSI - in North America, and CEPT/ETSI<sup>1</sup> - in Europe), merged the existing transmission rates with the enhanced operations and management facilities defined by SONET to form the SDH standards. These included standards for new transmission signal rates (Recommendation G.707 [3]), optical interfaces (G.708 [4]), and synchronous multiplexing structures (G.709 [5])<sup>2</sup>. As shown in Table 2.1, SDH adopted the term STM (Synchronous Transport Module) to define the hierarchy of transmission rates and corresponding frame structures.

The aim of the SDH recommendations was to reach a high degree of compatibility with SONET and therefore define a truly world-wide standard. However, in reality the transmission structure defined by SONET differs significantly from the ITU-T recommendations for SDH. In Table 2.1, it can be seen that a common rate exists at 155 Mb/s and this has been recommended as the lowest common rate for international SDH traffic. However, the SONET 155 Mb/s rate (STS-3) is constructed from a byte interleaved version of the STS-1 frame, with the result that the pointers and associated overheads do not correspond directly to that of an equivalent STM-1 frame. To accommodate this, an alternative SONET frame (STS-3c) was defined based on the SDH frame structure. The STS-3c

---

<sup>1</sup>Conference of European Postal Telecommunications/European Telecommunications Standards Institute.

<sup>2</sup>Additional recommendations have since been drawn up: G.781, G.782 and G.783 (Equipment), G.957 and G.958 (Optical and line interfaces), and G.784 (SDH network management functions) [6].



can thus be transported and byte interleaved to higher rates in the same fashion as an STM-1 for transport over common bearers.

### 2.1.2 SDH Frame Structure

The Synchronous Transmission Mode-level 1 (STM-1) is the basic frame structure and the first level of the SDH transmission hierarchy. This 155 Mb/s frame structure, as depicted in Figure 2.2, consists of 270 columns by 9 rows of bytes. Bytes are transmitted row by row from left to right, and the basic time constant of 8,000 frames/sec (125  $\mu$ s/frame) is preserved<sup>3</sup>. Since one frame is transmitted every 125  $\mu$ s, each byte is equivalent to a 64 Kb/s channel.

The STM-1 frame is partitioned into various functional regions. The first nine columns contain the Section OverHead (SOH) and the frame pointer bytes (see Section 2.1.3). The remaining 261 columns are used to carry the synchronous information payload which includes an extra nine bytes for Path OverHead (POH) (see Section 2.1.3).

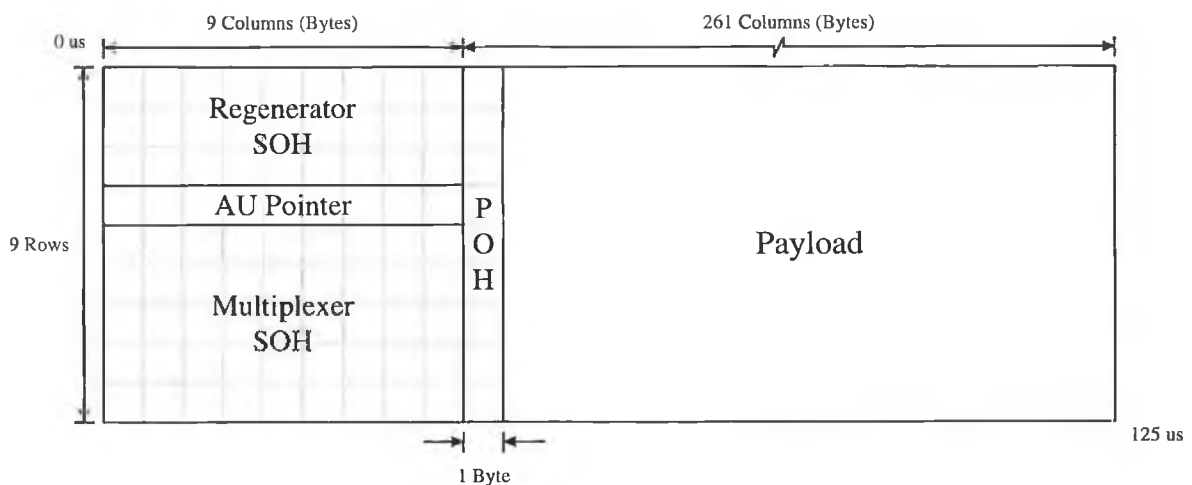
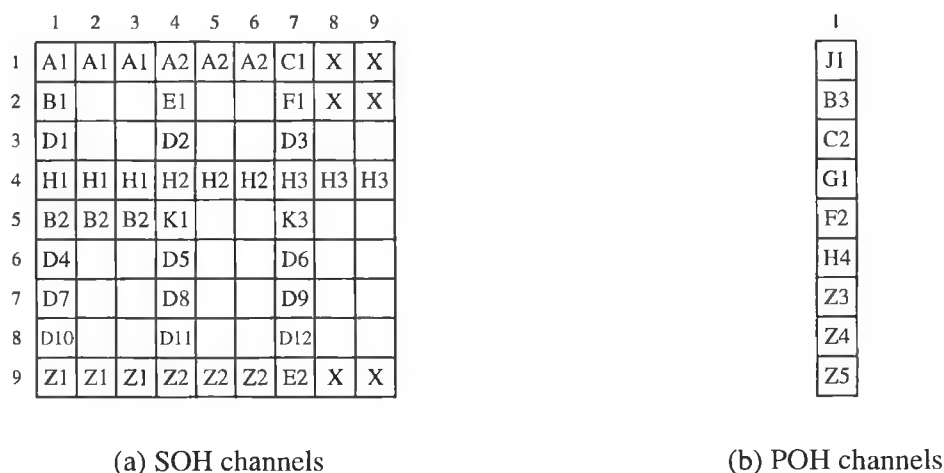


Figure 2.2 STM-1 frame structure

<sup>3</sup>This is consistent with Pulse Coded Modulation (PCM) techniques used to transform analog voice signals to digital bit streams. In accordance with Nyquist's Law, each voice signal requires a 64 Kb/s stream (the product of 8 kHz sampling by an 8-bit (byte) per sample coding).

### 2.1.3 Layer Concept and SDH Overhead Channels

The SDH signal is transmitted hierarchically via path, multiplexer, and regenerator sections, with each transmission section forming a layer. To reflect this layering concept, SDH overhead channels are divided into Path OverHead (POH) and Section OverHead (SOH), with the SOH being further divided into the Regenerator-SOH (RSOH) and the Multiplexer-SOH (MSOH). Figure 2.3 identifies the various overhead bytes and their relative positions within the frame structure. The division of overhead bytes according to layer functionality reflects the separation of processing functions within the various network elements.



**Figure 2.3** SDH overhead channels

The RSOH contains the overhead channels processed by all SDH network elements. These RSOH channels include framing bytes (A1s, A2s), which indicate the start of each frame, an STM-1 identification byte (C1), an 8-bit Bit-Interleaved Parity (BIP-8) check for error monitoring (B1), an orderwire channel (E1) for network maintenance personnel communications, a channel for unspecified operator applications (F1), and a three-byte data communication channel to carry maintenance and provisioning information (D1, D2, D3).

The MSOH channels are processed by all SDH equipment except regenerators. They include the STM-1 pointer bytes (H1s, H2s, H3s), a BIP-24 for line error monitoring (B1, B2, B3), a two-byte Automatic Protection Switch (APS) message channel (K1, K2), a nine-byte line data communications channel (D4-D12), six bytes reserved for future growth (Z1s, Z2s), and a line orderwire channel (E2).

The POH channels are processed at the STM-1 payload terminating equipment and are used for end-to-end path communications. They include a path trace byte (J1), a path BIP-8 for end-to-end

payload error monitoring (B3), a signal label to identify the type of payload being carried (C2), a path status byte to carry maintenance information (G1), a network user channel (F2), a multiframe alignment byte for position indication (H4), and three bytes (Z3-Z5) for use by path user and network operator.

### 2.1.4 STM-1 Payload

The recommendation G.707 specifies the technique for mapping, aligning and multiplexing lower order PDH tributary rates (G.703) into a synchronous signal carried within the STM-1 payload. The various PDH interface rates supported by SDH are shown in Figure 2.4. The acronyms C (Container), VC (Virtual Container), TU (Tributary Unit), TUG (Tributary Unit Group), AU (Administrative Unit), and AUG (Administrative Unit Group) define various stages of the required signal processing.

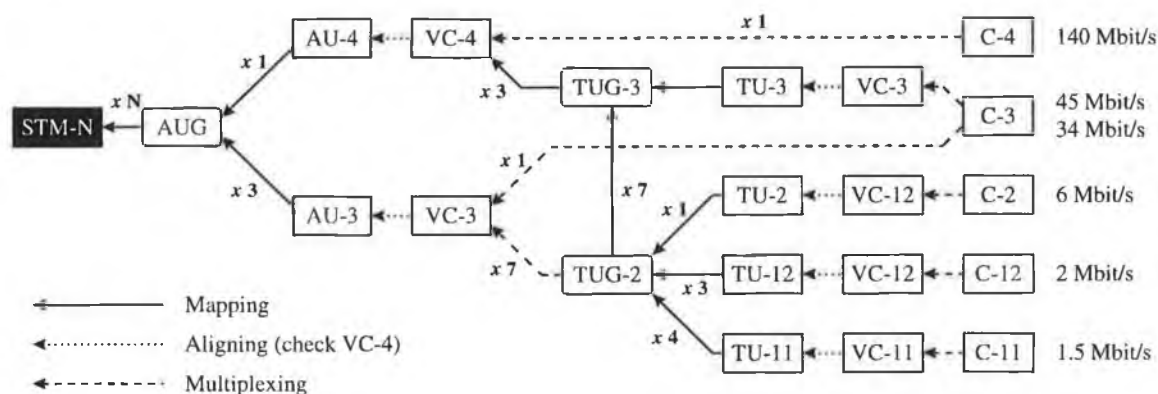


Figure 2.4 SDH multiplex structure

**Container, C-n (n=1x-4):** A Container is an information structure which forms the synchronous payload for each VC. There is a Container defined for each PDH rate.

**Virtual Container (VC):** A VC is an information structure which consists of an information payload and POH channels, and is used to support path-layer connections. Two types of VCs are defined: lower order VCs (VC-n, n=1x,2), which comprise of a single C-n (n=1x,2) plus a lower order VC POH appropriate to that level; and higher order VCs (VC-n, n=3,4), which comprise of either a single C-n (n=3,4) or an assembly of TUG-2s or TUG-3s, together with a VC POH appropriate to that level.

**Tributary Unit (TU):** A TU is an information structure which provides adaptation between the lower and higher order path-layers. The TUs (TU-n, n=1,2,3) consist of an information payload (a

VC-n) together with a TU pointer. The TU pointer indicates the offset of the payload frame relative to the higher order VC frame.

**Tributary Unit Group (TUG):** One or more TUs occupying fixed/defined positions in a higher order VC payload is termed a TUG. Two types of TUGs are defined: a TUG-2, which consists of a homogeneous assembly of identical TU-1s or a TU-2; and a TUG-3, which consists of a homogeneous assembly of TUG-2s or a TU-3.

**Administrative Unit (AU):** An AU is an information structure which provides adaptation between the higher order path-layer and the multiplex section-layer. The AUs (AU-n, n=3,4) consist of an information payload (a VC-n) and an AU pointer. The AU pointer indicates the offset of the payload frame with respect to the STM-1 multiplex section frame (i.e. phase alignment). Two AUs are defined: an AU-4, which consists of a VC-4 plus an AU pointer; and an AU-3, which consists of a VC-3 plus an AU pointer.

**Administrative Unit Group (AUG):** An AUG consists of a homogeneous, byte-interleaved, assembly of three AU-3s or an AU-4 occupying fixed/defined positions in an STM payload.

**Pointers:** Pointers enable the flexible and dynamic alignment of VCs within the AU and lower order TU frame structures. Dynamic alignment implies that the VC is allowed to "Float" within the AU frame. Thus the pointer is adaptive to changes in VC phase and frame rates. The highest order pointer, the AU-4 pointer, is contained in bytes H1, H2 and H3. In the case of three AU-3s, there are three pointers (each availing of a single H1, H2, H3 byte). The application of pointers and their detailed specifications are given in G.709.

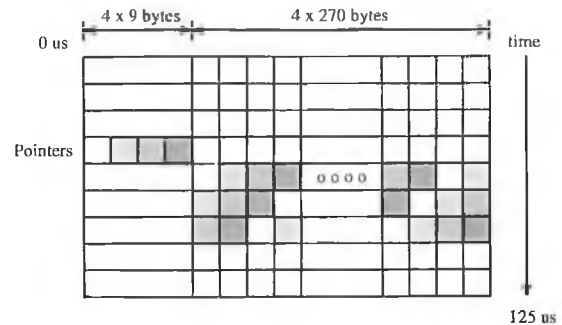
**Mapping SDH:** A procedure by which tributaries are adapted into VCs at the boundary of an SDH network.

**Aligning SDH:** A procedure by which the frame offset information is incorporated into a TU or AU when adapting to the frame reference of the supporting layer.

**Multiplexing SDH:** A procedure by which multiple lower order path layer signals are adapted into a higher order path, or the multiple high order Path Layer Signals are adapted into a multiplex section.

**Concatenation:** A procedure whereby multiple VCs are associated with one another and their combined capacity can be used as a single Container.

**Multiplexing to higher order STM-N rates:** Higher order SDH rates are obtained by byte interleaving  $N$  frame-aligned STM-1s to form an STM- $N$  frame structure, where  $N = 4$  (622 Mb/s) and 16 (2.4 Gb/s). Frame alignment and byte-interleaving enables STM- $N$  frames to carry broadband payloads of approximately 560 and 2240 Mb/s respectively. Figure 2.5 shows how an STM-4 is constructed from four STM-1s. The STM-16 is constructed in much the same way, either by interleaving sixteen STM-1s or four STM-4s. Although the STM-16 (2.4 Gb/s rate) is the highest rate defined so far, higher rates are proposed, the only restriction being technology.



**Figure 2.5** STM-4 frame structure

### 2.1.5 Benefits of SDH

**Worldwide Standard:** SDH is a worldwide standard which unifies European, North American and Japanese transmission rates, and supports the interconnection of existing PDH networks. This means that SDH can be overlaid on the existing PDH infrastructure, thus permitting new services to be developed and introduced gradually. The utilisation of the existing facilities also allows for an efficient and cost-effective evolutionary strategy towards SDH.

**Enhanced OA&M Capabilities:** SDH overcomes many of the problems associated with the existing PDH systems. It defines a comprehensive range of easily accessible overhead bytes which can support advanced network management capabilities. Based on these management facilities, network operators can provide enhanced and highly reliable services to customers through pre-emptive maintenance and dynamic service restoration.

**Enhanced Network Elements:** The simplified method of synchronous multiplexing defined by SDH enables the extraction and insertion of tributary signals directly into higher order signal rates without intermediate multiplexing. This has led to the development and standardisation of advanced and highly intelligent network elements, i.e. Terminal Multiplexers (TMs), Add-Drop Multiplexers (ADMs), and Digital Cross-Connects (DXCs). These network elements provide dynamic routing and restoration capabilities through the distributed management of network facilities (see section

2.2). The standardisation of these network elements should also facilitate the interworking of SDH equipment from different vendors<sup>4</sup>.

**Improved Performance & Flexibility:** The improved network performance and flexibility offered by SDH will enable network operators to respond quickly to meet customers' short or long term demands, while providing a more reliable and resilient network infrastructure for those user services that require a high degree of network integrity. Thus, the initial capital costs, attributed to the introduction of new network elements, should be offset by reduced operating costs and increased revenues from highly reliable and cost effective services.

**Broadband Networks:** While the short term focus is likely to be on cost savings, the long term goal is to provide a more flexibly managed and resilient network for high capacity broadband services. SDH offers easy accommodation of new services and technologies, such as B-ISDN and ATM, with the planned migration to higher order signal rates, such as STM-64 (10 Gb/s), as technology evolves.

## **2.2 Review of SDH-Based Network Elements**

As mentioned in the previous section, the simplified multiplexing/demultiplexing technique offered by SDH eliminates the need for the inefficient multi-step PDH-based process. As a consequence, network element structures have become more simplified and economically viable. This has led to the development and standardisation of highly intelligent network elements, such as TMs, ADMs and DXCs, which support access to the existing network infrastructure as well as forming the infrastructure for a fully operational SDH network (see Section 2.4). While high capacity optical line systems, and to a lesser extent radio relay systems, will provide the transmission medium, it is these network elements that will provide the distributed switching, routing and management functions needed to deliver broadband services. Moreover, it is only through these network elements that the cost benefits offered by SDH can be fully realised.

### **2.2.1 Terminal Multiplexer (TM)**

The TM is the most basic SDH network element. It provides point-to-point access to the core network for lower order traffic rates including existing PDH rates (G.703). Forming a tree or star-

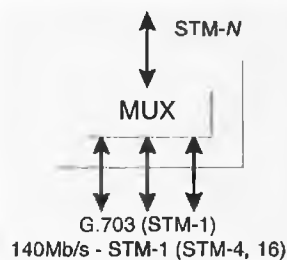
---

<sup>4</sup>In practice this is not yet the case. Differing interpretation of the standards by different manufacturers has, as yet, prevented true interpretability between SDH equipment from different vendors [7]

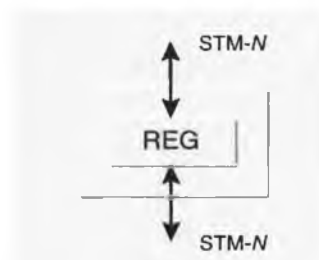
type topology, the TM is used to aggregate lower order traffic rates at the periphery of the access network. As depicted in Figure 2.6 (b), the TM can also be configured to offer regeneration functionality on long distance fibre spans, i.e. greater than 70 Km [8]. The following is a description of the TM equipment specifications at the various STM-*N* levels (Figure 2.6 (a)).

**TM (STM-1):** provides simple G.703 to STM-1 multiplex/demultiplex functionality; 63 (2 Mb/s) signals, 3 (45 Mb/s) signals, or a combination of these or other G.703 line rates can be multiplexed/demultiplexed to form an STM-1 at the input/output port.

**TM (STM-4, 16):** provide the same functionality except at the 140 Mb/s and/or STM-1 to STM-*N* rates. However, this can be extended to the lower order G.703 rates by applying an additional G.703 to STM-1 multiplex/demultiplex operation. This extension can be equally applied to the operations of ADM and DXC equipment.



(a) Terminal Multiplexer



(b) TM configured as a regenerator

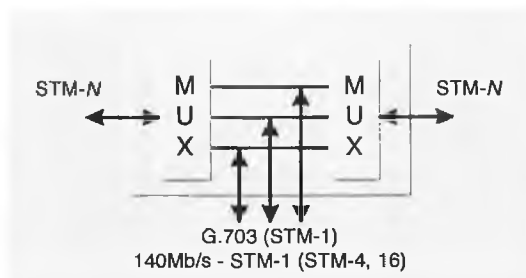
**Figure 2.6** Terminal Multiplexer (TM)

### 2.2.2 Add-Drop Multiplexer (ADM)

The ADM provides the same multiplexing functionality as the TM. However, it also provides low cost access to traffic on the core network. The ADM is the key component in ring-type topologies, which introduce increased service flexibility in both urban and rural areas. A group of ADMs, forming a ring-type topology, can be managed as a separate entity and thus offer distributed configuration and restoration. The following is a description of ADM equipment specifications at the various STM-*N* levels (Figure 2.7).

**ADM (STM-1):** has two I/O ports which enables signal rates to be either routed through or add/dropped at each ADM. Like its TM counterpart, the STM-1 based ADM provides G.703 to STM-1 multiplex/demultiplex functionality.

**ADM (STM-4, 16):** provide the same multiplex/demultiplex functionality except at the 140 Mb/s and/or STM-1 level.



**Figure 2.7** Add/Drop Multiplexer (ADM)

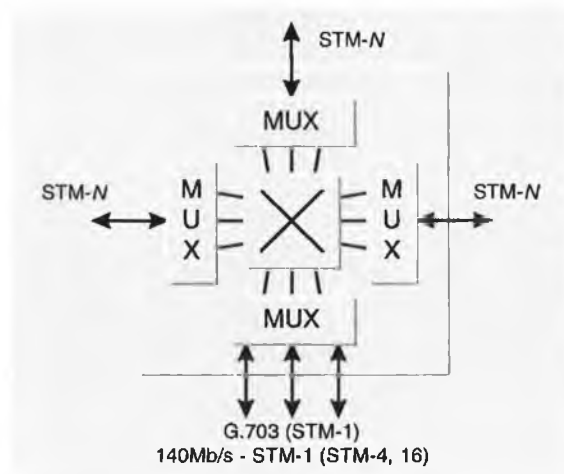
### 2.2.3 Digital Cross-Connect (DXC)

The DXC supports cross-connection of SDH and existing PDH signal rates through mapping and multiplexing at the STM-N line rates. It has the capability of monitoring OA&M overhead channels, and thus supports distributed control and enhanced network management operations. Together with its transparent switching characteristic, these capabilities can provide flexible and distributed restoration and reconfiguration. The DXC is the key component in the core network, forming a mesh-type topology which serves to groom/consolidate traffic in the access network. The following is a description of DXC equipment specifications at the various STM-N levels, (Figure 2.8).

**DXC (STM-1):** can have up to 256 I/O ports which allow signal rates to be either terminated or remultiplexed back (cross-connected) into an appropriate outgoing signal. The STM-1 based DXC, also known as a wideband DXC, provides cross-connection at the G.703 levels.

**DXC (STM-4, 16):** also known as broadband DXCs, these DXCs provide the same cross-connect functionality as the wideband DXC except at the 140 Mb/s and/or STM-1 level. It is noted that additional ports can be configured within each DXC, but this incurs much higher costs per port relative to the standard cross-connect configuration.





**Figure 2.8** Digital Cross-Connect (DXC)

## 2.3 SDH Survivable Topologies

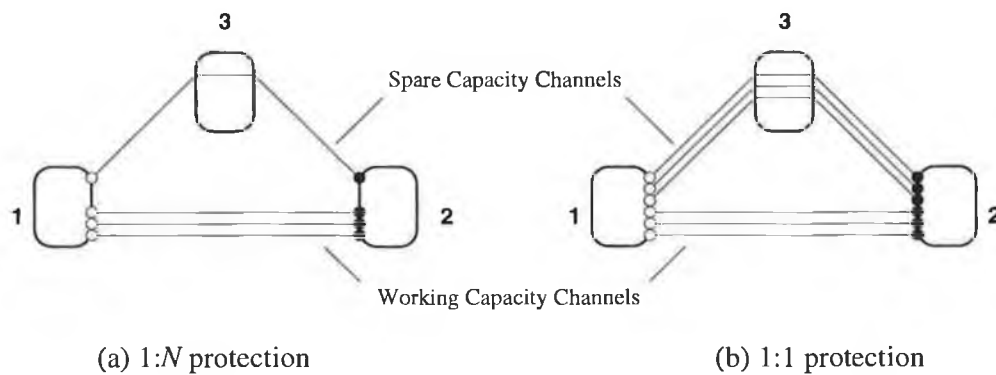
As the deployment of high capacity fibre transmission systems escalates, emphasis on service quality and network reliability has increased considerably. For the network operator, improved network reliability is essential both from the perspective of lost revenue and the fact that their customers, particularly business users, are now demanding a higher level of service quality. Moreover, as the deregulation of the telecoms industry nears and operator competition becomes more prevalent, these customers can dictate their choice in service, and demand better overall quality.

One of the main aims of the SDH standardisation process was to address the issue of network survivability. It has resulted in the development of enhanced self-healing techniques which are supported by the various network elements and their characteristic topologies. A review of these survivable topologies and restoration techniques is presented in the following sub-sections.

### 2.3.1 TM Point-to-Point Topologies

A TM topology provides point-to-point connections between any pair of nodes in the network. As such, it is said to have a connectivity requirement of one, where the connectivity requirement defines the number of node disjoint paths that exist between any pair of nodes. In terms of survivability, the connectivity requirement also defines the minimum number of links that must be removed before a node becomes disconnected. As a result, TM-based point-to-point survivable topologies must use diversely routed spare capacity links to provide protection against link failures. As shown in Figure 2.9, the spare capacity link can be configured in a 1:N or 1:1 arrangement. In

the case of 1: $N$  protection, one spare capacity channel is assigned to  $N$  working channels. In the event of a link failure, working channels are restored on a priority basis with the services carried on  $N-1$  channels being lost. As the emphasis on service quality increases, this arrangement is no longer acceptable, and the use of 1:1 protection has become the prerequisite. In this topology, a spare capacity link is assigned to each working capacity link. The restoration technique itself is based on automatic protection switching (APS). It works by transmitting the working signal on both links simultaneously. The receiving node then monitors the working signal and if it is lost or degraded it automatically switches to the spare capacity signal. Since the switching time is negligible, this is regarded as static instantaneous restoration. While this has an excellent response time, the drawback to this survivable topology is that it requires a large amount of redundant capacity on dedicated diversely installed links.



**Figure 2.9** TM point-to-point topologies

### 2.3.2 ADM Ring Topologies

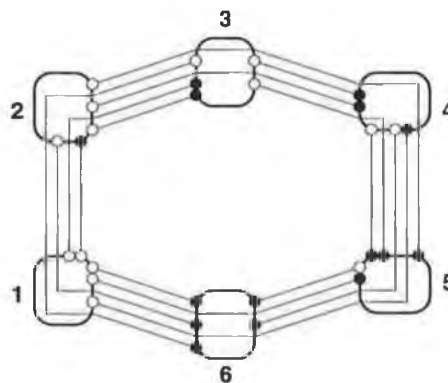
An ADM ring topology is composed of a group of nodes that support two way connections. In this case, each node (or nodal pair) will have a connectivity requirement of exactly two. That is, a minimum of two links must be removed before any node becomes disconnected (or two node disjoint paths exist between each pair of nodes on the ring). Several ring configurations, such as Uni-directional, Bi-directional and two/four fibre, have been proposed [9-16]. The ADM ring topology supports many of the enhanced features offered by SDH and is regarded as one of the most promising and cost effective restoration schemes. For example, an ADM ring can be managed as an entity and thus offers distributed configuration management. Due to this characteristic, this survivable topology is often referred to as an ADM self-healing ring (SHR). In general, this form of restoration is considered static (pre-determined) restoration, i.e. ~50 ms. In this topology, spare capacity is assigned to each link in a distributed fashion and is shared by all working links on the ring. As a result, there is a saving on the spare capacity requirement compared with the TM-based

1:1 APS schemes. There are two basic forms of distributed restoration proposed for ADM SHR topologies: Line-level and Path-level.

### 2.3.2.1 Line-level restoration

In line-level restoration, restored routes are formed around the failed link at the line-level upon detection of line alarms, such as LOS (Loss Of Signal) and line AIS (Alarm Indication Signal)<sup>5</sup>. Since spare capacity is diversely distributed and shared by all working traffic on the ring, the amount of redundant spare capacity is reduced. In order to coordinate this sharing, a set of messages must be transmitted between each node on the ring so that the spare capacity can be suitably assigned. Since these messages are defined at the line-level, they are accommodated within the SDH SOH.

To illustrate the concept of line-level restoration, consider the six node ring topology depicted in Figure 2.10, with all inter-nodal paths<sup>6</sup> shown clearly. We shall now consider the potential link failure between nodes 1 and 6. The objective of the restoration process is to reroute the inter-nodal demands traversing this failed link.



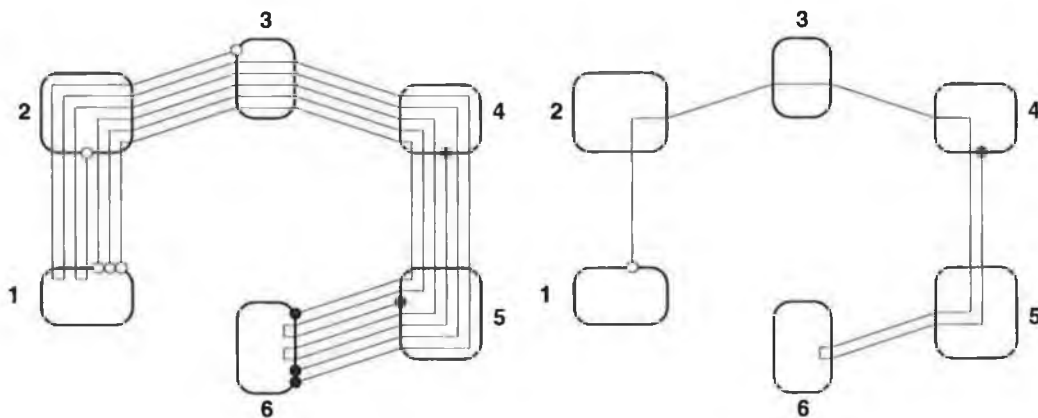
**Figure 2.10** ADM ring topology

---

<sup>5</sup>In a more simplified point-to-point based technique, each sending node on the ring transmits two copies of the same signal simultaneously in opposite directions around the ring, and if the working signal is lost or degraded, the receiving node automatically switches to the spare channel. However, this requires a dedicated (redundant) protection ring (spare capacity) for each working line.

<sup>6</sup>The method of routing inter-nodal demands is discussed in Chapter 3, Network Optimisation.

In the case of line-level restoration, the process is initiated by the terminating nodes of the failed link (designated as the sender and chooser respectively<sup>7</sup>). In this example, node 1 denotes the sender and node 6 the chooser. As depicted in Figure 2.11, since restored routes are configured between these nodes, the inter-nodal paths using the failed link do not need to be reconfigured. The restoration process can therefore perform revertive switching in a simple way when the failure is repaired. However, since reconfiguration of inter-nodal paths is not supported, the restoration process is limited to link failures, i.e. it cannot protect against nodal failures. The solution to this problem is to use path-level restoration, which is a relatively more complex but even more bandwidth-efficient technique.



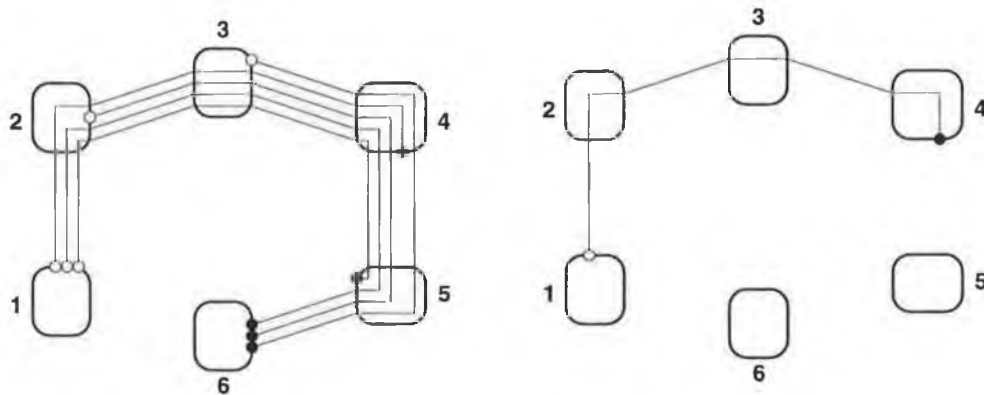
**Figure 2.11** Line-level restoration

### 2.3.2.2 Path-level restoration

In the case of path-level restoration, Figure 2.12, if the working signal on any given path is lost or degraded, the failed traffic is restored between the terminating nodes at the path level. In other words, the terminating nodes of each inter-nodal path traversing the failed link initiates their own independent restoration process. The message passing channels are therefore accommodated within the SDH POH, and the detection of the path AIS is the trigger for the process.

---

<sup>7</sup>The restoration is initiated at the sender and terminated at the chooser. A simple rule can be applied to determine the pair, i.e. the node with the smaller address is the sender and the other, the chooser [10].



**Figure 2.12** Path-level restoration

In regard to the viability of this technique, since restored routes are formed at the path level, the spare capacity requirement is reduced considerably. This can be reduced even further by performing a pre-processing phase, known as the release phase, which enables the working capacity on the failed paths to be released and therefore used to reroute failed capacity. Moreover, protection against nodal failures is also supported. The network can therefore protect against most network failures, and thus guarantee a better quality of service. However, since the management of path-level restoration is relatively more complex, additional processing capabilities must be embedded within each network element. In order to justify the deployment of network systems based on this restoration scheme, the overall cost benefits must offset the additional management and operational costs. In regard to the latter, it is expected that advances in network management systems and reduced equipment costs will ensure that path-level restoration will become the standard technique in future SDH networks.

### 2.3.3 DXC-based Mesh Topologies

A DXC-based mesh topology is formed by a group of nodes that have a connectivity requirement that exceeds two. That is, at least three node disjoint paths exist between each pair of nodes in the network configuration. As stated in Section 2.2.3, DXCs can provide distributed control for enhanced configuration management operations, such as distributed restoration and dynamic reconfiguration. As such, it is also referred to as a self-healing technique and is regarded as the most flexible restoration scheme [10], providing relatively fast restoration ( $\sim 2$  s), at either the line or path level. In this topology, spare capacity is assigned in a distributed fashion across the network and is shared by a greater number of failure scenarios. In the case of a link failure, the failed working capacity can therefore be rerouted over a variety of diverse paths. This is in contrast to the single alternative path offered by point-to-point and ring topologies. As a result, this topology provides a

higher level survivability against multiple link as well as nodal failures, while minimising the spare capacity requirement.

DXC-based self-healing is defined as a distributed restoration scheme whose control is initiated and executed locally at each DXC in an autonomous distributed fashion [9]. An alternative restoration scheme using centralised control could also be used, although the overall restoration time may take upwards of 10 minutes, which is totally unexceptionable<sup>8</sup>. While distributed control is more efficient, economic considerations must also be considered. These issues are dealt with in Appendix A1, where a comparison of both techniques is provided.

Distributed control involves the exchange of messages between DXCs in response to a network request/event such as a link failure<sup>9</sup>. Since intelligence resides locally at each DXC, the DXC controllers act like parallel processors computing alternative routes dynamically in real-time. Several restoration schemes have been proposed and studied. Although these studies cite that distributed restoration techniques can restore services within 2 s, actual restoration times may range from 150 ms up to tens of seconds depending on, among other factors, network size and the severity of the failure. As most services are impacted by a service outage greater than 2 s, the objective would be for the restoration of all single link failures. These issues are addressed in more detail in Appendix A2, where techniques used to improve restoration times are presented.

In general, the DXC-mesh topology can provide a higher level of survivability against multiple link as well as nodal failures, while minimising the spare capacity requirement. However, due to its more complex restoration and switching process, there is a need for more sophisticated CPU and memory storage capabilities. As such, it is deemed too expensive and unsuitable for sparse traffic [9, 10, 17]. DXC-based mesh topologies are therefore more applicable to the core network (see section 2.4.1), where traffic has already been concentrated from the lower network levels, while ADM and TM topologies tend to be more appropriate to the access network, where traffic must be groomed and aggregated up to the various STM signal rates.

---

<sup>8</sup> Studies show that most service are impacted by a service outage greater than 2 s [see Appendix A1].

<sup>9</sup>The concept of request/event message passing can also be applied to other forms of management processing, such as dynamic network reconfiguration (DNR) in response to significant changes in traffic patterns across the network [18].

### 2.3.4 Hybrid ADM-ring/DXC Topology

In reality, however, no such clearly defined boundary between the various topologies and their applications will exist. In particular, in the region between the core and access networks there exists an intermediary network hierarchy, the regional junction network (see section 2.4.1), which facilitates the interconnection between these networks. Due to the variance in its network characteristics, it is expected that the most cost effective solution within this regional junction network will consist of a hybrid topology which maximises the simplicity of ADMs and the flexibility of DXCs [9, 16, 19].

One of the main aims of the SDH standardisation process was to define a common network interface that allows direct interconnection of different network elements. This ensures that DXC-based mesh and ADM-based ring topologies can offer complementary rather than alternative network topologies. Since each topology is dependent to varying degrees on network characteristics, relative costs, and the required level of survivability, many different scenarios can be envisaged. In one such scenario, known as a "logical ring application" [16], a set of partial rings (chains of ADMs) are interconnected by DXCs, see Figure 2.13. Traffic from an ADM can terminate at ADMs either on the same chain or on any other interconnected chain, with the DXCs providing the interconnection capabilities. Depending on the network characteristics, it can be seen how variations on the above topology could be used.

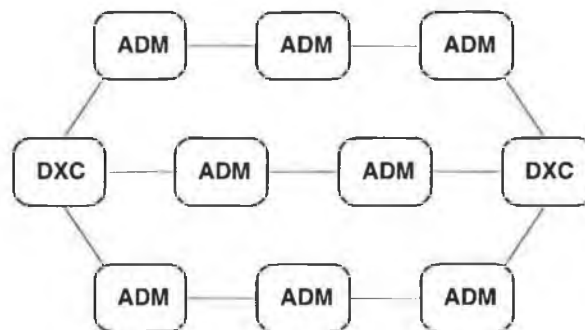


Figure 2.13 Hybrid topology (logical ring application)

### 2.3.5 Comparison of Survivable Topologies

From a network designers perspective, the unique characteristics and inherent cost benefits of the various network topologies must be taken into consideration in order to realise an optimum cost-effective solution. A summary of the various restoration techniques is provided in Table 2.2.

	TM-1:1 APS	ADM-Ring	DXC-Mesh
Restoration Time	Excellent	Good (~50 ms)	Fair (2-3 s)
Flexibility	Poor	Good	Excellent
Failures Handled	Fair	Good	Excellent
Spare Capacity	Poor	Good	Excellent

**Table 2.2** Survivable topology characteristics

## 2.4 SDH Network Architectures and Deployment Strategies

In this section, the current trends in SDH deployment as discussed in some of the many papers and articles published in this area are reviewed [7, 8, 16, 20-26]. As will be shown, there is no universal strategy for early SDH deployment because the benefits of such will vary from operator to operator, depending on, for example, the extent and age of the existing network infrastructure, the service aspirations and forecasted growth in traffic, and implications of future competition. What is clear is that SDH benefits and technology are accepted, to the extent that almost all network operators have been engaged in extensive trials of SDH products and various network configurations, as a basis for formulating and planning widescale deployment strategies. It should be noted that the information pertaining to the various papers and articles referenced in this and previous sections does not present or quantify exact SDH network solutions, but provides intuitive and invaluable information regarding general network characteristics and potential solutions.

### 2.4.1 SDH Network Architecture

The existing PDH network architecture is based on the exchange hierarchy it interconnects, namely<sup>10</sup>: the national trunk network, which interconnects the Trunk Exchanges (SX) and International Exchanges (IX); the regional junction network, which interconnects the Tandem Exchanges (TX) and the Local Exchanges (LX); and the local access network, which provides the link between the LXs and the customer premises. Each network level is characterised by different traffic loads and network configurations. While SDH will provide a more reliable and flexibly managed network infrastructure, practical considerations dictate that most of the existing fibre optic

---

<sup>10</sup>It should be noted that, while the naming conventions for the various network hierarchies may vary from operator to operator their characteristics and general attributes are the same.



cable and, in particular, node infrastructure be used. This will result in a hierarchy that will still closely relate to the exchange hierarchy it supports.

#### **2.4.1.1 National trunk network**

In the national trunk network, the traffic has already been concentrated from the lower network levels and serves to transport high-capacity aggregated traffic between the trunk nodes. The topology of this network will generally consist of a fully meshed network at the STM-4/STM-16, and, later, STM-64 signal levels, with broadband DXCs and ADMs providing restoration and grooming at the STM-1 signal level. The network acts as a gateway for the interconnection of the various regional junction networks, and will provide a restoration scheme with at least two diverse routes to two independent nodes in each of these networks.

#### **2.4.1.2 Regional junction network**

The regional junction network consists of tandem nodes which consolidate and groom traffic between the local access network and local exchanges it serves. In general, the topology of this network will consist of a hybrid network at the STM-1/STM-4 signal levels, with wideband DXCs and ADMs providing restoration and grooming at the G703 signal level. The ADM rings will serve minor transmission centres associated with LXs, while the wideband DXCs will serve major transmission centres associated with TXs and provide interconnection between the rings. The practical choice and combination of these two alternatives will depend on the network characteristics and under what conditions is it economical to deploy DXCs. In the case of large metropolitan areas, the topology might consist of broadband DXCs, wideband DXCs and ADMs, providing high-capacity aggregated links between major switching centres with the interconnection of ADM ring networks serving the minor switching centres. A restoration scheme based on dual parenting is also assumed.

#### **2.4.1.3 Local access network**

The local access network level will consist extensively of ADM ring topologies, while other topologies, such as TM/ADM star/hub topologies, will be used to aggregate the lower order traffic rates at the periphery of the access network. The ADM ring topology will provide consolidation and flexible routing between nodes at the STM-1 signal level. In addition, it offers fast and autonomous restoration at the G.703 signal rates and offers substantial advantages over the traditional point-to-point and star/hub topologies.

## 2.4.2 SDH Deployment Strategies

As mentioned above, there is no universal strategy for early SDH deployment because the benefits of such will vary from operator to operator. However, the following is a review of the main deployment strategies currently being used.

**Top-down:** The top-down approach involves the deployment of broadband DXCs and high capacity line systems into the regional and core networks. It is also proving attractive to those operators looking to increase their fibre capacity/utilisation and overall reliability.

**Bottom-up:** The bottom-up strategy is based on the deployment of small "islands" of SDH, such as STM-1 access rings, initially interconnected by SDH/PDH transmission links. Its main application is for network expansion in key areas, such as major business sites.

With both approaches, the full end-to-end benefits offered by SDH are limited to traffic which is originated, transported, and terminated wholly within these SDH network regions. As a result, a further two options are being adopted.

**Overlay:** The overlay option involves the creation of new regional or national SDH networks which can later be extended to a more general network. This might, initially, be deployed to provide high integrity premium services for major network users and provide the foundation for more general network deployment.

**Rip-out:** This approach involves the complete replacement of the existing infrastructure, over a relatively short period, on the assumption that the overall return on investment will be more quickly realised than from a hybrid PDH/SDH network.

All four strategies have their merits, with the bottom-up approach being favoured by many operators on the basis of return on investment [8, 24]. Alternatively, for smaller network operators, such as utilities, the rip-out approach may be more effective. In practice, early network deployment strategies tended to be evenly split between the first two approaches, although the trend appears to be moving more towards the overlay network.

In Germany [20], Deutsche Telekom first began the construction of an SDH core network in 1990. It is based on high-capacity broadband DXCs, interconnected by STM-4,16 line systems. This was followed by their VISYON pilot program in 1994, which involved the introduction of ADM rings based on STM-1 and STM-4 line systems.

In France [7], France Telecom adopted the top-down approach by initially targeting their core network, followed later by the introduction of SDH rings in the junction and access networks. Their main drivers were performance and cost-related benefits, which they believe are essential in an increasingly competitive global market.

In the UK [7], the establishment of an already liberalised environment goes further to illustrate the different approaches and business drivers in deploying SDH, as start-up operators attempt to attract customers and revenue away from the UK's established operators, BT and Mercury. For example, Cable TV operators are focusing on the cost-effective provision of telephony and broadband interactive services, such as video on demand to residential customers. Since they are starting with a "green field" scenario, they can take advantage of the enhanced capabilities and operating cost improvements of a fully integrated SDH network. In response, BT has begun an intensive SDH deployment program, which includes DXCs, ADMs and line systems, as well as an overall network SDH management system.

In the US [25, 26], similar deployment strategies relating to SONET-based equipment have also been witnessed. Other areas of interest include the Far East and the Pacific Basin [23], which represent one of the fastest growing telecommunications regions in the world. These regions have already seen a high degree of SDH activity, with countries like Hong Kong, Singapore, Australia, and New Zealand all moving from technical trials to imminent deployment.

### **2.4.3 Considerations Affecting SDH Deployment**

While it is accepted that SDH will form the infrastructure for future transmission networks, a number of issues affecting the early introduction of SDH have emerged. As was discussed in the previous section, there is no one strategy for realising a fully integrated SDH network. What is clear is the fact that it will have to, to some degree, work within the existing PDH network. This presents additional considerations for network operators in managing and operating these interworking networks. These include the transmission of SDH management information over PDH line systems, interaction of PDH and SDH restoration schemes, and the synchronisation of fragmented SDH islands. Although some of the benefits of SDH are lost in interfacing with existing segments of the PDH network, it is still feasible to do so. This will remain a limitation until SDH can be deployed throughout the network.

---

## Chapter 3 - SDH Network Optimisation

---

In Chapter 2, the benefits of SDH and how it will form the infrastructure for future transmission networks were covered. In particular, the various SDH-based network elements and their characteristic topologies and restoration schemes were reviewed. As discussed, the emphasis on service quality has made network survivability an important design criterion. While the overall objective is to develop an automated design tool that will aid network planners in designing and optimising these networks, the underlying design methodology must first be developed.

Section 3.1 concerns the formulation of the optimal design problem in terms of the notation and data structures used to represent both the input network and the resultant network solution. It also specifies the design constraints which must be satisfied in order to realise a feasible solution. The cost model adopted for this particular problem covered is then presented. Finally, based on this cost model, it will be shown that the problem can be reformulated as a combinatorial optimisation problem. This enables the natural decomposition of the problem into various sub-problems that can then be solved sequentially. The remaining sections concern the development of the solution methods used to efficiently solve these individual sub-problems.

### 3.1 Problem Formulation and Network Representation

This section concerns the formulation of the optimal design problem. It is noted that, while the network model defined is specific to the SDH design problem, the data structures and notation are based on the principles of graph theory and are consistent with those adopted for most network optimisation problems [27-30].

#### 3.1.1 Input data and network representation

The network is denoted as an undirected Graph  $G = (N, L)$ , where  $N = \{1, \dots, n\}$  represents the set of all nodes, and  $L = (l_{i,j})_{i,j=1, \dots, n}$  represents the set of all inter-nodal links. Associated with each link  $l_{i,j}$  between nodes  $i$  and  $j$  is a non-negative distance  $d_{i,j}$ , a demand requirement  $r_{i,j}$  and a fixed cost component  $f_{i,j}^{fixed}$ . These elements form part of the inter-nodal distance matrix,  $D = (d_{i,j})_{i,j=1, \dots, n}$ , the demand requirement matrix,  $R = (r_{i,j})_{i,j=1, \dots, n}$ , and the fixed cost matrix,  $F^{fixed} = (f_{i,j}^{fixed})_{i,j=1, \dots, n}$ , respectively.

As stated in Chapter 2, it is assumed that the geographical location of each node is known in advance and is submitted as an input parameter to the process. The inter-nodal demands are also predetermined values obtained from forecasted traffic models (see section 3.1.3). The fixed cost component represents the cost of installing a link between any two nodes irrespective of the capacity on that link (see section 3.1.4). It is assumed that all links may be present in the initial solution. If a link is not deemed economically feasible, then this is reflected by the fixed cost component of that link. In addition, it is not assumed that the distance matrix satisfies the triangular inequality. That is, for any  $i, j, k \in N$ ,  $d_{i,j} + d_{j,k} \geq d_{i,k}$  does not have to hold.

In denoting the network as being undirected, it can be assumed that matrices  $D$ ,  $R$  and  $F^{fixed}$  are symmetric. This is a realistic assumption and can be applied to most network types including SDH [11-13, 17]. This implies that these matrices can be made upper triangular without loss of generality<sup>11</sup>. This is an important property since it reduces both the computational time and memory requirement of the optimisation problem.

### 3.1.2 Output data and solution representation

The objective of the optimal network design problem is to generate a minimum cost topology, defined as a set of links and the optimal capacity assignment on each link to route all inter-nodal demands and satisfy the required level of survivability. We define an adjacency matrix,  $A = (a_{i,j})_{i,j=1,\dots,n}$ , a working capacity matrix,  $WC = (wc_{i,j})_{i,j=1,\dots,n}$ , and a spare capacity matrix,  $SC = (sc_{i,j})_{i,j=1,\dots,n}$ , to represent the solution. The adjacency matrix is basically a sub-set of  $L$ , the set of all inter-nodal links: where  $a_{i,j} = 1$  if a link is present in the optimal solution and  $a_{i,j} = 0$  otherwise. Matrices  $A$ ,  $WC$ , and  $SC$  are all symmetric following the assumption made above. In addition to these matrices, we define a paths matrix  $P = (p_{i,j})_{i,j=1,\dots,n}$ , which stores the optimal paths used to route the inter-nodal demands. This is generated during the joint optimal routing and working capacity assignment phase of the design process and is used in other sub-routines, e.g. the network connectivity test and spare capacity assignment based on path-level restoration.

---

<sup>11</sup>For example, the demand between nodes  $i$  and  $j$  denoted by matrix element  $r_{i,j}$  or  $r_{j,i}$  is equal and is the sum of the demands in either direction. Similarly, the distance and fixed cost associated with each link or relevant matrix element is the same in both directions.

### 3.1.3 Design constraints

**Demand Requirements:** As discussed in Chapter 2, practical considerations dictate that most of the existing network infrastructure be utilised. This is particularly true of node locations and the exchange hierarchy they support. Each hierarchy is characterised by different traffic loads and network configurations. There is a clear distinction between the access and core network hierarchies with the regional network providing the interconnection. Hence we can consider the design and optimisation of the access and core networks separately by including the regional network infrastructure in both design problems. It is assumed that traffic forecasting and route dimensioning (inter-nodal demands) are estimated in the pre-planning phase of the transmission design problem and are therefore given as input data [9, 29, 31]. Demands will depend on forecasted traffic growth and the service aspirations of operators, e.g. narrowband and broadband data services. For the joint regional/access network problem, each node in the access network will have a demand requirement, given in E1s, for every other node in the same region and a demand requirement for every other region. In the case of the joint core/regional network, each node will have a demand requirement, given in STM-1s, for all other nodes in the network.

**Survivability:** One of the most important design constraints in SDH network optimisation is survivability. The major issue in designing and planning SDH networks is how to best utilise the unique attributes of the various network topologies and their associative network elements to satisfy the various network constraints at a minimum cost. Due to the variance in these characteristics, an optimal solution would consist of a hybrid of these topologies. The overall optimisation process must therefore be capable of evaluating the minimum cost of providing a required level of survivability for any given network configuration in order to compare and derive an optimal network solution.

**Connectivity:** For a solution to be feasible, all nodes (or inter-nodal paths) must satisfy a minimum connectivity requirement of one. That is, at least one node disjoint path must exist between each pair of nodes in the network. As discussed in Chapter 2, the TM, ADM, and DXC network elements and their corresponding topologies have a connectivity requirement of one, two, and greater than two, respectively. Since an optimal solution will consist of a hybrid of these topologies, it is more appropriate to describe the solution in terms of its average connectivity requirement or connectivity characteristic, where the connectivity characteristic defines the number of links in any given solution. This is an important network characteristic since it impacts directly on the network cost components. For example, as network connectivity increases, the fixed cost component, which is a function of the number of links, will increase. Inversely, the variable capacity dependent cost component (see next section) will decrease as inter-nodal demands are routed over shorter link paths

and the spare capacity is shared and distributed over a greater number of diversely routed paths. In Section 3.4.2, it will be shown how this attribute can be used as a constraint in reducing the initial complexity of the design problem.

**Network Cost Function:** To evaluate the cost of any given network solution, we must be able to model its cost. The choice of network cost function is the key to defining an efficient optimisation design process. This influences the choice of routing method and contributes to the overall complexity of the problem.

### 3.1.4 Formulation of Cost Function

Due to the range and complexity of the various cost components contributing to the overall cost of a network, the objective of minimising such cost can be very complicated. Analysis of cost functions has shown that the imposition of suitable assumptions can result in good approximations which simplify the overall optimisation problem [12, 13, 27, 28, 31-33]. Denoting solution space  $S$  as the set of all feasible solutions, let us initially define the general cost function for the installation of a given network solution,  $s \in S$ , in terms of nodal and link costs:

$$F(s) = F(A, WC, SC) = \sum_{\forall n \in N} f_n(wc_n + sc_n) + \sum_{\forall l \in L} f_l(wc_l + sc_l) \cdot a_l \quad (3.1)$$

In this equation,  $f_n(wc_n + sc_n)$  and  $f_l(wc_l + sc_l) \cdot a_l$  define general functions for the capacity switched (nodal) and capacity routed (link) costs, respectively. These cost functions shall now be reviewed separately in the context of approximating the overall cost model.

#### 3.1.4.1 Nodal cost function

One of the key features of SDH was the development of highly intelligent network elements. As the objective is to design an optimal network solution based on the cost/survivability trade-off between the various network elements and their characteristic topologies, their cost characteristics must somehow be modelled. While most network optimisation techniques focus exclusively on link cost optimisation [17, 28, 32-35], those that do consider nodal costs model all or part of the cost as link costs [31]. This is in keeping with the form of this cost model. To justify this possibly gross assumption, the cost components involved shall be examined.

In general, the nodal cost function can be rather complicated. However, it is a reasonable assumption to approximate the cost as consisting of a fixed set-up cost and a variable, capacity dependent, cost. The nodal cost function therefore becomes:

$$f_n(wc_n + sc_n) = [f_n^{fixed} + f_n^{var} \cdot (wc_n + sc_n)] \quad (3.2)$$

The fixed cost component relates to the set-up cost associated with each network element. It is comprised of the basic node structure and hardware component costs. Since all  $N$  nodes will be present in each potential solution, we can neglect this cost component. The same cannot be said of the variable cost component since it is a function of the network element type and the signal rate at which it operates. A correlation of these costs [9, 12, 13] is shown in Table 3.1, where X represents a reference cost based on the cost of a TM (STM-1).

Signal Rate	TM Cost	ADM Cost	DXC Cost
STM-1	X	X	3.94 X*
STM-4	1.77 X	1.77 X	6.98 X
STM-16	2.97 X	2.97 X	19.58 X

**Table 3.1** Correlation of network element costs

It can be seen that the relative cost at each signal level is the same for TMs and ADMs, while the corresponding cost of DXCs is substantially greater (due to the more complex switching and control hardware required for grooming and cross-connection). This cost component is therefore a function of network connectivity, i.e. the number of links that terminate at each node, and represents the cost of line termination and interface cards. Since the choice of signal rate will depend on the capacity being switched at each node, i.e. terminating link capacities, we can model these costs as a distance independent component of link costs by introducing a terminal cost component  $f^{terminal}$ . Since each link has two terminating end points, the capacity switched component of the link cost function can be given by:

$$f_l^{switched}(wc_l + sc_l) = 2 \cdot (wc_l + sc_l) \cdot f^{terminal} \quad (3.3)$$

### 3.1.4.2 Link cost function

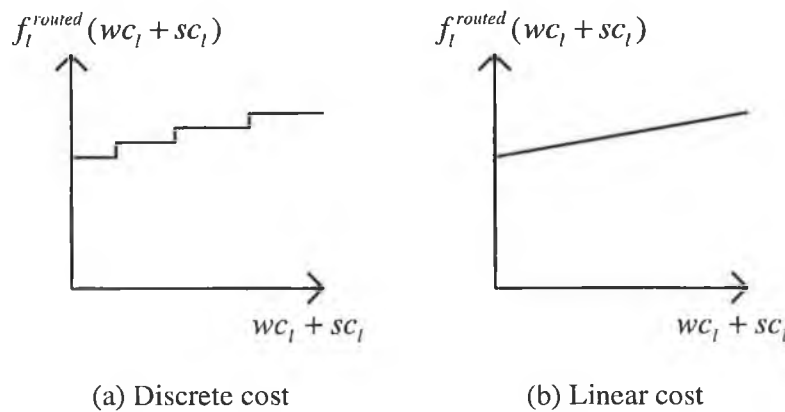
A review of the work published in this area was presented in [27]. This defines a number of generalised cost functions which can be applied to the various facility types, such as fibre optics, coaxial cable, and copper. In [27] and other recent works relating to facility design and planning (for both telecommunications and data networks) [28, 32-35], results indicate that the economies-of-scale associated with high-capacity transmission systems make the deployment and utilisation of

---

\* Configured for 240 (2 Mb/s) ports, with a cost of 0.84 X for each additional bay (60 ports).



fibre-optic line systems a viable choice in network solutions. Since SDH is predominantly based and defined for transmission over fibre optics<sup>12</sup>, in this section we consider the main cost functions relating to this facility type. These are depicted in Figure 3.1 below.



**Figure 3.1** Link cost functions

In both cases, cost is an increasing function of distance. As with nodal cost function, the link cost function can be rather complicated. The same assumption as before can be applied and this cost can be approximated as consisting of a fixed set up cost and a variable, capacity dependent, cost. This gives:

$$f_l(wc_l + sc_l) = d_l \cdot [f_l^{fixed} + f_l^{var} \cdot (wc_l + sc_l)] \quad (3.4)$$

The fixed cost component, which is often a significant percentage of the total cost, represents the cost of installing a fibre cable over link  $l$  irrespective of the capacity on that link. It consists of the cost of obtaining right of way, and the labour and material costs associated with digging a trench and laying down a conduit<sup>13</sup>. Depending on the path and the nature of the installation, these costs may vary dramatically, i.e. digging up a street in a city centre will be much more expensive than working in an open field. In addition, if the fibre is intended as an upgrade over an existing link, then the existing conduit should be utilised where possible. It will be assumed that these costs will be known in advance and given as input data to the design process. As defined in section 3.1.1, the

<sup>12</sup>It is noted that SDH is also defined for transmission over radio (microwave). However, this form of transmission is more applicable to remote sections of the network where land-based links are not feasible. As such, the apparent economic justification and installation of such systems will generally be known in advance and are therefore omitted from the overall design process.

<sup>13</sup>Due to the compact dimensions of fibre cables these costs are assumed to be independent of the number of cables being routed, i.e. the cost is not a function of capacity routed.

respective fixed costs components,  $f_l^{fixed}$ , associated with each link  $l \in L$ , are represented by the fixed cost matrix,  $F^{fixed} = (f_l^{fixed})_{l \in L}$ .

The variable cost component of any link  $l$  is a non-decreasing function of the total capacity being routed over that link. The stepwise cost function, shown in Figure 3.1 (a), where cost is a discontinuous function of capacity, most closely represents reality<sup>14</sup>. While this form of cost function has been applied to some network optimisation problems [31], the spare capacity assignment problem has not been considered, and even for the optimal routing and allocation of working capacity, this leads to inefficiencies in memory requirements and computational time. This is due to the fact that, as proven in [27], optimal splitting of inter-nodal demands across the network must be considered. The problem is complicated by the number of diverse routes and inter-nodal paths vying for minimum cost routes.

This has led to the more widely accepted use of continuous cost approximations, as shown in Figure 3.1 (b), where the variable cost component is linearly proportional to capacity. It was also proven in [27], that the joint optimal routing and assignment of working capacity does not require the splitting of demand, and can be found by implementing well-established shortest-path routing algorithms. However, these algorithms only consider the capacity routing costs, i.e. the distance dependent costs. An additional minimum-hop constraint is therefore imposed in order to minimise the capacity switching costs (see Section 3.2.3). Using these working capacity assignments, the optimal spare capacity assignments, that ensure 100% restoration against all single link failures, can then be calculated.

Adopting the linear cost approximation, the capacity routed component of the link cost function is given by:

$$f_l^{routed}(wc_l + sc_l) = d_l \cdot f_l^{fixed} + d_l \cdot (wc_l + sc_l) \cdot f^{fibre} \quad (3.5)$$

In this equation, the variable cost component is denoted as  $f^{fibre}$ . Combining equations 3.4 and 3.5, the modified link cost function becomes:

$$f_l(wc_l + sc_l) = d_l \cdot f_l^{fixed} + d_l \cdot (wc_l + sc_l) \cdot f^{fibre} + 2 \cdot (wc_l + sc_l) \cdot f^{terminal}, \text{ if } a_l = 'y' \quad (3.6)$$

The total cost of installing a given network solution is therefore given by:

---

<sup>14</sup>This is due to the fact that fiber cables are generally defined for a set of discrete capacities, which is related to the grade of fiber and the optical equipment used.

$$F(s) = \sum_{\substack{\forall l \in L \\ \text{where } a_l=1}} [d_l \cdot f_l^{\text{fixed}} + d_l \cdot (wc_l + sc_l) \cdot f^{\text{fibre}} + 2 \cdot (wc_l + sc_l) \cdot f^{\text{terminal}}] \cdot a_l \quad (3.7)$$

The objective of the optimisation problem is therefore to find the network solution,  $s \in S$ , which minimises this cost.

### 3.1.5 Reformulation as a Combinatorial Optimisation Problem

A natural decomposition of the optimisation design problem is as follows:

- (i) Given a network configuration, how should capacity be assigned in order to route all inter-nodal demands and satisfy a given level of survivability at the minimum cost?
- (ii) Given any input network, what link configuration represents the optimal solution?

Based on the cost model given in equation 3.7, if sub-problem (i) can be solved in polynomial time, then sub-problem (ii) will consist of a finite, although large, set of feasible solutions. In other words, if the solution space  $S$  can be viewed as discrete with a finite number of possible link configurations, then sub-problem (ii) can be considered a combinatorial optimisation problem [27]. However, even if the problem can be reduced to a combinatorial problem, a large set of feasible solutions must still be considered. Given an  $n$  node network, there are  $n(n-1)/2$  potential links, and the number of potential configurations is of the order  $O(2^{n^2})$ . Since the problem grows exponentially with respect to  $n$ , it is classified as NP-hard, for which no polynomial time solution can be found [27, 30]. Fortunately, a number of heuristic-based algorithms have been developed to solve these problems. The desirable characteristics of these algorithms are that they run in reasonable time, while providing satisfactory near-optimal solutions. The most effective of these algorithms are based on a *local search* methodology. A review of these algorithms, and consequently the solution approach used for solving this optimisation problem, is presented in the Section 3.4.1.

For the above condition to hold, it is assumed that a minimum cost solution can be found for each potential configuration, i.e. sub-problem (i) can be solved. This sub-problem is composed of two sequential solution steps. The first step concerns the joint optimal routing and working capacity assignment problem. Then, based on this solution, the second step concerns the optimal spare capacity assignment problem. As stated in the previous section, the solution to the first step can be found by implementing a minimum-hop/shortest-path routing algorithm. Since this algorithm is of polynomial time complexity, the above condition holds (see Section 3.2.3). An exact solution to the second step can also be found, however its computational complexity increases dramatically with

the number of nodes in any given network, and it turns out that the problem itself is deemed intractable for networks with more than 20 nodes [10]. Considerable work has therefore been devoted to this problem in an attempt to find a feasible solution method. These sub-problems will now be considered in more detail.

## **3.2 Joint Optimal Routing and Working Capacity Assignment Problem**

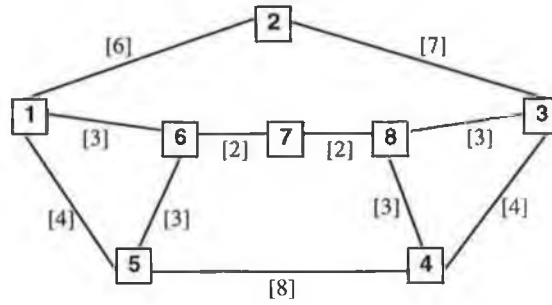
### **3.2.1 Problem Formulation**

The input to the joint optimal routing and working capacity assignment problem is the current network configuration, defined by the adjacency matrix  $A$ . Based on this configuration, the objective is to generate the optimal paths for routing the inter-nodal demand requirements. Paths are selected in accordance with the minimum-hop/shortest path constraint and are stored in the paths matrix  $P$ . Based on these paths, working capacity is then assigned to each link to satisfy the inter-nodal demand requirements.

### **3.2.2 Adoption of a Minimum-Hop Routing Constraint**

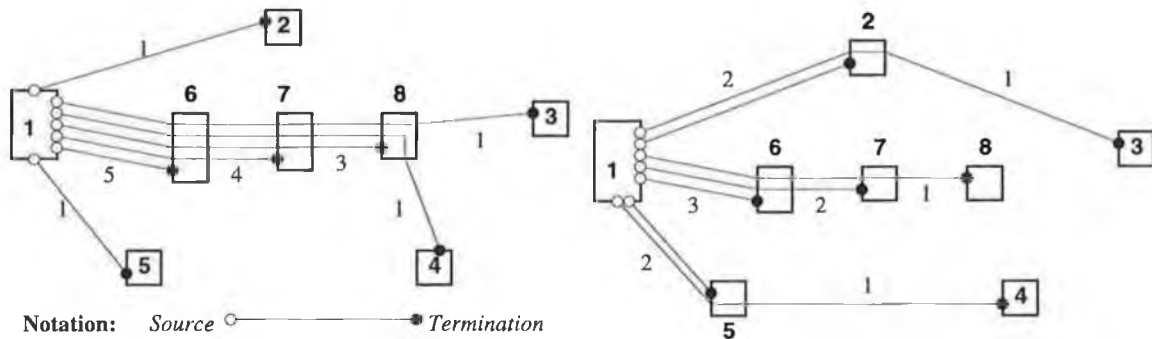
This problem only concerns the minimisation of the working capacity components of the cost function. As stated in section 3.1.2, for a linear cost function with a fixed offset, optimal routes can be found by implementing well-established shortest-path algorithms. However, this does not consider the distance-independent capacity switched component of the cost function and the residual effect on the spare capacity requirement. To minimise the former, a minimum-hop routing constraint is adopted. This minimises the average number of working capacity channels per link and results in a more balanced distribution of capacity. As a consequence, the overall spare capacity requirement is also minimised.

The minimum-hop routing constraint works on the concept of choosing inter-nodal paths with the minimum number of hops, i.e. links. If more than one path of equal hop-count exists, then the path representing the shortest distance would be selected. To illustrate the benefits of minimum-hop routing over standard shortest-path routing, consider the network example shown in Figure 3.2.



**Figure 3.2** 8-node network example (*note: link distances are shown in square brackets*)

As an initial example, both routing techniques are used to find the optimal paths between node one and all other nodes in the network. The solutions are shown in Figure 3.3. For the benefit of this example, all inter-nodal demands are set at unity to indicate the number of paths traversing each link, i.e.  $r_{1,4} = 1$  is routed over  $l_{1,6} \rightarrow l_{6,7} \rightarrow l_{7,8} \rightarrow l_{8,4}$  using the standard shortest-path constraint, and via  $l_{1,5} \rightarrow l_{5,4}$  using the minimum-hop/shortest-path constraint.



(a) Standard shortest-path routing

(b) Minimum-hop/shortest-path routing

**Figure 3.3** Optimal routing from node one to every other node in the network

To compare the cost differences, the capacity routed and switched components will be looked at separately. As all links will be present in the overall solutions, the fixed cost components can be neglected. The objective therefore is to minimise the variable capacity dependent costs. These costs are shown in Table 3.2, where:

$$F_{routed}^c = \sum_{\forall l \in L} a_l \cdot d_l \cdot c_l \cdot f^{fibre}, \text{ denotes the capacity routed costs} \quad (3.8)$$

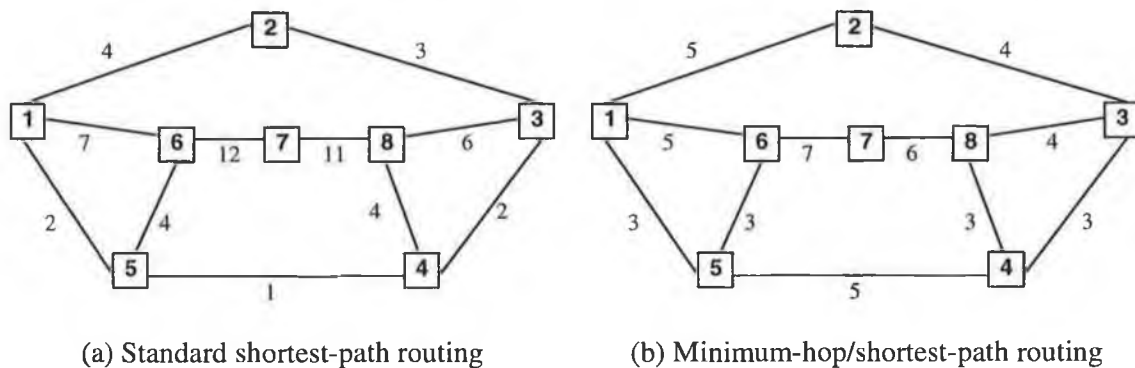
$$F_{switched}^c = \sum_{\forall l \in L} 2 \cdot a_l \cdot c_l \cdot f^{terminal}, \text{ denotes the capacity switched costs} \quad (3.9)$$

$$\bar{c}_l = \frac{\sum_{\forall l \in L} a_l \cdot c_l}{\sum_{\forall l \in L} a_l}, \text{ denotes the average number of capacity channels per link} \quad (3.10)$$

	$F_{routed}^{wc_{i,j}}$	$F_{switched}^{wc_{i,j}}$	$\bar{c}_l$
<b>Standard shortest-path</b>	45. $f^{fibre}$	32. $f^{terminal}$	2.3
<b>Minimum-hop/shortest-path</b>	50. $f^{fibre}$	24. $f^{terminal}$	1.7

**Table 3.2** Working capacity dependent costs resulting from node one

It can be seen from these results that minimum-hop routing minimises the terminal costs while standard shortest-path routing minimises the distance-dependent fibre costs. The solutions also illustrate how the distribution of capacity differs between both routing methods. The shortest-path algorithm selects paths of minimum accumulative link distance, irrespective of the number of individual links (hops) involved. As a result, the average number of working capacity channels increases and certain links and nodes become overloaded. This distribution characteristic is said to be "unbalanced" [18]. Consequently, while there is an apparent cost saving in terms of fibre span costs, the unbalanced distribution increases the terminal switching costs. It also has a residual effect on the overall spare capacity requirement. Since this relates to the overall routing characteristics, consider the respective solutions depicted in Figure 3.4.



**Figure 3.4** Optimal routing and working capacity assignment solutions

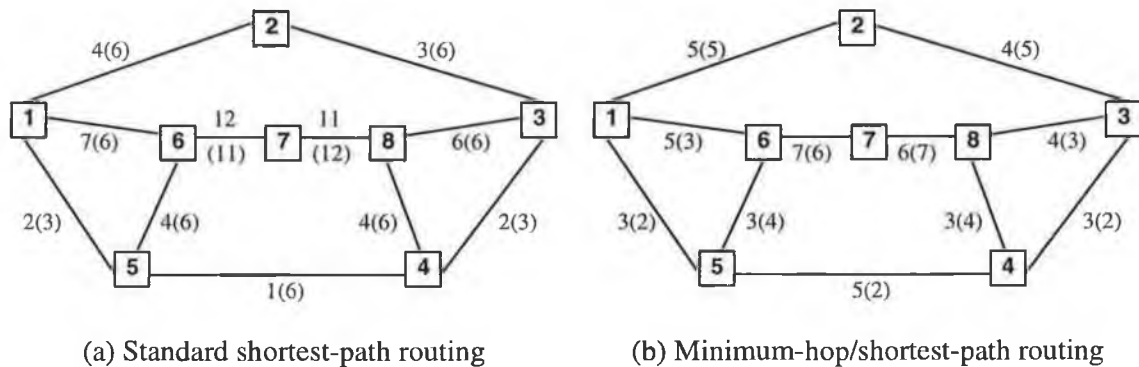
The overall capacity routed and switched costs based on both routing methods are shown in Table 3.3. As expected, the minimum-hop and standard shortest-path algorithms have minimised the terminal and fibre costs, respectively.

	$F_{routed}^{wc}$	$F_{switched}^{wc}$	$\bar{c}_l$
Standard shortest-path	193. $f^{fibre}$	112. $f^{terminal}$	5.1
Minimum-hop/shortest-path	178. $f^{fibre}$	96. $f^{terminal}$	4.4

**Table 3.3** Total working capacity dependent costs

The standard shortest-path algorithm has also resulted in a highly unbalanced distribution of capacity. That is, links  $l_{1,6}$ ,  $l_{6,7}$ ,  $l_{7,8}$ ,  $l_{8,3}$ , and their intermediary nodes are overloaded while those on the periphery, representing longer path distances, remain under-utilised. The residual effect of this on the overall spare capacity requirement can now be considered.

The spare capacity assignment problem and solution methods are presented in Section 3.3. The objective is to take the optimal working capacity solution  $s = (A, WC)$ , and to optimally assign spare capacity that guarantees 100% restoration against all single link failures. Applying this to the solutions above results in the overall capacity assignment solutions shown in Figure 3.5. The spare capacity assignments are shown in parentheses.



**Figure 3.5** Optimised working and spare capacity assignment solutions

The cost differences between both solutions can now be compared in terms of the total network cost, i.e. the sum of working and spare capacity assignments. The overall capacity routed and switched costs for both solutions are given in Table 3.4. For the solution based on the standard shortest-path routing, both the capacity routed and switched costs are now greater than those of the minimum-hop solution. This is attributed to the increase in the average number of working capacity channels and their distribution characteristic.

	$F_{routed}^{wc+sc}$	$F_{switched}^{wc+sc}$	$\bar{c}_l$
<b>Standard shortest-path</b>	426. $f^{fibre}$	244. $f^{terminal}$	11.1
<b>Minimum-hop/shortest-path</b>	358. $f^{fibre}$	182. $f^{terminal}$	8.3

**Table 3.4** Total network capacity dependent costs

In the first instance, shortest-path routing has resulted in a 16% increase in the average number of working capacity channels per link. This is equivalent to the increase in the average number of links (hops) per path, i.e. overloading. If one of these links fails, then each capacity channel traversing that link must be rerouted back over its initial path and then over the longer distance link paths which were neglected in the first place. This in its own right leads to an increase in the average number of spare capacity channels, but of greater consequence is the effect on the optimal utilisation of these assignments.

As stated in Section 2.3.3, spare capacity is a decreasing function of network connectivity since it is assigned in a distributed fashion across the network and is shared by a greater number of possible link failures. In other words, failed capacity, which is itself reduced<sup>15</sup>, can be rerouted over a greater number of diverse paths. Moreover, as the network becomes more balanced, spare capacity is shared more optimally. This is based on the concept of network flow and link cut-sets, which is defined in section 3.3.2. The basic concept is: for each link that fails, the working capacity (flow requirement) must be rerouted over the set of alternative paths (link cut-sets), such that the conservation of flow is satisfied. In other words, the sum of spare capacity (potential flow) on these paths must be greater than or equal to the flow requirement. Optimal flow is a measure of the difference between these two values. In the case of an overloaded link, the flow requirement would represent an upper bound on the flow across certain portions of the network. However, if an under-utilised link fails, there is an excess of flow across these portions of the network, i.e. the flow attributed to the overloaded link is not being optimally utilised by the other link failures. As the network becomes more balanced, the excess flow is decreased and the spare capacity requirement is minimised.

The imposition of the minimum-hop constraint therefore minimises the total capacity routed and switched costs. The above network configuration exemplifies the overall cost benefits of using this routing method. While the standard shortest-path routing method can result in the same solution, this is only possible if the network configuration is such that this routing method selects the same

---

<sup>15</sup>As network connectivity increases, inter-nodal demands are routed over shorter path distances and the average number of working capacity channels decreases.



optimal paths as the minimum-hop routing method. This would be the case in sparse networks, where the shortest-path will generally represent the minimum-hop path. A full analysis of these routing methods under varying network conditions is given in Chapter 6.

### 3.2.3 Routing Algorithm and Computational Complexity

Since the routing algorithm must be applied to each network configuration considered, efficient methods for implementing this algorithm must also be considered. The shortest-path algorithm applied here is due to Dijkstra [30, 36, 37]. It provides the basis for the most efficient algorithms known for solving this problem. The algorithm is based on a labelling criterion and is used to find the shortest-path between any two nodes in a network. Depending on the relative proximity of these nodes, the process can vary from 1 to  $n - 1$  iterations, with a polynomial time complexity of order  $n$ ,  $O(n)$ , for each iteration. Since the process has a maximum run time of  $(n - 1) \cdot O(n)$ , which is approximately  $O(n^2)$ , by setting this as the prerequisite, the shortest-path from any node to all other nodes in the network can be found. Applying this to all  $n$  nodes results in an overall computational time complexity of order  $O(n^3)$ . The minimum-hop constraint can then be introduced without incurring any additional computational overhead. The development and implementation of this algorithm is presented in Chapter 4.

## 3.3 Optimal Spare Capacity Assignment Problem

In order to compare and derive an optimal network solution, the overall optimisation process must be capable of evaluating the minimum cost of providing a required level of survivability for any feasible network configuration. Unfortunately, due to the nature and complexity of this problem, efficient and/or optimal techniques are lacking in this area. One technique that is capable of finding an optimal solution is discussed in [17] and presented in [10]. It uses a linear programming procedure in conjunction with a maximum flow algorithm. However, it is shown that its computational complexity and memory requirement increase dramatically with network size and it is deemed intractable for networks with more than 20 nodes. This has led to the development of an innovative and highly efficient solution method, which, in its basic form, is capable of finding feasible near-optimal network solutions.

### 3.3.1 Problem Formulation

The input to the optimal spare capacity assignment problem is the current network configuration and the optimal working capacity assignment solution denoted as  $s = (A, WC)$ . The paths matrix is

also used to compute the spare capacity requirement based on path-level restoration. Since this introduces additional complexity into the problem, the basic problem formulation in terms of the line-level restoration constraint will first be considered. The objective is to minimise the sum of the spare capacity assignments across the network which guarantee 100% restoration (survivability) against all single link failures.

Based on underlying methodology used in [10], the solution approach is composed of two phases. The first phase involves the generation of optimal flow constraints, which define the optimal paths used to reroute the working capacity on each link failure in a distributed fashion across the network infrastructure. The second phase then concerns the optimisation technique used to minimise the sum of the spare capacity requirements subject to these flow constraints.

### 3.3.2 Generation of Optimal Flow Constraints

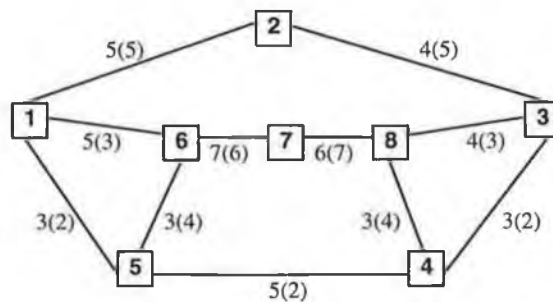
To generate the optimal flow constraints, efficient maximum-flow algorithms, such as Ford and Fulkerson's, could be utilised [30, 36, 37]. The basic concept is to find the optimal path(s) from a source to a terminal along which the potential flow satisfies the current flow requirement. In the case of the spare capacity assignment problem, the source and terminal represent the terminating nodes of a potential link failure; the failed working capacity, the flow requirement; and the spare capacity assignments on each link, the potential flow. The objective therefore is to maximise the utilisation of the potential flow, i.e. minimise its requirement, while satisfying the various flow requirements.

The maximum-flow algorithm that is considered here is based on the preflow-push concept of Karzanov [38]. Considered one of the most efficient techniques, it uses link cut-sets and partial cut-sets to compute the flow across the network. A cut is defined as a set of links in a connected network whose removal disconnects the subset of nodes on either side of the cut. Moreover, the capacity assigned to these links defines the potential flow across that portion of the network. In the case of a partial cut, the working capacity on one of these links represents the flow requirement and the spare capacity on the remaining links represents the potential flow. To reroute the flow requirement, the resulting link flow equation must satisfy the conservation of flow, such that:

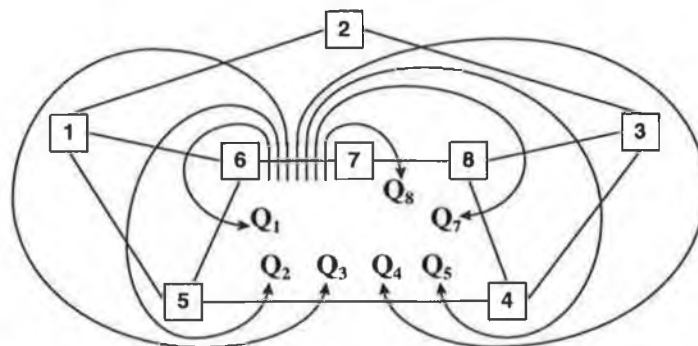
$$\sum_{\substack{\forall a_{i,j} \text{ cut} \\ \text{where } a_{i,j} \neq a_{x,y}}} sc_{i,j} \cdot a_{i,j} \geq wc_{x,y}, \text{ where } l_{x,y} \text{ is the failed link} \quad (3.11)$$

Starting with the source node, the objective is to cut each node in turn from the network and to compute the link flow equation associated with that cut. Using a labelling criterion to record the

status and input flow of each node, as each node is cut, the input flow on that node is pushed across the network towards its adjacently connected nodes. At each iteration, the objective is to calculate which node represents the optimal flow and to cut that node from the network. The process continues until all intermediary nodes between the source and the terminal have been cut. The link flow equations associated with each of these cuts then define the optimal paths used to reroute the flow requirement of the failed link.



**Figure 3.6** Network solution (with optimal working and spare capacity assignments)



**Figure 3.7** Network example (showing the set of partial-cuts attributed to a failure on  $l_{6,7}$ )

To illustrate how this process works, consider the network solution (with optimal working and spare capacity assignments) shown in Figure 3.6. In this example, let link  $l_{6,7}$  represent the failed link. Nodes 6 and 7 therefore represent the source and terminal, and  $wc_{6,7} = 7$  the flow requirement. The flow requirement is initially assigned to the source and all other nodes are labelled as being uncut with a zero input flow assignment. The process is initiated by making the first partial-cut  $Q_1$  around the source node, see Figure 3.7. The objective is to push the input flow assigned to this node over the cut links to its adjacent nodes, i.e. nodes 1 and 5. The link flow equation associated with this cut is given by:

$$sc_{6,1} + sc_{6,5} \geq wc_{6,7} \quad (3.12)$$

Inserting the values given, it can be seen that this equation satisfies the conservation of flow. An input flow corresponding to the potential flow (spare capacity assignment) on each of the cut links is then assigned to the respective adjacent nodes. There are now two possible paths to reroute the flow requirement, where the number of paths is equal to the number of nodes that have an input flow assignment. The objective therefore is to find the node which represents the optimal flow and to cut that node from the network.

Taking node 1, the input flow is currently 3 and the output flow is 7, i.e.  $sc_{1,2} + sc_{1,5}$ . This node is said to have an excess potential of  $7 - 3 = 4$ . Similarly, the excess potential around node 5 is 0. The node which exhibits the minimal excess potential is said to represent the optimal flow, since it maximises the potential flow across that portion of the network. Since this is exhibited by node 5, the next partial cut  $Q_2$  is made around this node and the corresponding link flow equation is given by:

$$sc_{6,1} + sc_{5,1} + sc_{5,4} \geq wc_{6,7} \quad (3.13)$$

In regard to inter-nodal flow, an input flow corresponding to the potential flow on any newly cut link is then added to that already assigned to the respective adjacent nodes. That is, node 1 will now have an accumulative input flow of 5 units, while node 4 will have an initial input flow of 2 units. Since two alternative paths exist, the excess potential at the respective nodes must be calculated. Node 1 now has an excess of 0, due to the new flow conditions around it, while node 4 has an excess of 4. The next partial-cut  $Q_3$  is therefore made around node 1 and the resulting link flow equation is given by:

$$sc_{1,2} + sc_{5,4} \geq wc_{6,7} \quad (3.14)$$

This process is applied to each of the remaining nodes in the network. The order in which these nodes are cut and the corresponding link flow equations are as follows:

$$Q_4 \text{ (node 2): } \quad sc_{2,3} + sc_{5,4} \geq wc_{6,7} \quad (3.15)$$

$$Q_5 \text{ (node 3): } \quad sc_{3,4} + sc_{3,8} + sc_{5,4} \geq wc_{6,7} \quad (3.16)$$

$$Q_6 \text{ (node 4): } \quad sc_{3,8} + sc_{4,8} \geq wc_{6,7} \quad (3.17)$$

$$Q_7 \text{ (node 8): } \quad sc_{8,7} \geq wc_{6,7} \quad (3.18)$$

The objective of rerouting the failed capacity has now been attained and the optimal link flow equations found. Indexing each link as an ordered list<sup>16</sup>, these equations are represented by the cut-set matrix  $Q_x^z = (q_{x,y}^z)$ , where  $z$  is the index of failed link, i.e.  $z = 10$ ,  $x$  is the cut index, i.e.  $x = 1, \dots, n-1$ , and  $y$  is the link index, i.e.  $y = 1, \dots, l^{num}$ , where  $l^{num}$  is the number of links in the solution. Then, for each cut  $x$ ,  $q_{x,y}^z = 1$  if link  $l_y$  is partially cut, and  $q_{x,y}^z = 0$  otherwise. Applying this to the link flow equations results in the following matrix representation:

$$\begin{bmatrix} Q_1^{10} \\ Q_2^{10} \\ Q_3^{10} \\ Q_4^{10} \\ Q_5^{10} \\ Q_6^{10} \\ Q_7^{10} \end{bmatrix} \begin{bmatrix} SC^{ind} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} SC_1^{ind} \\ SC_2^{ind} \\ SC_8^{ind} \\ SC_8^{ind} \\ SC_8^{ind} \\ SC_8^{ind} \\ SC_8^{ind} \\ SC_9^{ind} \\ SC_{10}^{ind} \\ SC_{11}^{ind} \end{bmatrix} \geq \begin{bmatrix} WC_{10}^{ind} \\ WC_{10}^{ind} \\ WC_{10}^{ind} \\ WC_{10}^{ind} \\ WC_{10}^{ind} \\ WC_{10}^{ind} \\ WC_{10}^{ind} \end{bmatrix}$$

It is noted that the spare capacity and working capacity index correspond to the ordered link list. For example, the link flow equation attributed to the partial cut  $Q_4$ , i.e.  $z = 4$  is given by:

$$Q_4^{10} \cdot SC^{ind} = SC_{l_4^{ind}} + SC_{l_7^{ind}} = WC_{l_{10}^{ind}}, \text{ [which is equivalent to the flow equation 3.15]} \quad (3.19)$$

<sup>16</sup>An ordered list is used to efficiently reference the matrix elements. Each link is indexed according to the index of its terminating nodes, i.e.  $l_1^{ind} = l_{1,2}$ ,  $l_2^{ind} = l_{1,5}$ ,  $l_3^{ind} = l_{1,6}$ ,  $l_4^{ind} = l_{2,3}$ ,  $l_5^{ind} = l_{3,4}$ ,  $l_6^{ind} = l_{3,8}$ ,  $l_7^{ind} = l_{4,5}$ ,  $l_8^{ind} = l_{4,8}$ ,  $l_9^{ind} = l_{5,6}$ ,  $l_{10}^{ind} = l_{6,7}$  and  $l_{11}^{ind} = l_{7,8}$ . The alternative would be to use an  $N \times N$  matrix to represent each cut, which would be totally inefficient.

Inserting the corresponding spare capacity assignments, it can be shown that the conservation of flow for each equation is satisfied:

$$Q^{10} . SC^{ind} = \begin{bmatrix} 7 \\ 8 \\ 7 \\ 7 \\ 8 \\ 7 \\ 7 \end{bmatrix}$$

It can be seen from this that the potential flow across the majority of partial-cuts is exactly equal to the capacity flow requirement. These cuts are referred to as being saturated, i.e. they exhibit no excess flow. As discussed in Section 3.2.3, this represents the optimal utilisation of spare capacity across those portions of the network. It has also been shown from this example that the number of flow equations is equal to the number of nodes cut, i.e.  $n-1$ . However, in the case of uniformly distributed flow, where the majority of cuts are saturated, additional flow constraints may be required in order to generate a feasible solution. This is an inevitable limitation of any flow algorithm where the existence of excess flow in one equation might be carried forward and contribute to the saturated flow of another. In terms of computational time complexity, the maximum flow algorithm requires a minimum of  $n-1$  iterations with each iteration having a polynomial time complexity of order  $O(n)$ . Applying this to all  $l^{num}$  links in any given network configuration results in an overall computational time complexity of order  $O(n^2) . l^{num}$ . In other words, the overall time complexity attributed to this phase of the solution method is a function of the number of links (or connectivity characteristic).

In this example, the solution was used to generate the flow constraints. However, in the actual spare capacity assignment problem, the objective would be to find this solution. To generate these flow constraints, efficient maximum flow algorithms, such as the preflow-push algorithm, could be used. However, these algorithms require some form of reference flow. In [10], the working capacity assignments on each link were used. This process works inversely to the process above, i.e. the node exhibiting the highest excess flow is selected at each iteration. However, since the working capacity assignments are not inversely proportional to the optimum spare capacity assignments, this could result in sub-optimal flow constraints. An alternative method developed here is discussed in Section 3.3.4. It introduces an initial spare capacity assignment phase based on the local optimisation of

flow around each node. Using these values, the above process can then be used to generate the optimal flow constraints.

### 3.3.3 Optimisation Phase and Problem Complexity

Assuming that the flow constraints have been generated, the objective of the second phase is to minimise the sum of the spare capacity assignments subject to these constraints. This is stated as follows:

$$\text{Object Function: } \sum_{y=1}^{l^{num}} sc_{l_y^{ind}} \cdot a_{l_y^{ind}} \rightarrow \min \quad (3.20)$$

$$\text{Subject to: } \sum_{y=1}^{l^{num}} q_{x,y}^z \cdot sc_{l_y^{ind}} \geq wc_{l_z^{ind}}, \quad x = 1, \dots, n-1, \quad z = 1, \dots, l^{num} \quad (3.21)$$

This is the general form of a linear programming problem for which efficient solution methods, such as the Simplex Method, have been developed [30, 39]. Unfortunately, since its computational time and memory requirement increase dramatically with network size, it is deemed intractable for networks with more than 20 nodes.

In regard to the memory requirement, a matrix of  $(n-1) \cdot l^{num}$  elements is required for each link failure, which results in an overall requirement<sup>17</sup> of  $l^{num} \cdot (n-1) \cdot l^{num}$ . At this point each flow equation is represented as a linear inequality<sup>18</sup>. To apply a LP technique, these equations must first be transformed into equation form by subtracting a non-negative surplus or slack variable  $s_v$ , where  $v = 1, \dots, (n-1) \cdot l^{num}$ . The objective function would then be subject to:

$$\sum_{y=1}^{l^{num}} q_{x,y}^z \cdot sc_{l_y^{ind}} - s_{z,x} \geq wc_{l_z^{ind}}, \quad x = 1, \dots, n-1, \quad z = 1, \dots, l^{num} \quad (3.22)$$

This introduces an additional matrix (memory) requirement of  $[(n-1) \cdot l^{num}]^2$  elements. Since the order of complexity can be even greater, this technique would therefore be unsuitable for this design problem. An alternative technique is presented in the next section.

---

<sup>17</sup>This does not account for the presence of additional flow constraints.

<sup>18</sup>Prior to the optimisation process the existence of an inequality (excess flow) will not be known, so each equation must be treated as being a potential inequality.

### 3.3.4 Proposed Solution Method

In the above approach, a solution is only found after performing the LP technique on the optimal flow constraints. The solution method proposed here overcomes this problem by generating a solution at each stage of the process. It works by initially assigning spare capacity locally around each node. This highly efficient technique exploits the unique properties of optimal network flow and the distribution of working capacity assignments to approximate a feasible near-optimal solution. Taking these initial assignments as reference values, the maximum flow algorithm is then used to generate the optimal flow constraints for each link failure. During this step, additional spare capacity can be assigned in order to satisfy the conservation of flow for each constraint. This results in a feasible near-optimal solution that can then be optimised using an adaptive minimisation technique. This is an exhaustive process that incurs a relatively large computational overhead and memory requirement. The basic spare capacity generator, which is composed of the first two steps, is therefore used in the overall optimisation process. It is noted that the overall spare capacity generator, including the minimisation step, is also offered as a stand-alone process to evaluate exact survivability costs. These solution steps will now be described in more detail.

#### 3.3.4.1 Initial spare capacity assignment algorithm

The input to this process is the current network configuration and the optimal working capacity assignment solution  $s = (A, WC)$ . The objective is to optimally assign spare capacity around each node. Although the process is localised, since each link is present in two separate nodal processes, interaction between flow assignments will exist. Intuitively, by using this property, the failed capacity on any link (flow requirement) can be distributed (pushed) to some degree across the network from one process (node) to the next. These assignments would then provide the near-optimal reference flows required for the generation of the optimal flow constraints.

It was found that the assignment of optimal flow is dependent on the order in which nodes are selected. Since the distribution of spare capacity is a function of network connectivity and the distribution of working capacity (see Section 3.2.3), the proposed selection criterion works by evaluating the potential flow around each node in accordance with the following equation:

$$n_i^{potential\_flow} = \frac{wc_i^{max}}{n_i^{con} - 1} \quad (3.23)$$

In this equation,  $wc_i^{max}$  denotes the maximum link capacity terminating at node  $i$ , and  $n_i^{con}$  the connectivity of that node, i.e. number of terminating links. The objective is to move from regions (nodes) of high potential, which represent the upper bounds on the spare capacity requirement, to

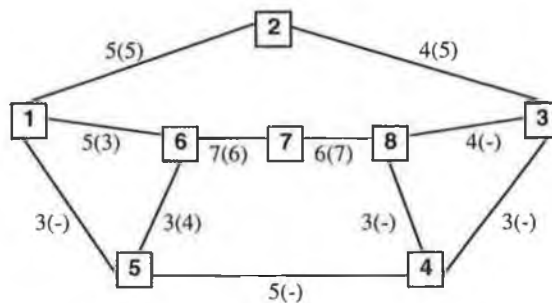


those of lower potential and thus push the flow requirements to some degree across the network. Nodes are therefore sorted according to the maximum value, to specify the order in which they are selected.

Having determined this, the initial spare capacity assignment process is applied to each node in turn. The objective is to cut each link around the node and to assign spare capacity to the remaining links, such that the corresponding flow requirement is optimally rerouted. However, since interaction between processes exists, the spare capacity assignments from previous processes must also be taken into account. The first step therefore is to ascertain the flow requirement on each link according to the following constraint:

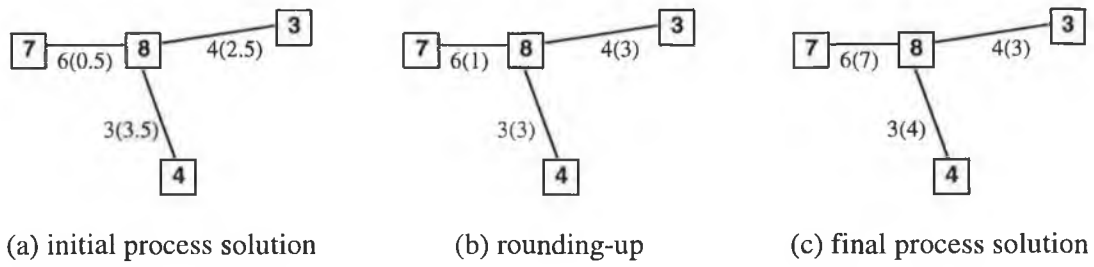
$$f_j^{req} = \max\{wc_{i,j}, sc_{i,j}\} \quad (3.24)$$

In this equation,  $i$  represents the current node and  $j$  any adjacent node. To illustrate this concept, consider the network solution shown in Figure 3.8. In this solution, spare capacity from three previous nodal processes has been assigned, i.e. around nodes 7, 2, and 6. According to equation 3.22, the next process is to be applied to node 8. Looking at the working and spare capacity assignments on each of the terminating links, it can be seen that the spare capacity on link  $l_{8,7}$  represents the upper bound on its flow requirement, i.e.  $f_7^{req} = \max\{wc_{8,7}, sc_{8,7}\} = 7$ . This illustrates the *push* concept, with the spare capacity requirement attributed to the failure on link  $l_{7,6}$  now being rerouted across another region of the network.



**Figure 3.8** Initial spare capacity assignments around nodes 7, 2, and 6

Having determined the flow requirements on each link, each of these links must be cut in turn and spare capacity assigned over the remaining links, such that the corresponding flow requirement is optimally rerouted (pushed). The method used here is to assign spare capacity in proportion to the working capacity assignment on each of the remaining links, such that the sum of the total capacity (working and spare) is balanced. To illustrate this concept consider the spare capacity assignment process around node 8, as shown in Figure 3.9.



**Figure 3.9** Spare capacity assignment process around node 8

Neglecting previous spare capacity assignments, the initial solution is shown in Figure 3.9 (a). It can be seen that the total capacity on each link is exactly equal to 6.5 units. Since we are dealing with integral line rates, these values must be rounded to the nearest integer. If, as in this case, the excess is evenly distributed, the general rule is to round up on links that represent the upper bound. The solution to this is shown in Figure 3.9 (b). However, these are regarded as tentative assignments, since the spare capacity assignments from previous iterations must be considered. As each link is cut, the following constraint is therefore applied:

$$sc_j^\phi = \max\{sc_{i,j}, sc_j^\phi\} \quad (3.25)$$

That is, the spare capacity assignment on any link is only updated if the current requirement represents the upper bound. If the spare capacity assignment exceeds the current requirement, then this excess flow can be used to reduce the requirements on the other links. Any link that satisfies this condition is therefore removed from the process, the excess flow is subtracted from the flow requirement, and the process is applied again. To illustrate this concept, consider the tentative spare capacity assignments attributed to the cut on link  $l_{8,3}$ :

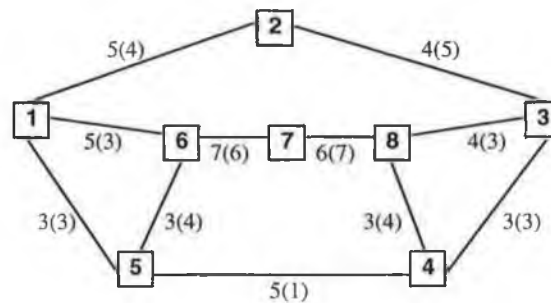
$$sc_4^\phi = \max\{sc_{8,4}, sc_4^\phi\} = \max\{0, 3\} = 3 \quad (3.26)$$

$$sc_7^\phi = \max\{sc_7, sc_{8,7}^\phi\} = \max\{7, 1\} = 7 \quad (3.27)$$

In this case, since the spare capacity requirement on  $l_{8,7}$  exceeds the current requirement, this link is removed from the process and the excess flow is subtracted from the flow requirement. As the flow requirement is now  $\leq 0$ , the process terminates and the spare capacity assignments are left unchanged, i.e.  $sc_{8,4} = 0$  and  $sc_{8,7} = 7$ . Spare capacity is then assigned for each of the other cut links. The solution is shown in Figure 3.9 (c).

The solution after applying the initial spare capacity assignment process to all 8 nodes is shown in Figure 3.10 below. We can see that it is a good approximation of the optimal network solution shown previously in Figure 3.6. These assignments now provide the near-optimal reference flows

required for the next solution step. In terms of computational complexity, each nodal process has a polynomial time complexity of  $O(n^2)$ , which results in an overall time complexity for this solution step of  $O(n^3)$ .



**Figure 3.10** Initial spare capacity assignment solution

### 3.3.4.2 Global spare capacity assignment algorithm

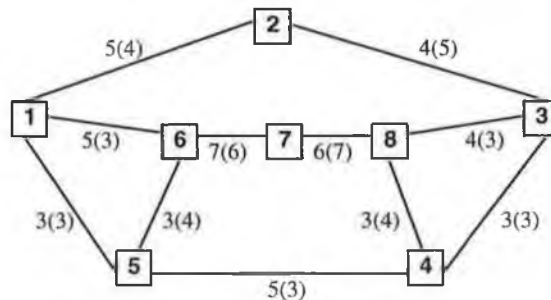
While the methodology used in the previous step was capable of rerouting (pushing) the flow requirements to some degree across the network, as these flow requirements get dissipated (split) at each node, the conservation of flow across the entire network may not be satisfied. This phase therefore concerns the near-optimal assignment of spare capacity across the entire network through the generation of optimal flow constraints.

The near-optimal reference flow from the initial spare capacity assignment phase provides the input to this phase. Using these reference values, the optimal flow constraints for each possible link failure can then be generated using the maximum flow algorithm presented in Section 3.3.2. Since the working capacity assignment on certain links represents the upper bound on the flow across regions of the network, links are initially sorted according to the maximum value. This then specifies the order in which the maximum flow algorithm is applied to each link.

As the conservation of flow across certain regions of the network may not be satisfied, each time a partial cut is made, the potential flow across that cut must be evaluated. If this is not satisfied, then additional spare capacity is assigned to the relevant links in accordance with the method of proportionality discussed in the previous section. It is noted that, for the basic spare capacity assignment algorithm it is not necessary to generate the cut-set matrix  $Q$ , since these constraints are only used in the adaptive minimisation step.

Applying this to our 8 node network results in the solution shown in Figure 3.11. Comparing this to the optimal network solution, Figure 3.6, we can see that the output of the basic spare capacity generator represents a feasible near-optimal solution. It is this joint process that will be used in the

overall solution method. In this case, since the minimisation step is not considered, the flow constraints do not need to be stored, i.e. memory requirement is reduced.

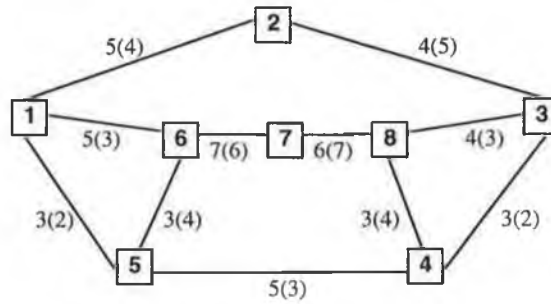


**Figure 3.11** Global spare capacity assignment solution

### 3.3.4.3 Adaptive minimisation algorithm

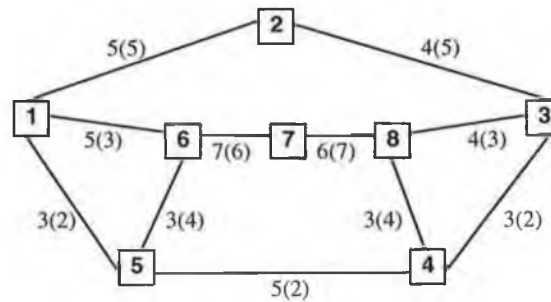
This algorithm, which is offered as a stand-alone process, is used to minimise the sum of the spare capacity assignments subject to the optimal flow constraints. Taking the feasible near-optimal solution from the previous phase, this adaptive technique works on the principle of increasing the spare capacity requirement over certain links in order to reduce the requirement over a greater number of links while still satisfying flow constraints. The criterion used to select which links are increased and which are decreased stems from an intuitive study of the optimal flow constraints.

In Section 3.3.2, it was shown that the potential flow across certain cuts were saturated and that these cuts represented the optimal utilisation of spare capacity across these portions of the network. Using this property, the first step is to analyse each of the flow constraints in order to evaluate the utilisation characteristics of each link, i.e. how many saturated cuts does each link contribute to. The objective is to increase the spare capacity requirement over links that are highly utilised and decrease the spare capacity requirement on links that are under utilised. Initially only improvements (reductions) in the overall spare capacity requirement are accepted; that is, if the increase over a certain number of links results in the reduction over a greater number of links. Using this adaptive technique, a solution is finally found, where no additional improvements can be made. Applying this to the 8 node network results in the optimal solution shown in Figure 3.12 below.



**Figure 3.12** Adaptive minimisation (of capacity switched costs)

We have - in effect - minimised the total capacity switched costs of the network. The next step is to minimise the capacity routed costs. Taking each link in turn, we find the subset of links that appear in all of the saturated cuts that this link contributes to. The link that represents the shortest distance is then compared to the current link distance, and if this is found to be shorter, then its requirement is increased while the requirement on the current link is decreased. This sub-routine continues until no additional improvements can be made. Applying this to the 8 node network results in the solution in Figure 3.13. Comparing this to the previous solution, we can see that  $sc_{4,5}$  ( $d_{4,5} = 8$ ) has been decreased and  $sc_{1,2}$  ( $d_{1,2} = 6$ ) increased. Since the LP technique does not consider distance as a constraint, this solution method is therefore capable of finding even lower cost solutions.



**Figure 3.13** Adaptive minimisation (of capacity routed costs)

### 3.3.5 Spare Capacity Assignment based on Path-Level Restoration

As defined in Section 2.3.2, in path-level restoration working capacity is rerouted between the terminating nodes of each inter-nodal path using the failed link. The result is a reduction in the overall spare capacity requirement, which can be reduced even further by releasing and making available the working capacity on all paths using the failed link.

### 3.3.5.1 Initial spare capacity assignment algorithm

To assign spare capacity based on path-level restoration, a path-trace algorithm is applied to the paths matrix  $P$  to find all inter-nodal paths using the failed link. For each path found, a second path-trace sub-routine is then used to trace along the inter-nodal path and release the working capacity on each link found (see Chapter 4, Section 4.4.4).

Taking the 8 node network as an example, the initial spare capacity assignment around node 7 is again considered. As before, the flow requirement on each link emanating from this node is evaluated according to equation 3.22. Then, taking each link in turn, we reroute the flow requirement over the remaining links. At this point, we introduce the release phase. All inter-nodal paths using that link are found and the working capacity along these paths is traced and released. A released capacity matrix  $RC = (rc_{i,j})_{i,j \in N}$  is used to store the aggregated working capacity released on each link. Taking link  $l_{6,7}$  as an example, a graphical representation of this is depicted in Figure 3.14. The released capacity on each of these links is stored in the relevant matrix element, i.e.  $rc_{7,8} = 3$ .

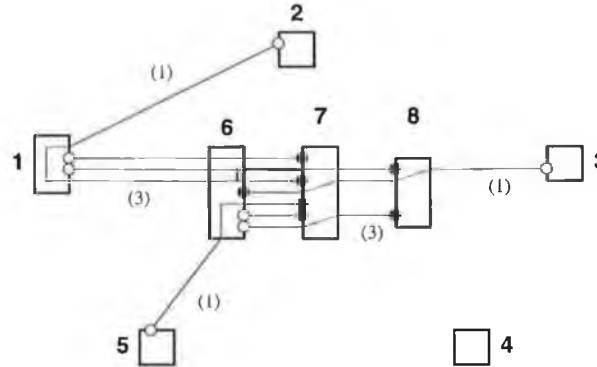


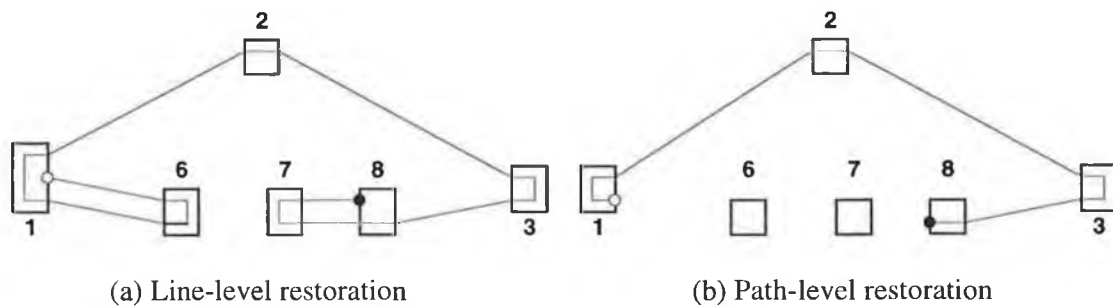
Figure 3.14 Released capacity illustration

We are only interested in the released capacity on each of the remaining links in the nodal process, i.e.  $rc_{7,8} = 3$ . Since the flow requirement on the failed link can now be off-set by the released capacity, a new flow requirement is calculated according to the following constraint:

$$f_j^{req} = f_j^{req} - 2 \cdot \sum_{\forall k \neq j} rc_{i,k} \cdot a_{i,k} \quad (3.28)$$

In this equation,  $i$  represents the current node, i.e. node 7,  $j$  the terminating node of the current link failure, i.e. node 6, and  $k$  any other node adjacent to  $i$ , i.e. node 8. In the case of link  $l_{6,7}$ , the new flow requirement is 1 unit, i.e.  $f_6^{req} = f_6^{req} - 2 \cdot rc_{7,8} = 7 - 6 = 1$ . The remainder of the process is then the same as that defined in Section 3.3.4.1.

Since the original flow requirement has been reduced quite considerably, it is appropriate at this point to verify equation 3.28. Taking the restoration of the inter-nodal path between nodes 1 and 8 as an example, the difference between line and path-level restoration is shown in Figures 3.15 (a) and (b).



**Figure 3.15** Comparison of line and path-level restoration

In the case of line-level restoration, the demand is routed over the original path towards the source node of the failed link, i.e. node 6. The source node initiates the restoration process and reroutes the demand to the terminal node, i.e. node 7. When it reaches the terminal it is routed over its original path again. The restoration path in this case is:  $l_{6,1} \rightarrow l_{1,2} \rightarrow l_{2,3} \rightarrow l_{3,8} \rightarrow l_{8,7}$ .

In the case of path-level restoration, the source node of the failed path, i.e. node 1, initiates the process and reroutes the demand requirement to the corresponding terminal node, i.e. node 8. The restoration path in this case is:  $l_{1,2} \rightarrow l_{2,3} \rightarrow l_{3,8}$ . Since two links have been removed from the restoration path, the corresponding spare capacity requirement on these links is reduced. In addition, the working capacity on the original path is released and made available. This is equivalent to subtracting twice the released capacity from the flow requirement on each of these links as given by equation 3.28.

### 3.3.5.2 Global spare capacity assignment algorithm

As before, the solution from the previous step provides the reference flow values. The order in which the maximum flow algorithm is applied is also the same. However, to generate the flow constraints based on path-level restoration, a number of modifications to this process must be made.

Taking each link in turn, the first step is to apply the release phase. As was the case in the previous process, all inter-nodal paths using that failed link are found and the working capacity along each path is traced and released. The maximum flow algorithm is now applied. As before, the objective is to cut one node at a time from the network and to compute the link flow equation associated with that partial-cut. In the case of path-level restoration, the released capacity on each link contributing

to that flow equation can now be used to reduce the flow requirement, i.e. the failed working capacity. Based on the principles applied in the previous section, the resulting link flow equation must now satisfy the following conservation of flow:

$$\sum_{\substack{\forall a_{i,j} \text{ cut} \\ a_{i,j} \neq a_{x,y}}} sc_{i,j} \cdot a_{i,j} \geq wc_{x,y} - \sum_{\substack{\forall a_{i,j} \text{ cut} \\ a_{i,j} \neq a_{x,y}}} 2 \cdot rc_{i,j} \cdot a_{i,j}, \text{ where } l_{x,y} \text{ denotes the failed link} \quad (3.29)$$

As was the case with the original process, each time a partial-cut is made the potential flow across this link cut-set is compared to the flow requirement. If it is found not to satisfy the conservation of flow, then additional spare capacity is assigned to these links according to the method of proportionality.

### 3.3.5.3 Overall complexity

The adaptive minimisation technique can then be applied to this solution as before. Applying the new three-step process to our 8 node network results in the solution shown in Figure 3.16.

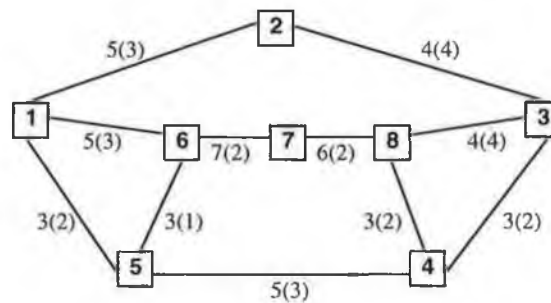


Figure 3.16 Path-level Restoration

Comparing this to the solution based on line-level restoration (see Figure 3.6), it was found that there is a 37 % reduction in the sum of the overall spare capacity requirement:

$$\text{Line-level Restoration: } \sum_{\forall l \in L} sc_l \cdot a_l = 43, \text{ Path-level Restoration: } \sum_{\forall l \in L} sc_l \cdot a_l = 27$$

As discussed in Chapter 2, while this reduction can be translated into a reduction in the total capacity routed and switched costs, it must offset the additional processing costs (embedded in each node) needed to coordinate this more complex restoration process. In a similar way, the evaluation of spare capacity based on path-level restoration incurs additional computational complexity that may be deemed too time consuming for the overall optimisation process.



## 3.4 Network Optimisation Solution Method

### 3.4.1 Review of Solution Methods

An analysis of the most effective solution methods, both established and recently developed, for solving combinatorial optimisation problems was conducted by McGibney [27]. While that work concerned the general concepts of network optimisation, not considering the survivability constraint or explicitly defining the transmission infrastructure, its findings have some bearing on the selection of the overall solution method adopted in this work. Since all methods under consideration are based on a local search methodology, a general introduction to this and related issues is presented.

#### 3.4.1.1 Local search methodology

As defined in [27], "local search (also called neighbourhood search) is a general approach to solving optimisation problems where there is a finite, but large, set of feasible solutions". This approach is therefore well suited to the present combinatorial optimisation problem. Given an initial starting solution, the idea is to generate a set (neighbourhood) of feasible solutions that are, in some sense, near to the current solution. A new solution is then selected from this set in some manner, i.e. the maximum cost improvement. The selection process is termed a *move* and the local search routine continues making a series of moves from the initial solution until some stop criterion is satisfied. That is, it makes a locally optimal choice at each step in the hope that this will lead to a globally optimal solution. As stated in [27], "the number of local optima and their proximity (in terms of cost) to the global optimum influences the relative effectiveness of these algorithms". Each algorithm can be defined according to the following procedure constraints:

- choice of initial starting solution
- definition of a neighbourhood
- selection criteria

The choice of initial starting solution will depend on the local search method. In the definition of neighbourhoods, a clear trade-off exists between computational time and quality of local optima. Larger neighbourhoods might mean that a better solution could be found, but at the expense of increased computational complexity.

For all solution methods under consideration, the neighbourhood of a solution is defined as the set of solutions differing in link arrangement from the current one by the *presence* or *absence* of one link. Selecting a solution from this neighbourhood implies that a *move* consists of a link *insertion* or

*removal*. The criteria for selecting a new solution from this neighbourhood is then determined by the algorithm.

### 3.4.1.2 Greedy algorithms

For *Greedy* algorithms, the choice of initial starting solution is shown to be that of a fully connected network. In its pure form, a greedy algorithm always selects the move that represents the best cost improvement at that moment. In other words, it opts for the path of steepest descent from the initial solution and continues in this fashion until a local optima is reached. However, this is relatively time-consuming, as the cost of every solution in the neighbourhood must be evaluated at every iteration, i.e.  $n(n-1)/2$  new solutions. The alternative is to use Minoux's *Accelerated Greedy* algorithm [40]. This highly efficient algorithm takes advantage of a *monotonicity* property which requires a small number (often just one) of the all solutions to be evaluated at each iteration.

The algorithm, which only holds for link removals, works on the principle that given an initial starting solution, the set of all potential link removal solutions is evaluated. These solutions are then sorted according to the maximum cost improvement, and the best solution is then selected. Based on the new solution, the monotonicity property holds that, if the sub-set of link removal solutions is again evaluated, then their respective costs will never be less than that of the previous iteration. Therefore, instead of evaluating this new solution set, the link removal solution that represented the next best cost improvement in the original set is only evaluated. In accordance with the monotonicity property, if its cost is unchanged, or is at least lower than the next best solution, then it is removed, since no other link removal will result in a better solution. On the other hand, if its cost has changed and it is no longer better than the next best solution, then it is inserted back into the solution set and resorted. The process continues until no further cost improvements can be found.

### 3.4.1.3 Modern methods

More recently, a number of advanced solution methods have been developed. These algorithms are characterised by their ability to avoid getting trapped in local optima. The advantage of this is that better solutions can be found than with pure greedy methods. However, in comparison to the greedy approach, more computational time is required for these algorithms to converge. *Tabu Search* and *Simulated Annealing* are amongst the most promising of these algorithms.

Tabu Search is essentially a greedy algorithm, with the exception that this approach can accept the least-cost non-improving move in the event that no improving moves are possible. In other words, it

avoids getting trapped in local optima by adopting a steepest decent/mildest ascent procedure. Since it is very likely that the process will revert back to the local optima at the next move, in order to guide the process out of this, reverse moves are temporarily forbidden (Tabu). The ability of this algorithm to find a better solution depends on the size of the Tabu list, i.e. the list of most recent moves, and the number of additional iterations (non-improving moves) it is allowed to make.

Simulated Annealing is another newly developed algorithm. It is based on the analogy between the process of annealing solids and the problem of solving large combinatorial problems. In the physical annealing process, a material is initially melted and is then slowly cooled so that the particles in its crystallised form can arrange themselves in a state of minimum energy. If the material is cooled too fast, then locally sub-optimal configurations will be introduced into the crystallised material. In the simulated annealing algorithm, the set of feasible solutions are analogous to the various cooling stages of the physical system. At each iteration of the process, a random link is selected. Depending on its current status, its cost of removal or insertion is calculated. The new solution can be accepted even if it represents a non-improving move. The degree to which a non-improving move is accepted relates to the decreasing cooling schedule. The algorithm terminates when non-improving moves are no longer accepted and no improving moves are possible. Generally, the longer the algorithm is allowed to run, i.e. the longer the cooling schedule, the better the solution generated.

#### **3.4.1.4 Comparison of solution methods**

Based on results obtained from [27], Table 3.5 shows a comparison of the various local search algorithms in terms of the quality of solution. A 20 node network with 3 variations on the ratio of fixed to variable cost components were selected for this case study.

Algorithm	Cost		
	$k_{char} = 0.1$	$k_{char} = 1.0$	$k_{char} = 10.0$
Accelerated Greedy	X	X	1.008 X
Pure Greedy	X	X	1.010 X
Tabu Search	X	X	1.004 X
Simulated Annealing	X	X	X

$k_{char} = 0.1$ : fixed costs are on average 10 times less than variable costs

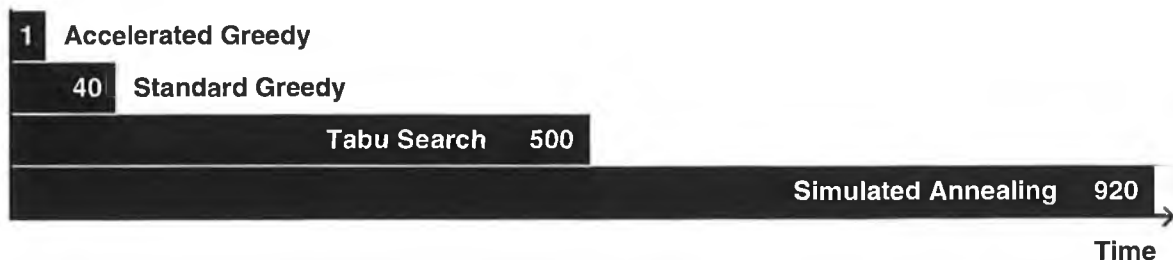
$k_{char} = 1.0$ : fixed costs are the same as the variable costs

$k_{char} = 10.0$ : fixed costs are on average 10 times greater than variable costs

X: the best cost solution for that ratio characteristic

**Table 3.5** Comparison of best cost solutions

It can be seen from these results that all solution methods tend toward the same overall cost. This is a characteristic of many design problems where the global optima are relatively flat and all local optima lead to about the same cost [32]. If this is the case, then the efficiency of these algorithms is of more relevance. A comparison of the average computational time requirements for each algorithm is shown in Figure 3.17 below.



**Figure 3.17** Comparison of computational time requirements

The results indicate that the relative cost reductions resulting from the modern solution methods may not justify the considerable increase in computational time required for these algorithms to converge. Moreover, survivability was not considered in the above design problem, the effect of which would have resulted in even greater convergence times. Due the additional complexity attributed to this constraint, some form of greedy approach, preferably Accelerated Greedy, would be best adopted. However, it was found that the conditions needed for these algorithms to work effectively and efficiently did not hold for the present problem. In particular, it was observed that the solution space consisted of a large number of locally sub-optimal solutions, and that these algorithms tended to get trapped in these local optima. As a result, modern solution methods, such

as Tabu Search, had to be considered in order to refine this solution. This has led to the adoption of a two-phase solution approach, which is presented in the next section.

### **3.4.2 Proposed Solution Method**

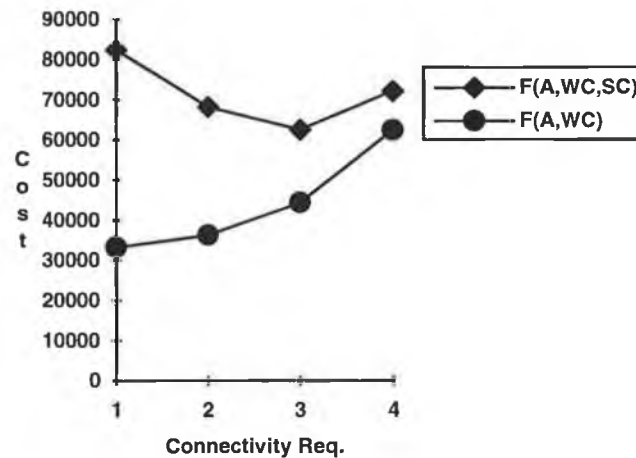
As with all network design problems, there is a trade-off between design time and the quality of solution returned. As discussed in the previous section, the proposed solution method attempts to address this issue by adopting a two-phase solution approach. The first phase concerns the initial optimisation of the problem. Based on a modified Accelerated Greedy algorithm, this phase is used to reduce the initial complexity of the problem by finding a more practical starting solution. The second phase then concerns the refinement of this solution and is based on the more computationally intensive Tabu Search algorithm. An overview of each phase is presented in the following subsections.

#### **3.4.2.1 Initial optimisation phase**

Taking some form of initial network configuration as its input, the objective of this phase is to find a more practical starting solution, which will then be fed into the refinement phase. Since computational efficiency is the main concern, the solution method is based on the Accelerated Greedy algorithm. As stated in the previous section, for greedy algorithms the choice of initial starting solution was found to be that of a fully connected network. This is particularly true of the Accelerated Greedy algorithm, which only considers link removals. However, the imposition of the spare capacity assignment problem has an adverse effect on the efficiency of this algorithm. For instance, since the time complexity attributed to this sub-problem is a function of network connectivity, starting with a fully connected network would prove totally inefficient. Of more significance is the fact that the monotonicity property no longer holds, and a much greater number of solutions must be evaluated at each iteration. To overcome these problems, a modified technique has been developed which takes advantage of the correlation between network connectivity and the various cost characteristics.

As discussed in Section 3.1.3, as network connectivity decreases, the fixed costs component will decrease while the capacity dependent variable costs will increase. This is illustrated in Figure 3.18, which shows the solution cost characteristics for a range of benchmark topologies. Each benchmark topology represents the minimum cost solution (in terms of working capacity costs) that satisfies a given connectivity requirement, i.e. in the range 1 - 4. The total network cost solution (in terms of working and spare capacity costs) is also shown for each topology. It can be seen that the working capacity solution cost is a decreasing function of network connectivity. As the results in Chapter 6

will verify, depending on the input network constraints, the average connectivity requirement of the local optima for this optimisation problem will range between 1.0 and 3.0. The results will also show that the local optima for the total network cost solution will range between 2.5 and 4.5. As can be seen from this example, the total network solution cost for each benchmark topology provides a good indication as to the approximate location of the local optima. It is this concept that is adopted for the modified algorithm.



**Figure 3.18** Solution costs for the respective benchmark topologies

Starting with a fully connected network and only considering working capacity solution costs, the Accelerated Greedy algorithm is applied as previously described, with the exception that, for each potential cost improvement, i.e. link removal, a special connectivity-test algorithm is used to verify that the resulting configuration satisfies an initial connectivity requirement of four<sup>19</sup>. The link removal (move) is only accepted if this connectivity constraint is satisfied and the process continues until no further links can be removed. The resulting solution represents the first benchmark topology and the total network solution cost is calculated. The connectivity requirement is then decremented by one and the next benchmark topology is found. This continues until all benchmark topologies and their corresponding cost solutions have been found. The benchmark topology that represents the minimum reference cost is then used as the input to the refinement phase.

<sup>19</sup>This represents the upper bound for the connectivity characteristic of any local optima (see Chapter 6).

### **3.4.2.2 Refinement phase**

Taking the solution resulting from the previous phase as its input, the objective of this phase is to find the minimum cost solution. As stated above, the solution method adopted for this phase is based on the more computationally intensive Tabu Search algorithm. Since both the working and spare capacity assignment solutions are evaluated for each potential configuration, this phase is also known as the dual optimisation phase. As discussed in the previous section, the ability of this algorithm to find a better cost solution depends on the size of the Tabu list and the number of additional iterations (non-improving moves) it is allowed to make. The design tool also facilitates user interaction with the refinement phase (see Chapter 4). Once the local optimal has been found, the designer can manually edit the corresponding solution by inserting or removing any number of links. The dual optimisation algorithm can then be applied to this new configuration to see if an improved solution can be found. This user interaction procedure forms the basis for the highly effective perturbation technique used to accelerate the process. The implementation of this and all other solution methods described in this chapter are presented in the next chapter.

---

## Chapter 4 - Implementation of Design Procedure

---

Chapter 3 concerned the formulation of the optimal design problem and the development of the overall solution method. As discussed, due to the cost model adopted, the problem could be reformulated as a combinatorial optimisation problem. This enabled the natural decomposition of the problem into a number of sub-problems for which efficient solution methods were proposed and described. In this chapter, the implementation of these individual solution methods is considered in terms of the algorithmic procedures involved and how these form part of the overall optimal design procedure facilitated by the design tool.

### 4.1 Optimal Design Procedure

While the *NetOpt* design tool itself will be presented in the next chapter, it is appropriate at this stage to give a general overview of the optimal design procedure facilitated by the application.

The first step in any design procedure is to initialise the input network to be optimised. This is facilitated by an interactive procedure which enables the network designer to input the following constraints:

- Number of nodes  $n$ : As with all input constraints, it is assumed that the number of nodes will be known in advance and therefore given as an initial input parameter. This constraint will define the dimensions of the various data structures used to represent both the input network and output solution.
- Node locations: The application has full mouse support which, in conjunction with various display features, facilitates the placement of each node within a virtual design field. The  $(x_i, y_i)$  coordinates of each node location are then stored in the respective nodal arrays,  $X = (x_i)_{i=1, \dots, n}$  and  $Y = (y_i)_{i=1, \dots, n}$ . These are used to display the network solution.
- Inter-nodal constraints: Once the nodal infrastructure has been defined the user must then initialise the inter-nodal constraints:
  - i) Fixed costs,  $F^{fixed} = (f_{i,j}^{fixed})_{i,j=1, \dots, n}$
  - ii) Demand requirements,  $R = (r_{i,j})_{i,j=1, \dots, n}$
  - iii) Distances,  $D = (d_{i,j})_{i,j=1, \dots, n}$



- **Cost components:** Once the inter-nodal constraints have been initialised the designer must then specify the network cost components:  $f^{fibre}$  and  $f^{terminal}$

With the input network initialised<sup>20</sup> the network optimisation process can be applied. As defined in Section 3.4.2, this solution method is composed of two phases. The first phase concerns the initial optimisation of the problem. Based on a modified Accelerated Greedy algorithm, this phase reduces the initial complexity of the problem by finding a more practical starting solution. The second phase concerns the refinement of this solution and is based on the more computationally intensive Tabu Search algorithm. The main objective of this phase is to guide the process toward the global optima. As mentioned in the previous chapter, the user interaction procedure facilitated by the design tool also enables the network designer to manually guide the process toward the global optimal solution. As a result, the computational time complexity of the Tabu Search algorithm can be reduced and its effectiveness improved substantially (see Section 4.2.2).

While this describes the basic design procedure, it must be appreciated that any realistic design procedure can iterate through any number of cycles, which include the editing and refinement of the network constraints and the optimal solution states. With this in mind, the remainder of this chapter concerns the implementation of the algorithms used to realise the optimal solution.

## 4.2 Network Optimisation Algorithm

The NETWORK OPTIMISATION ALGORITHM is initially defined in terms of the input constraints, the output solution, and the objective function used to realise the solution. For consistency, this format is applied to the individual solution steps described in subsequent sections. Defining the algorithm we have:

**Input:** A fully connected network configuration defined by the adjacency matrix  $A$ , and the design constraints defined by the network designer. These include the optimal routing strategy and restoration scheme employed by the relevant sub-processes. The default settings are minimum-hop/shortest-path routing and line-level restoration.

**Process Constraints:** In regard to the refinement phase, the network designer must specify the Tabu list size and the number of additional non-improving moves it is allowed to make.

---

<sup>20</sup>As will be discussed in Chapter 5, the application offers various utilities which enable the network designer to access and edit the network constraints at any point during the design procedure.

**Objective:** To find the optimal cost solution.

**Output:** The optimal cost solution.

**Process Variables:** Let  $s_{\min}^{cost}$  denote the minimum solution cost,  $A^{\min} = (a_{i,j}^{\min})_{i,j \in N}$  the solution configuration, and  $c_{req}$  the connectivity requirement, respectively. With this notation in mind, the algorithm can now be stated:

---

**ALGORITHM 4.1: NETWORK OPTIMISATION ALGORITHM**

*Step 1.* Initialise  $s_{\min}^{cost} = \infty$  and  $c_{req} = 4$ .

*Step 2.* Apply the INITIAL OPTIMISATION ALGORITHM (see Section 4.2.1) to the current network to generate the minimum cost solution,  $s = (A, WC)$ , that satisfies  $c_{req}$ .

(a) Apply the TOTAL NETWORK COST ALGORITHM (see Section 4.2.3.2) to this benchmark topology to calculate the overall solution cost,  $s^{cost} = F(A, WC, SC)$ .

(b) If  $s^{cost} \leq s_{\min}^{cost}$ , then this benchmark topology represents the minimum reference solution cost. Set  $s_{\min}^{cost} = s^{cost}$  and  $A^{\min} = A$ .

i) if  $c_{req} > 1$ , then the next benchmark topology must be checked. Let  $c_{req} \rightarrow c_{req} - 1$  and return to *Step 2*.

ii) otherwise, no further reduction in network connectivity can be made. Go to *Step 3*.

(c) Else, the previous benchmark topology represented the minimum reference solution cost. Reset  $A = A^{\min}$  and go to *Step 3*.

*Step 3.* The initial network configuration has therefore been reduced to a more practical starting solution. The more computationally intensive DUAL OPTIMISATION ALGORITHM (see Section 4.2.2) is now applied to this solution in order to find the optimal cost solution.

*step 4.* End.

---

### 4.2.1 Initial Optimisation Algorithm

As described in Chapter 3, the INITIAL OPTIMISATION ALGORITHM is based on a modified Accelerated Greedy algorithm and is used to find the optimal cost solution (in terms of working capacity costs) that satisfies a given connectivity requirement. It is used in conjunction with the NETWORK OPTIMISATION ALGORITHM to find the starting solution for the more computationally intensive DUAL OPTIMISATION ALGORITHM. Defining the algorithm:

**Input:** The current network configuration defined by the adjacency matrix  $A$ , and the current connectivity requirement  $c_{req}$ .

**Objective:** To find the minimum cost solution  $s = (A, WC)$  that satisfies  $c_{req}$ .

**Output:** The minimum cost solution  $s = (A, WC)$ .

**Process Variables:** Let  $S^{cost} = (s_l^{cost})_{l \in I^{num}}$  denote the set of all potential link removal solution costs, and  $I^{index} = (i_l^{index})_{l \in I^{num}}$  and  $J^{index} = (j_l^{index})_{l \in I^{num}}$  the inter-nodal index of each link. In addition, let the variables  $s_{min}^{cost}$ ,  $l^{num}$ , and  $k$  represent the minimum solution cost, the number of potential links, and the delimiter variable, respectively. Using this notation, the algorithmic procedure can be stated as follows:

---

**ALGORITHM 4.2: INITIAL OPTIMISATION ALGORITHM**

**Step 1.** Calculate the cost of the current network solution and set it equal to the minimum solution cost, i.e. let  $s_{min}^{cost} = F(s^{(1)}) = F(A^{(1)}, WC^{(1)})$  [see Note 1].

**Step 2.** For each link in the current configuration  $A^{(1)}$ , calculate its cost of removal and store those values that correspond to a potential cost improvement. Since the number of potential links is initially unknown, we set  $l^{num} = 0$ . Then,  $\forall i, j \in N$ , where  $i < j$  and  $a_{i,j} = 'y'$ :

- (a) Set  $a_{i,j} = 'n'$ , i.e. remove link  $l_{i,j}$  from the current network configuration, and calculate the new solution cost  $s^{cost} = F(s^{(1-l_{i,j})})$  [see Note 2].
- (b) If  $s^{cost} \leq s_{min}^{cost}$ , then this represents a cost improvement: let  $l^{num} \leftarrow l^{num} + 1$ , and set  $s_{l^{num}}^{cost} = s^{cost}$ ,  $i_{l^{num}}^{index} = i$  and  $j_{l^{num}}^{index} = j$ .
- (c) Reset  $a_{i,j} = 'y'$ .

**Step 3.** Initialise the delimiter (iteration) variable  $k = 1$ .

**Step 4.** Sort the set of solution costs and their corresponding index array values (within the range  $k \rightarrow l^{num}$ ) according to the minimum cost solution.

**Step 5.** For each iteration  $k$ ,  $s_{min}^{cost} = F(s^{(k)})$  denotes the cost of the minimum solution.

- (a) If  $s_k^{cost} \leq s_{min}^{cost}$ , then confirm that this still represents the minimum cost solution:
  - i) set  $a_{i_k^{index}, j_k^{index}} = 'n'$ , and calculate the new solution cost  $s^{cost} = F(s^{(k-l_{i_k^{index}, j_k^{index}})})$  [see Note 2]. Update the solution status field and analysis QuitEvent [see Note 3].
  - ii) if  $s^{cost} \leq s_{min}^{cost}$  and  $s^{cost} \leq s_k^{cost}$  (the next-best solution cost), then update the solution status fields [see Note 3], let  $k \rightarrow k + 1$ , set  $s_{min}^{cost} = s^{cost}$ , and return to Step 5.
  - iii) else if  $s^{cost} \leq s_{min}^{cost}$  and  $s^{cost} > s_k^{cost}$ , reset  $a_{i_k^{index}, j_k^{index}} = 'y'$ , update  $s_k^{cost} = s^{cost}$  and return to Step 4.
  - iv) else if  $s^{cost} > s_{min}^{cost}$ , then the link removal is no longer feasible: let  $k \rightarrow k + 1$  and return to Step 4.
- (b) Else if  $s_k^{cost} > s_{min}^{cost}$ , then end. In accordance to the *monotonicity* property, no further cost improvements can be found.

**Step 6.** End.

*Note 1.* Calls the BASIC NETWORK COST ALGORITHM (see Section 4.2.3.1)

*Note 2.* Calls the BASIC NETWORK COST ALGORITHM. The connectivity requirement and terminating nodes of each link are also passed as constraints.

*Note 3.* A procedural call to the application which updates a number of status fields to indicate the current solution status while the process is running. In addition, the application provides the facility to quit a process at any time. These application specific facilities are discussed in Chapter 5.

---

## 4.2.2 Dual Optimisation Algorithm

As described in Chapter 3, the DUAL OPTIMISATION ALGORITHM is based on the Tabu Search methodology and is used in the refinement phase of the optimisation procedure. Considered one of the more advanced solution methods, it is characterised by its ability to avoid getting trapped in local optima.

The solution method works by excepting least-cost non-improving moves if no improving move is possible, i.e. when a local optimum is reached, and by preventing the process from reverting back to this local optimum. This latter condition, which is also known as cycling, is prevented by creating a Tabu list of the most recent non-improving moves. Since a number of sub-optimal link configurations may exist<sup>21</sup>, the size of the Tabu list is an important process constraint. If it is too small, the process may not be able to escape from the local optima, while if it is too large, the process may be restricted in its search for optimal solutions. Results from [27] indicate that the best solutions can be achieved with Tabu lists ranging between 5 and 20 moves.

While the Tabu list prevents immediate cycling, long term cycling can still occur. This is where a local optimum is reached more one than once. Since this will continue for the remainder of the process, it is important to detect and break these cycles. The most effective technique doing so is to introduce some form of perturbation into the process. In [27], two alternatives were investigated. The first involves altering the status of a small number of links chosen at random. The second alternative is to generate a completely new random starting solution. The process then continues with this new solution until another cycle is detected or the limit on the number of iterations is reached. During initial trials of the Tabu Search process, it was found that cycling did not occur that

---

<sup>21</sup>This is particularly true of larger networks where a number of sub-optimal configurations may exist in the overall network configuration.

often for this particular problem, i.e. 1000 iterations<sup>22</sup>. As this corresponds to a run time of ~ 9.5 hours on a 100 MHz Pentium™ PC, there was no apparent need for these techniques. However, the concept of introducing some form of perturbation to accelerate the process was investigated.

Taking the same test case networks, the Tabu Search process was again applied. However, in this case the process was temporarily suspended after 100 iterations and the status of a number of links altered. The process then resumed with this new solution for the next 100 iterations and the same procedure was applied. This continued for 10 separate runs, i.e. an overall run time of 1000 iterations. In comparison to the basic process, this procedure was consistently able to find a better cost solution after less than 5 runs. In effect, the perturbation provided the impetus to literally jump from the current local optima and, as a result, the process was able to perform a much wider search of the solution space.

The effectiveness of this technique is totally dependent on the sub-set of links chosen at the end of each run, since they have the effect of displacing the sub-optimal structures in the current solution. As they are selected at random, there is no guarantee that the resultant moves will be effective. Intuitively, an alternative is to allow the user to interact with this process. As an experienced network designer would have the ability to visualise what a good solution looks like, he would be able to detect potential sub-optimal structures. As mentioned in previous sections, the design tool facilitates this procedure by enabling the network designer to interact with a graphical representation of the solution. The designer can then insert or remove any number of links and apply the Tabu Search process to see if a better cost solution can be found. As the results in Chapter 6 will show, this procedure proved highly effective in guiding the dual optimisation process toward the global optimal solution, while reducing the overall computational requirement considerably. With this in mind, the algorithmic procedure for the DUAL OPTIMISATION ALGORITHM can now be defined.

**Input:** The current network configuration defined by the adjacency matrix  $A$  and the process constraints.

**Process Constraints:** The Tabu list size  $l_{size}$  and the number of additional (Tabu) iterations  $t_{max}$ .

**Objective:** To find the minimum cost solution  $s = (A, WC, SC)$ .

**Output:** The minimum cost solution  $s = (A, WC, SC)$ .

---

<sup>22</sup>These were conducted on three independent 24 node networks. From these results, it was deduced that the solution space consisted of a large number of well distributed sub-optimal solution states.

**Process Variables:** Let  $s_{\min}^{cost}$  denote the minimum solution cost and  $A^{\min} = (a_{i,j}^{\min})_{i,j \in N}$  its configuration. For each move (iteration), let  $s_{best}^{cost}$  denote the solution cost of the best move,  $i_{best}$  and  $j_{best}$  the inter-nodal index of the move (link), and  $m_{best}$  the status of the move (i.e. insert or remove). To maintain the Tabu list, let  $I^{tabu} = (i_x^{tabu})_{x \in L_{size}}$  and  $J^{tabu} = (j_x^{tabu})_{x \in L_{size}}$  denote the set of inter-nodal indices of the most recent non-improving moves. In addition, let  $k$  represent the overall iterative variable and  $t$  the number of non-improving moves (iterations). Finally, let the matrix  $s^{tabu} = (s_{i,j}^{tabu})_{i,j \in N}$  represent the Tabu status of each inter-nodal link. Using this notation, the algorithm can now be stated as follows:

---

**ALGORITHM 4.3: DUAL OPTIMISATION ALGORITHM**

**Step 1.** Initialise all entries in Tabu status matrix to *false* and set  $k = t = 1$ . Calculate the total cost of the initial network solution [see *Note 1*] and set this equal to the minimum solution cost, i.e.  $s_{\min}^{cost} = F(s^{(1)})$ .

**Step 2.** For each iteration  $k$ , let  $s^{(k)}$  denote the minimum cost solution. Set  $s_{best}^{cost} = \infty$  and search the neighbourhood of this solution for the best-cost solution (move)<sup>23</sup>. That is,  $\forall i, j \in N$ , where  $i < j$ , then:

- (a) If  $a_{i,j} = y'$  and  $s_{i,j}^{tabu} = false$ , then:
  - i) set  $a_{i,j} = n'$  and calculate the new solution cost  $s^{cost} = F(s^{(k-l_{i,j})})$  [see *Note 1*].
  - ii) if  $s^{cost} \leq s_{best}^{cost}$ , then set  $s_{best}^{cost} = s^{cost}$ ,  $i_{best} = i$ ,  $j_{best} = j$ , and  $m_{best} = y'$ .
  - iii) reset  $a_{i,j} = y'$ .
  - iv) update solution status field and analyse QuitEvent [see *Note 2*].
- (b) If  $a_{i,j} = n'$  and  $s_{i,j}^{tabu} = false$ , then:
  - i) set  $a_{i,j} = y'$  and calculate the new solution cost  $s^{cost} = F(s^{(k+l_{i,j})})$  [see *Note 1*].
  - ii) if  $s^{cost} \leq s_{best}^{cost}$ , then set  $s_{best}^{cost} = s^{cost}$ ,  $i_{best} = i$ ,  $j_{best} = j$  and  $m_{best} = y'$ .
  - iii) reset  $a_{i,j} = n'$ .
  - iv) update solution status field and analyse QuitEvent [see *Note 2*].

**Step 3.** With the solution neighbourhood searched, implement the move that represents the best cost. That is, set  $a_{i_{best},j_{best}} = m_{best}$ .

**Step 4.** Check if this represents a non-improving move and update the relevant variables accordingly.

- (a) If  $s_{best}^{cost} \leq s_{\min}^{cost}$ , then this represents a new minimum solution cost. Set  $s_{\min}^{cost} = s_{best}^{cost}$  and  $A^{\min} = A$ . Update the solution status fields [see *Note 2*].

---

<sup>23</sup>Represents the largest cost improvement or, in the case of a non-improving move, the smallest loss.

- (b) Else if,  $s_{best}^{cost} > s_{min}^{cost}$  and  $t \leq l_{size}$ , then the current move<sup>24</sup> is pushed onto the Tabu list. Set  $s_{i_{best}, j_{best}}^{tabu} = true$ ,  $i_t^{tabu} = i_{best}$  and  $j_t^{tabu} = j_{best}$ , and let  $t \rightarrow t + 1$ .
- (c) Else if,  $s_{best}^{cost} > s_{min}^{cost}$  and  $t \leq l_{size}$ , then the least recent move is removed from the top of the list; all other moves are iterated up one place; and the current move is pushed onto the list. That is:
- i) set  $s_{i_t^{tabu}, j_t^{tabu}}^{tabu} = false$ , i.e. reinstate the least recent non-improving move (link).
  - ii) for  $x = 1, \dots, (l_{size} - 1)$ , set  $i_x^{tabu} = i_{x+1}^{tabu}$ , and  $j_x^{tabu} = j_{x+1}^{tabu}$ .
  - iii) set  $s_{i_{best}, j_{best}}^{tabu} = true$ ,  $i_{l_{size}}^{tabu} = i_{best}$  and  $j_{l_{size}}^{tabu} = j_{best}$ , and let  $t \rightarrow t + 1$ .

*Step 5.* Consider the next iteration:

- (a) If  $t \leq t_{max}$ , then let  $k \rightarrow k + 1$  and return to *Step 2*.
- (b) Else, end. No further non-improving moves are permitted. Set  $A = A^{min}$ .

*Step 6.* End.

*Note 1.* Calls the TOTAL NETWORK COST ALGORITHM (see Section 4.2.3)

*Note 2.* See *Note 3* of the INITIAL OPTIMISATION ALGORITHM

---

## 4.2.3 Network Cost Algorithms

Given any network configuration, the cost of the network solution must be calculated. This may represent the working capacity solution cost or the working and spare capacity solution cost. As such, there are two cost algorithms defined for the overall design problem.

### 4.2.3.1 Basic network cost algorithm

The BASIC NETWORK COST ALGORITHM is defined as follows:

**Input:** The current network configuration defined by the adjacency matrix  $A$ .

**Additional Constraints:** In the case of the INITIAL OPTIMISATION ALGORITHM, the connectivity requirement  $c_{req}$  and the terminating nodes of the removed link are given as input constraints.

**Objective:** To generate the optimal working capacity solution  $s = (A, WC)$ .

**Output:** The solution cost  $F(s)$  given by Equation 3.7.

---

<sup>24</sup>The Tabu list must be full before we can iterate through it (see *Step 4. (c)*)

**Process Variables:** Let  $s^{cost}$  represent the cost of the solution and *connected* a boolean status variable. Based on this notation, the algorithmic procedure is stated as follows:

---

**ALGORITHM 4.4: BASIC NETWORK COST ALGORITHM**

*Step 1.* Apply the JOINT OPTIMAL WORKING CAPACITY ASSIGNMENT ALGORITHM based on the input parameters. Set the return value equal to *connected*.

- (a) If *connected* = *true*, then calculate the cost of this solution  $s^{cost} = F(A, WC)$ .
- (b) Else if *connected* = *false*, then the current solution is not feasible. Set  $s^{cost} = \infty$ .

*Step 2.* Return  $s^{cost}$ .

---

**4.2.3.2 Total network cost algorithm**

The TOTAL NETWORK COST ALGORITHM is defined as follows:

**Input:** The optimal working capacity solution  $s = (A, WC)$  and the restoration constraint.

**Objective:** To generate the overall network solution  $s = (A, WC, SC)$ .

**Output:** The solution cost  $F(s)$  given by Equation 3.7.

**Process Variables:** Let  $s^{cost}$  represent the cost of the solution. The algorithmic procedure is stated as follows:

---

**ALGORITHM 4.5: TOTAL NETWORK COST ALGORITHM**

*Step 1.* Apply the BASIC SPARE CAPACITY ASSIGNMENT ALGORITHM to  $s = (A, WC)$  and generate the overall network solution  $s = (A, WC, SC)$  based on the current restoration constraint. Then calculate the cost of this solution  $s^{cost} = F(s)$ .

*Step 2.* Return  $s^{cost}$ .

---

**4.3 Joint Optimal Routing & Working Capacity Assignment Algorithm**

The JOINT OPTIMAL ROUTING AND WORKING CAPACITY ASSIGNMENT ALGORITHM is composed of various sub-routines and function calls to other algorithms. A high-level view of the algorithm will be considered before looking at the individual steps. Defining the problem:

**Input:** The current network configuration, defined by the adjacency matrix  $A$  and the inter-nodal distance matrix  $D$ .



**Additional Constraints:** For the INITIAL OPTIMISATION ALGORITHM, in order to verify that the removal of a given link represents a feasible solution, the connectivity requirement and the inter-nodal index of the link are provided as additional input parameters.

**Objective:** To generate the optimal working capacity assignment solution  $s = (A, WC)$ .

**Output:** The connectivity status of the network represented by the boolean variable, *connected*, and the solution  $s = (A, WC)$ .

**Process Variables:** Boolean status variable, *connected*. Based on this notation, the high-level algorithmic procedure is stated as follows:

---

**ALGORITHM 4.6: JOINT OPTIMAL ROUTING & WORKING CAPACITY ASSIGNMENT ALGORITHM**

*Step 1.* Generate the optimal paths by calling the OPTIMAL ROUTING ALGORITHM. Set the return value equal to *connected*. If *connected* = true, continue. Else, go to *Step 4*.

*Step 2.* [Optional] Call the CONNECTIVITY TEST ALGORITHM based on the additional input constraints. Set the return value equal to *connected*. If *connected* = true, continue. Else, go to *Step 4*.

*Step 3.* Based on the optimal routes generated in *Step 1*, assign working capacity by calling the WORKING CAPACITY ASSIGNMENT ALGORITHM.

*Step 4.* Return *connected* to the calling function.

---

### 4.3.1 Optimal Routing Algorithm

The OPTIMAL ROUTING ALGORITHM, is developed in the following sub-sections.

#### 4.3.1.1 Dijkstra's shortest-path algorithm

This algorithm is based on a labelling concept and is an efficient method for finding the shortest-path between any pair of nodes in a network. The following defines the problem:

**Input:** The current network configuration defined by the adjacency matrix  $A$  and the inter-nodal distance matrix  $D$ .

**Objective:** Compute the shortest-path between any pair of nodes, denoted as  $s$  (*source*) and  $t$  (*terminal*).

**Output:** A boolean variable *connected*, which indicates if a path exists. If a path does exist, then it will be stored in the  $s^{th}$  row of the paths matrix,  $P_s(p_{s,i})_{i \in N}$ .

**Process Variables:** Let  $D^{path} = (d_i^{path})_{i \in N}$  denote the set of path distances between the source and all other nodes in the network, and  $S^{checked} = (s_i^{checked})_{i \in N}$  the status (label) of each node. In addition,

let  $w_{node}$  represent the working node,  $d_{min}^{path}$  the minimum-path distance,  $n_{min}$  the minimum-node index, and  $w_{found}$  a boolean status variable. Using this notation, the algorithm can now be stated as follows:

---

**ALGORITHM 4.7: DIJKSTRA'S SHORTEST-PATH ALGORITHM**

*Step 1.* Since no paths are known, initialise  $d_i^{path} = \infty$ ,  $s_i^{checked} = false$  (unchecked), and  $p_{s,i} = x$  (undefined),  $\forall i, i \in N$ . Then start out by labelling source node  $s$  as being checked. That is, let  $d_s^{path} = 0$ ,  $s_s^{checked} = true$ , and  $p_{s,s} = s$ . Let  $w_{node} = s$  be the initial working node.

*Step 2.* Let  $w_{found} = false$  and  $d_{min}^{path} = \infty$ , and  $\forall i, i \in N$ , where  $i \neq s$  and  $s_i^{checked} = false$ :

- (a) If  $a_{w_{node},i} = y$ , then recompute  $d_i^{path}$ , such that  $d_i^{path} = \min\{d_i^{path}, d_{w_{node}}^{path} + d_{w_{node},i}\}$  represents the minimum shortest-path distance between  $w_{node}$  and  $i$ . That is, each of the nodes adjacent to  $w_{node}$  are examined.
- (b) If  $\min\{d_i^{path}, d_{w_{node}}^{path} + d_{w_{node},i}\} = d_{w_{node}}^{path} + d_{w_{node},i}$ , then an improved shortest-path distance between nodes  $s$  and  $i$  has been found. The corresponding path matrix element is also updated, such that  $p_{s,i} = w_{node}$ , in order to later reconstruct the final path.
- (c) If  $d_i^{path} < d_{min}^{path}$ , then a new working node has been found. Set  $w_{found} = true$ , and let  $d_{min}^{path} = d_i^{path}$  and  $n_{min} = i$ .

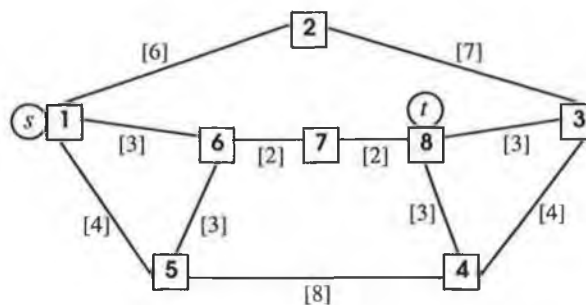
*Step 3.* Consider the next iteration:

- (a) If  $t \neq n_{min}$  and  $w_{found} = true$ , then set  $w_{node} = n_{min}$  and  $s_{n_{min}}^{checked} = true$ , and return to *Step 2*.
- (b) Else if  $t = n_{min}$  and  $w_{found} = true$ , then end. The shortest-path from  $s$  to  $t$  has been found.
- (c) Else if  $t \neq n_{min}$  and  $w_{found} = false$ , then end. No path exists between  $s$  and  $t$ .

*Step 4.* Return  $connected = w_{found}$

---

To illustrate how this is implemented, consider the network topology shown in Figure 4.1 below. The objective is to find the shortest-path between nodes 1 and 8, i.e.  $s = 1$  and  $t = 8$ .

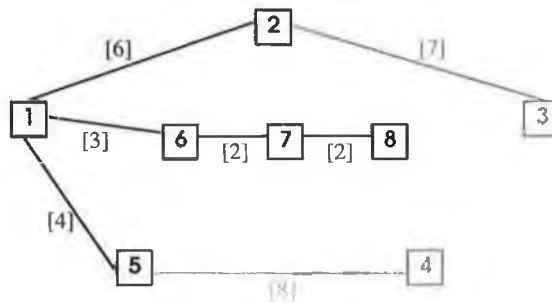


**Figure 4.1** Example network (note: link distances are shown in square brackets)

The full implementation of the algorithm, showing the results and status of the network at each iteration, is given in Appendix A3.1. The shortest-path was found during the fifth iteration of the process. The following is the status of the solution, in terms of the various nodal sets:

$$P_1 = [1, 1, 2, 5, 1, 1, 6, 7], D^{path} = [0, 6, 13, 12, 4, 3, 5, 7], \text{ and } S^{checked} = [1, 1, 0, 0, 1, 1, 1, 1]$$

A graphical representation of this solution is shown in Figure 4.2 below.



**Figure 4.2** Graphical representation of solution (checked nodes are shown in bold)

It is noted that, since five iterations were required, and since a shortest-path was found at each iteration, we have, in effect, also found the shortest-path between node 1 and nodes 6, 5, 7, and 2 respectively. In addition, although nodes 3 and 4 have path distances assigned to them, these distances remain tentative as they are still labelled as being unchecked.

**Computational Complexity:** Each iteration has a computational time complexity of the order of  $O(n)^*$ . Since the shortest-path to node 8 was found during the fifth iteration, the run time for this process was  $5.O(n)$ . The computational complexity therefore varies between  $O(n)$  and  $(n-1).O(n)$  depending on the proximity of the terminating node.

#### 4.3.1.2 Improvements to existing algorithm

Since the objective is to generate optimal routes between all nodal pairs, applying this process to each individual pair of nodes would prove very inefficient. However, the constraint that terminates the process on receipt of  $t$  can be removed and the process continued until the shortest-paths from  $s$  to all other nodes are found. This would result in a maximum run time of  $O(n^2)$ . To impose this new constraint, an iteration variable  $k$  is introduced and the modified algorithm becomes:

---

\* For each iteration all  $n$  nodes must be probed to ascertain their current status.

---

**ALGORITHM 4.8: MODIFIED SHORTEST-PATH ALGORITHM**

*Step 1.* Same as before, except: initialise  $k = 1$ .

*Step 2.* Same as before.

*Step 3.* Consider the next iteration:

- (a) If  $k < n - 1$  and  $w_{found} = true$ , then let  $k \leftarrow k + 1$ ,  $w_{node} = n_{min}$  and  $S_{n_{min}}^{checked} = true$ , and return to *Step 2*.
- (b) Else if  $k = n - 1$  and  $w_{found} = true$ , then end. The shortest path between node  $s$  and all other nodes in the network has been found.
- (c) Else if  $k < n - 1$  and  $w_{found} = false$ , then end. No path exists between node  $s$  and the remaining unchecked nodes.

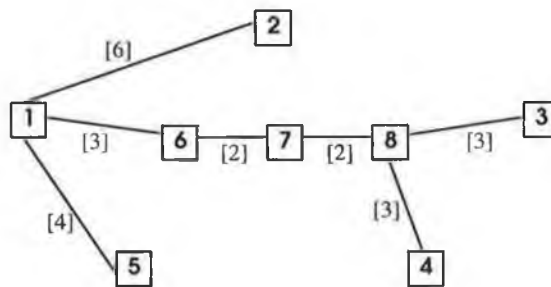
*Step 4.* Return  $connected = w_{found}$ .

---

Applying this to the network example gives the following solution [see Appendix A3.2]:

$$P_1 = [1, 1, 8, 8, 1, 1, 6, 7], D^{path} = [0, 6, 10, 10, 4, 3, 5, 7], \text{ and } S^{checked} = [1, 1, 1, 1, 1, 1, 1, 1]$$

A graphical representation of this solution is shown in Figure 4.3 below.



**Figure 4.3** Graphical representation of solution

#### 4.3.1.3 Imposing minimum-hop routing constraint

The minimum-hop routing constraint is now introduced. It works on the principle of choosing paths based on the minimum number of hops, i.e. links. If more than one minimum-hop path exists between any two nodes, then the path representing the shortest distance is selected.

**Initialisation Phase:** Let  $H^{count} = (h_i^{count})_{i \in N}$  denote the min-hop count between the source and all other nodes in the network, and  $w_{hop}$  the hop count of the current working node. The optimal routing algorithm then becomes:

---

**ALGORITHM 4.9: MINIMUM-HOP/SHORTEST-PATH ALGORITHM**

*Step 1.* Same as before, except: set  $h_s^{count} = w_{hop} = 0$ .

*Step 2.* Same as before, except:  $\forall i, i \in N$ , where  $i \neq s$ , and ( $s_i^{checked} = true$  or  $h_i^{count} > w_{hop}$ ):

(a) If  $a_{w_{node},i} = 1$ ,  $h_i^{count} > w_{hop}$ , and  $\min\{d_i^{path}, d_{w_{node}}^{path} + d_{w_{node},i}\} = d_{w_{node}}^{path} + d_{w_{node},i}$ , then  $p_{s,i} = w_{node}$  and  $h_i^{count} = w_{hop} + 1$ .

(b) Else if  $a_{w_{node},i} = 1$ ,  $h_i^{count} > w_{hop} + 1$ , then  $d_i^{path} = d_{w_{node}}^{path} + d_{w_{node},i}$ ,  $p_{s,i} = w_{node}$  and  $h_i^{count} = w_{hop} + 1$ .

(c) If  $d_i^{path} < d_{min}$ , then let  $w_{found} = true$ ,  $d_{min}^{path} = d_i^{path}$  and  $n_{min} = i$ .

*Step 3.* Same as before, except: let  $w_{hop} = h_{n_{min}}^{count}$ .

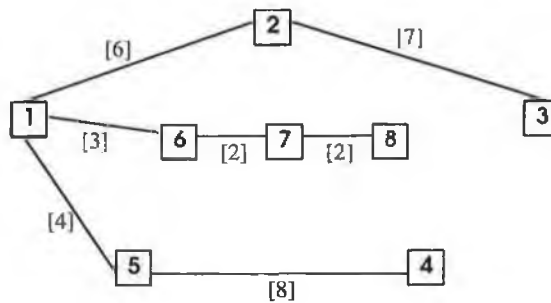
*Step 4.* Return  $connected = w_{found}$ .

---

Applying this to the network example gives the following solution [see Appendix A3.3]:

$$P_1 = [1, 1, 2, 5, 1, 1, 6, 7], D^{path} = [0, 6, 2, 2, 4, 3, 5, 7], \text{ and } S^{checked} = [1, 1, 1, 1, 1, 1, 1, 1]$$

A graphical representation of this solution is shown in Figure 4.4 below.



**Figure 4.4** Graphical representation of solution

#### 4.3.1.4 Overall routing algorithm

Since the computational complexity of the above algorithm is  $O(n^2)$ , if this method is applied to all  $n$  nodes, then the run time for the overall routing process is of the order  $O(n^3)$ . This is the minimum computational requirement for finding the shortest-path between all nodal pairs in a

network and is comparable to other efficient algorithms such as Floyd [30, 36, 41]. The overall algorithm can now be stated as follows:

---

**ALGORITHM 4.10: OPTIMAL ROUTING ALGORITHM**

*Step 1.*  $\forall i \in N$ :

- (a) Apply the MINIMUM-HOP/SHORTEST-PATH ALGORITHM to the current source node  $i$ .
  - i) if *connected* = *true*, then continue.
  - ii) else if *connected* = *false*, then the current network configuration is unconnected and the process is terminated.

*Step 2.* Return *connected*.

---

It is noted that, in *Step 1* the process is terminated if the current network configuration is found to be unconnected. In other words, if a link removal results in any portion of the network being disconnected then the process is automatically terminated and this result is returned to the calling function. Using a separate process for each node implies that an unconnected network will be detected during the first iteration. This is an improvement on Floyd which requires the entire process to be performed before the status of network connectivity can be ascertained.

### 4.3.2 Interpretation of Paths Matrix and Path-Trace Algorithm

As shown in the above implementations, the optimal path from any given node  $s$  to all other nodes in a network can be represented by the  $s^{\text{th}}$  row of the paths matrix, i.e.  $P_s = (p_{s,j})_{j \in N}$ . This highly efficient method for storing optimal routes is due to the tree-type (inheritance) routing structure generated. From Figure 4.4 it can be seen that, for node  $t$ , the corresponding path element  $p_{i,t} = x$  represents the index to the node immediately preceding  $t$  on the path from  $t$  to  $s$ . Similarly, node  $y$ , where  $y = p_{i,x}$ , represents the preceding node on the path between  $x$  and  $s$ , and is therefore the next node on the path between  $t$  and  $s$ . Using this recursive step, the path between node  $s$  and any other node in the network can be traced. The algorithm can now be defined as follows:

**Input:** The paths matrix  $P$  and the index to any inter-nodal link denoted  $s$  and  $t$ .

**Objective:** To trace all nodes (links) along the path between  $s$  and  $t$ . In addition, depending on the calling function, an appropriate sub-routine can also be applied to each link found.

**Process Variables:** Let  $x$  and  $y$  denote the trace variables.

---

**ALGORITHM 4.11: PATH-TRACE ALGORITHM**

*Step 1.* Initialise the trace variables by setting  $x = y = t$ .

*Step 2.* While  $y \neq s$ :

(a) Let  $x = p_{s,x}$ .

(b) Nodes  $x$  and  $y$  now define the current link ( $l_{x,y}$ ) on the path between nodes  $s$  and  $t$ . A [SUB-ROUTINE] can now be applied to this link.

(c) Let  $y = x$ .

---

Taking  $P_1 = [1, 1, 2, 5, 1, 1, 6, 7]$  from the last solution, it can be illustrated how this algorithm is implemented by tracing the path between nodes 1 and 4, i.e.  $s = 1$  and  $t = 4$ .

---

**EXAMPLE 4.1: IMPLEMENTATION OF PATH-TRACE ALGORITHM**

*Step 1.* Let  $x = y = 4$ . In terms of terminating and intermediate nodes, the known path is: 1-...-4.

*Step 2.* Let  $x = p_{s,x} = p_{1,4} = 5$ . Then link  $l_{5,4}$  is the first link on the path between nodes 1 and 4, and we have 1-...-5-4. Let  $y = x = 5$ . Since  $y \neq 1$ , repeat.

*Step 2.* Let  $x = p_{1,5} = 1$ . Then  $l_{1,5}$  is the next link on the path between nodes 1 and 4, and the path becomes 1-...-1-5-4. Let  $y = x = 1$ . Since  $y = s = 1$ , the path between nodes 1 and 4 has been traced, i.e. 1-5-4 or  $l_{1,5} \rightarrow l_{5,4}$ . Note that, using the standard shortest-path solution the equivalent path would be  $l_{1,6} \rightarrow l_{6,7} \rightarrow l_{7,8} \rightarrow l_{8,4}$ .

---

The path-trace algorithm is one of the most utilised sub-routines in the overall design and optimisation process. It is called by numerous algorithms and other sub-routines, including:

- The WORKING CAPACITY ASSIGNMENT ALGORITHM (see Section 4.3.4).
- The path-level SPARE CAPACITY ASSIGNMENT ALGORITHM (see Section 4.4.4).
- PATH TRACE : A NetOpt utility that enables designers to display the routing tree from any given node to all other nodes on the network (see Chapter 5).
- PATHS USING: A NetOpt utility that enables designers to list and display all paths being routed over any given link in the current network solution (see Chapter 5).

### 4.3.3 Connectivity Test Algorithm

Called in conjunction with the MODIFIED ACCELERATED GREEDY ALGORITHM, this algorithm is used to generate a benchmark topology that satisfies a given connectivity requirement. The problem is defined as follows:

**Input:** The inter-nodal distance and paths matrices  $D$  and  $P$ , the terminating nodes of any link  $s$  and  $t$ , and the connectivity requirement  $c_{req}$ .

**Objective:** To verify that  $c_{req}$  node disjoint paths exist between nodes  $s$  and  $t$ .

**Output:** Verification: *connected* is *true* or *false*.

**Process Variables:** Let matrix  $D^{path} = (d_i^{path})_{i \in N}$  denote the set of path distances between the source and all other nodes, and  $S = (s_i)_{i \in N}$  the status of each node. In addition, let  $p_{count}$  denote the path counter,  $p_{found}$  a boolean status variable, and  $c_{count}^s$  and  $c_{count}^t$  the connectivity counter for the source and terminating node, respectively. Using this notation, the algorithm can now be stated:

#### ALGORITHM 4.12: CONNECTIVITY-TEST ALGORITHM

*Step 1.* Initialise  $c_{count}^s = c_{count}^t = 0$  and  $p_{found} = false$ . Then  $\forall i, i \in N$ :

- (a) If  $a_{s,i} = y'$  and  $i \neq t$ , then let  $s_i = a'$  (adjacent) and  $c_{count}^s \leftarrow c_{count}^s + 1$ , otherwise set  $s_i = u'$  (unchecked). Similarly, if  $a_{s,i} = y'$  and  $i \neq s$ , then let  $c_{count}^t \leftarrow c_{count}^t + 1$ .
- (b) If  $c_{count}^s < c_{req}$  or  $c_{count}^t < c_{req}$ , then go to *Step 4*, i.e. the current configuration fails the basic requirement of having  $c_{req}$  node disjoint paths if  $c_{req}$  adjacent nodes do not exist at either end. Otherwise, initialise  $p_{count} = 0$  and set  $s_s = c'$  (checked).

*Step 2.* (*iteration*  $p_{count}$ )  $\forall i, i \in N$ , let  $d_i^{path} = 0$  (the accumulative path distance), if  $s_i = u'$  and  $i \neq t$ , compute the path distance between nodes  $i$  and  $t$  using the PATH TRACE ALGORITHM with sub-routine: if  $s_y = u'$  (confirm it is node disjoint), then let  $d_i^{path} = d_i^{path} + d_{x,y}$ , else, reset  $d_i^{path} = 0$  to denote a non-disjoint path exists between  $i$  and  $t$ .

*Step 3.*  $\forall i, i \in N$ :

- (a) If  $s_i = a'$ ,  $d_i^{path} > 0$ , and  $d_{min}^{path} > d_{s,i} + d_i^{path}$ , then set  $d_{min}^{path} = d_{s,i} + d_i^{path}$ ,  $p_{found} = true$ ,  $A_{min} = i$ , and  $I_{min} = i$  (Adjacent and Intermediary path nodes).
- (b) Else if  $s_i = a'$  and  $d_i^{path} = 0$ , then a non-disjoint path exists between adjacent node  $i$  and terminal node  $t$ . A new path must therefore be sought between node  $i$  and any intermediate node  $j \in N$ , such that  $j$  is unchecked and specifies a node disjoint path between itself and the node  $t$ . That is,  $\forall j \in N$ , if  $s_j = u'$  and  $d_j^{path} > 0$ , compute the path distance between  $i$  and  $j$  using the PATH TRACE ALGORITHM with the same sub-routine used in *Step 2*. If  $d_{min}^{path} > d_{s,i} + d_i^{path}$ , set  $d_{min}^{path} = d_{s,i} + d_i^{path}$ ,  $p_{found} = true$ ,  $A_{min} = i$ , and  $I_{min} = j$ .



Step 4. Consider the next iteration:

- (a) If  $p_{found} = true$  and  $p_{count} < c_{req} - 1$ , let  $p_{count} \leftarrow p_{count} + 1$ , and label all nodes on this path as checked. That is, set  $s_{A_{min}} = s_{I_{min}} = c'$ , and using the PATH TRACE ALGORITHM with subroutine:  $s_y = c'$ , label all nodes on the two sub-paths  $p(A_{min}, I_{min})$  and  $p(I_{min}, end)$  as checked. Set  $p_{found} = false$  and return to Step 2.
- (b) Else if  $p_{found} = true$  and  $p_{count} = c_{req} - 1$ , then end.  $c_{req}$  node disjoint paths exist.
- (c) Else if  $p_{found} = false$  and  $p_{count} < c_{req} - 1$  then end.  $c_{req}$  node disjoint paths do not exist.

Step 5. Return  $connected = P_{found}$ .

---

### 4.3.4 Working Capacity Assignment Algorithm

The WORKING CAPACITY ASSIGNMENT problem is defined as follows:

**Input:** The inter-nodal demand requirement and paths matrices  $R$  and  $P$ .

**Objective:** In accordance with the optimal paths, assign working capacity on each link in order to satisfy (route) all inter-nodal demands.

**Output:** The optimal working capacity solution  $s = (A, WC)$ .

The algorithm is stated as follows:

---

#### ALGORITHM 4.13: WORKING CAPACITY ASSIGNMENT ALGORITHM

Step 1. Since the existing entries in the working capacity matrix are no longer valid, reset them to zero.

Step 2.  $\forall i, j \in N$ , where  $i < j$ , apply the PATH-TRACE ALGORITHM between nodes  $i$  and  $j$  with the following subroutine:  $wc_{x,y} = wc_{x,y} + r_{i,j}$  (i.e. increment the capacity on each link along the path).

Step 3. End.

---

### 4.4 Optimal Spare Capacity Assignment Algorithm

As presented in Chapter 3, the solution method to the OPTIMAL SPARE CAPACITY ASSIGNMENT problem is composed of three sequential solution steps. The problem is defined as follows:

**Input:** The current network configuration and optimal working capacity solution  $s = (A, WC)$ . The paths matrix  $P$  is also used to assign spare capacity based on path-level restoration. In addition, the inter-nodal distances matrix  $D$  is used during the adaptive minimisation step to minimise the capacity routing costs.

**Objective:** Generate the optimal network spare capacity solution  $s = (A, WC, SC)$ , that ensures 100% restoration against any single link failure.

**Optional Constraints:** Due to the computational requirement, the final adaptive minimisation step is only called by a stand-alone process. In addition, there is the option of assigning spare capacity based on line-level or path-level restoration.

**Output:** The optimal spare capacity solution  $s = (A, WC, SC)$ .

Based on the various solution steps, the following is a high-level view of the OPTIMAL SPARE CAPACITY ASSIGNMENT ALGORITHM:

---

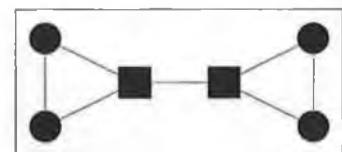
**ALGORITHM 4.14: OPTIMAL SPARE CAPACITY ASSIGNMENT ALGORITHM**

- Step 1.* Apply the INITIAL SPARE CAPACITY ASSIGNMENT ALGORITHM.
  - Step 2.* Using the resultant assignments as reference flows, apply the GLOBAL SPARE CAPACITY ASSIGNMENT ALGORITHM.
  - Step 3.* [Optional] Apply the ADAPTIVE MINIMISATION ALGORITHM to minimise the sum of the spare capacity assignments subject to the optimal flow constraints.
  - Step 4.* [Optional] Confirm that the current flow assignments are feasible. If not, add new constraints and return to *Step 3*.
  - Step 5.* End.
- 

It is noted that, in the case of the BASIC SPARE CAPACITY ASSIGNMENT ALGORITHM, only *Steps 1, 2* and *5* are implemented.

#### 4.4.1 Initial Spare Capacity Assignment Algorithm

In Section 3.3.4, the solution method to the optimal spare capacity assignment problem was illustrated using an 8 node network configuration. In that example, all links in the network supported distributed restoration, i.e. at least two node disjoint paths existed between each node. However, since point-to-point connections may



exist, we must initially remove these links from the process and assign spare capacity on a point-to-point basis. To illustrate this concept consider the simple 6 node network configuration shown here. It is composed of two ring topologies interconnected by a single link. Now, if a link on either ring fails, then there is no use in rerouting the failed capacity over the interconnecting link since there is no return path. Hence, we must remove this link from the process.

To find which links support distributed restoration, each link is temporarily removed from the current configuration and a modified<sup>25</sup> version of DIJKSTRA'S ALGORITHM is applied to the link to see if an alternative path exists. If the removal results in an unconnected network, i.e. the return value  $connected = false$ , then the link is removed from the process by setting  $a_l = f'$ . In addition, spare capacity is assigned to this link on a point-to-point basis, i.e.  $sc_l = wc_l$ . Inversely, if the return value  $connected = true$ , then the link supports distributed restoration and set  $a_l = t'$  to reflect this. Since this is a valid link, the link index arrays and the various nodal arrays (*process variables*) are updated. With this in mind, the overall problem can be defined as follows:

**Input:** The current network solution  $s = (A, WC)$ .

**Objective:** To optimally assign spare capacity locally around each node.

**Output:** The approximate solution  $s = (A, WC, SC)$ .

**Process Variables:** Let the following arrays denote the various nodal characteristics defined for the process:  $N^{potential\_flow} = (n_i^{potential\_flow})_{i \in N}$ ,  $N^{con} = (n_i^{con})_{i \in N}$ ,  $N^{max\_wc} = (n_i^{max\_wc})_{i \in N}$ , and  $N^{index}$ . Let  $F^{req} = (f_i^{req})_{i \in N}$  denote the flow requirements on each link during the various nodal processes. In addition, let  $l^{num}$  denote the number of links that support distributed restoration, and  $I^{index}$  and  $J^{index}$  the index arrays of these links (used in the next Solution Step). Using this notation, the algorithm can now be stated as follows:

#### ALGORITHM 4.15: INITIAL SPARE CAPACITY ASSIGNMENT ALGORITHM

*Step 1.* Initially, the number of links supporting distributed restoration, the spare capacity matrix, and the various nodal arrays are set to zero. Then,  $\forall i, j \in N$ , where  $i < j$  and  $a_{i,j} = y'$ :

- (a) Apply DIJKSTRA'S ALGORITHM to ascertain if the link supports distributed restoration. The result is given by the return value *connected*.
- (b) If  $connected = true$ , let  $a_{i,j} = t'$  to reflect this. Since this is a valid link, the various arrays and control variables must be updated (for both terminating nodes):
 

$n_i^{con} \leftarrow n_i^{con} + 1, n_j^{con} \leftarrow n_j^{con} + 1, n_i^{max\_wc} = \max\{n_i^{max\_wc}, wc_{i,j}\},$	let
$n_j^{max\_wc} = \max\{n_j^{max\_wc}, wc_{i,j}\}, i_{l^{num}}^{index} = i, j_{l^{num}}^{index} = j,$	
$\text{and } l^{num} \rightarrow l^{num} + 1.$	
- (c) Otherwise no alternative route exists, so we must remove this link from the process and assign spare capacity on a point-to-point basis: let  $a_{i,j} = f'$  and  $sc_{i,j} = wc_{i,j}$ .

*Step 2.* The potential flow characteristic  $n_i^{potential\_flow}$  around each node is then evaluated according to equation 3.22. Nodes are then sorted according to the maximum value to specify the order in which

<sup>25</sup>In this modification, the standard DIJKSTRA'S ALGORITHM (see Section 4.3.1.1) is applied to find the shortest-path between the terminating nodes of the link, with the exception that the paths matrix is not updated.

they are selected. During this process, the nodal index array  $N^{index}$  is used to store the ordered list of sorted nodes.

*Step 3.* Each node is then visited in accordance with the above index list and spare capacity is assigned locally. That is,  $\forall i \in N$ :

(a) Initialise  $k = 0$ , where  $k$  is used to denote the number of adjacently connected nodes to  $n_i^{index}$  that support distributed restoration. A nodal index array  $N^{flow\_index}$  is also used to store the reference to these nodes<sup>26</sup>. In other words,  $\forall j \in N$ :

i) if  $a_{n_i^{index}, j} = t'$ , then evaluate the inter-nodal flow requirement  $f_j^{req}$  according to equation 3.23, set  $n_k^{flow\_index} = j$  and let  $k \rightarrow k + 1$ .

(b) Cut each of these links in turn and optimally reroute the corresponding flow requirement over the remaining links. That is,  $\forall j \in k$ :

i) [Optional] apply the RELEASE PHASE ALGORITHM to this link, i.e.  $l_{n_i^{index}, n_j^{flow\_index}}$ , if spare capacity is to be assigned according to path-level restoration (see Section 3.3.5).

ii) assign spare capacity in accordance with the method of proportionality presented in Chapter 3, Section 3.3.4.1.

*Step 4.* End.

#### 4.4.2 Global Spare Capacity Assignment Algorithm

Based on the solution generated in the previous step, the GLOBAL SPARE CAPACITY ASSIGNMENT problem can now be defined.

**Input:** The approximate spare capacity solution  $s = (A, WC, SC)$ , the number of links in the current process  $l^{num}$ , and their corresponding index arrays given by  $I^{index}$  and  $J^{index}$ .

**Objective:** To generate the optimal flow constraints and assign additional spare capacity to satisfy these constraints. It is noted that the optimal flow constraints are only stored if the final minimisation step is to be applied.

**Output:** A feasible near-optimal solution  $s = (A, WC, SC)$ , and if required the optimal flow constraints represented by the partial cut-set matrix  $Q$ .

**Process variables:** Let  $S^{visited} = (s_i^{visited})_{i \in N}$  and  $S^{checked} = (s_i^{checked})_{i \in N}$  denote the status of each node during the various link processes. In addition, let  $f^{req}$  denote the flow requirement on each link, and

<sup>26</sup>These links are only of interest for the remainder of this sub-routine. Since they represent a sub-set of all possible links, i.e. adjacent nodes, these reference parameters are used to reduce the computational requirement, i.e. remove redundant iterations.

$F^{excess} = (f_i^{excess})_{i \in N}$  and  $F^{input} = (f_i^{input})_{i \in N}$  the status of flow around each node. Using this notation, the algorithm can be stated as follows:

---

**ALGORITHM 4.16: GLOBAL SPARE CAPACITY ASSIGNMENT ALGORITHM**

*Step 1.* If required, all entries in the cut-set matrix  $Q$  are set to zero. The link index arrays,  $I^{index}$  and  $J^{index}$ , are initially sorted according to maximum working capacity on each link, to specify the order in which they are selected (see Section 3.4.2.2).

*Step 2.* For each link, compute the optimal flow equations. That is,  $\forall k, k \in I^{num}$ :

- (a) [Optional] Apply the RELEASE PHASE ALGORITHM to this link, i.e.  $I_{i_k^{index}, j_k^{index}}$ , if spare capacity is to be assigned according to path-level restoration (see Section 3.3.5).
- (b) Let  $f^{req} = WC_{i_k^{index}, j_k^{index}}$  be the current flow requirement. Assign  $f^{req}$  to node  $i_k^{index}$ , i.e. the source node, and reset the input flow assignments on all other nodes to zero. In addition, label node  $i_k^{index}$  as being visited and checked, and all other nodes as unvisited and unchecked.
- (c) For all nodes that are visited but not checked, calculate the excess potential. That is,  $\forall i, i \in N$ , if  $s_i^{visited} = s_i^{checked} = false$ , then let  $f_i^{excess} = -(f_i^{input})$  and then:  
 $\forall j, j \in N$ , if  $a_{i,j} = t'$  and  $s_j^{checked} = false$ , then  $f_i^{excess} = f_i^{excess} + sc_{i,j}^*$ .
- (d) Label the node with the minimum excess potential as being checked, and make a partial cut around this node to generate the corresponding flow constraint (see Section 3.4.2).  $\forall i, i \in N$ , if  $s_i^{checked} = true$ , then:  
 $\forall j, j \in N$ , if  $a_{i,j} = t'$  and  $s_j^{checked} = false$ , then label [Optional] the corresponding link entry in the cut-set matrix as being cut and reduce  $f^{req}$ , such that  $f^{req} = f^{req} - sc_{i,j}^{**}$ . If this link has not yet been cut, assign input flow to this node as follows:  $f_j^{input} = f_j^{input} + sc_{i,j}$ .
- (e) If  $f^{req} > 0$ , i.e. the current link cut-set does not satisfy the conservation of flow, additional spare capacity is assigned on all the relevant links with accordance to the method of proportionality (see Section 3.3.4.1). Return to (c).

*Step 3.* End.

---

### 4.4.3 Adaptive Minimisation Algorithm

This solution step is only implemented in the stand-alone spare capacity assignment procedure. The problem is defined as follows:

---

\* In the case of path-level restoration:  $f_i^{excess} = f_i^{excess} + sc_{i,j} + 2.rc_{i,j}$

\*\* In the case of path-level restoration:  $f^{req} = f^{req} - (sc_{i,j} + 2.rc_{i,j})$

**Input:** The near-optimal solution  $s = (A, WC, SC)$  and the optimal flow constraints defined by the partial cut-set matrix  $Q$ .

**Objective:** To minimise the capacity switching and routing costs.

**Output:** The minimum spare capacity cost solution  $s = (A, WC, SC)$ .

As the overall ADAPTIVE MINIMISATION ALGORITHM is quite detailed, only the main functional steps will be presented here.

---

**ALGORITHM 4.17: ADAPTIVE MINIMISATION ALGORITHM**

- Step 1.* The flow constraints are analysed and the utilisation characteristics of each link evaluated. Links are then sorted in accordance of the highest utilisation.
- Step 2.* An iterative process is then applied to these links to minimise the capacity switching costs. The objective is to increment the spare capacity requirement on the most utilised links and decrement the requirement on a greater number of least utilised links. The process continues until no further cost improvements can be found.
- Step 3.* A second iterative process is then applied to each link to minimise the capacity routing costs. The objective is to decrement the spare capacity requirement on longer span lengths, by incrementing the requirement on shorter distance links. This continues until no further cost improvements can be found.
- Step 4.* End.

---

#### 4.4.4 Release Phase Algorithm

Used in the assignment of spare capacity based on path-level restoration, the RELEASE PHASE ALGORITHM is defined as follows:

**Input:** The path matrix  $P = (p_{i,j})_{i,j \in N}$  and the terminating nodes of the failed link,  $s$  and  $t$ .

**Objective:** To find all paths using  $l_{s,t}$  and to release the working capacity along each path.

**Output:** The released spare capacity on each link represented by matrix  $RC = (rc_{i,j})_{i,j \in N}$ .

---

**ALGORITHM 4.18: RELEASE PHASE ALGORITHM**

- Step 1.* Initialise all entries in the released capacity matrix at zero.
- Step 2.*  $\forall i, j \in N$ , where  $i > j$ , apply the PATH TRACE ALGORITHM between nodes  $i$  and  $j$  with the following subroutine: if  $x = s$  and  $y = t$  (or vice versa), then the path between nodes  $i$  and  $j$  is routed over the failed link. If this condition holds, then retrace the path between nodes  $i$  and  $j$ , this time releasing the working capacity on each link along the path. In other words, now use the subroutine:  $rc_{x,y} = rc_{x,y} + r_{x,y}$ .
- Step 3.* End.

---

## Chapter 5 - NetOpt Design Tool

---

The previous chapters dealt with the formulation of the optimal design problem and the development of the design methodology used to solve this problem. In this chapter, an overview of the NetOpt design tool and how it facilitates the generation, editing, and optimisation of survivable SDH networks is presented.

### 5.1 Overview of NetOpt

NetOpt is an event-driven application that offers users a wide range of menu commands and associated utilities that facilitate all aspects of the optimal design procedure. The application has a *DOS*-based graphical user interface (GUI) with full mouse support. Extensive use of colour graphics and dialog boxes (similar to those of *Windows*-based applications) aids user interaction throughout the design procedure (Figure 5.1). The main area of the application window is assigned to the active design field. With full scroll bar and zoom functionality, it is used to display and render specified regions of the network configuration within the active network domain (see Section 5.1.1). A pull-down menu and push-button tool bar are located along the top of the design field, with a status bar is along the bottom. The menu and tool bar provide access to the various NetOpt commands (see Section 5.1.2 and 5.1.3), while the status bar is used to display information required for the generation and evaluation of the active network.

#### 5.1.1 Active Design Field

The active design field is a scrollable window that enables the network designer to view any area of the active *network domain* at the current zoom setting. It also has full mouse support, which enables the user to interact with the active design field. During the initialisation phase of a new input network, this facilitates the placement of nodes within the active network domain (see Section 5.1.1.1). Once defined, for all subsequent stages of the design procedure, the current network solution is displayed. The user can also interact with this solution and click on any active network element displayed and apply the various *edit* and *view* commands offered by the application (see Section 5.1.2).

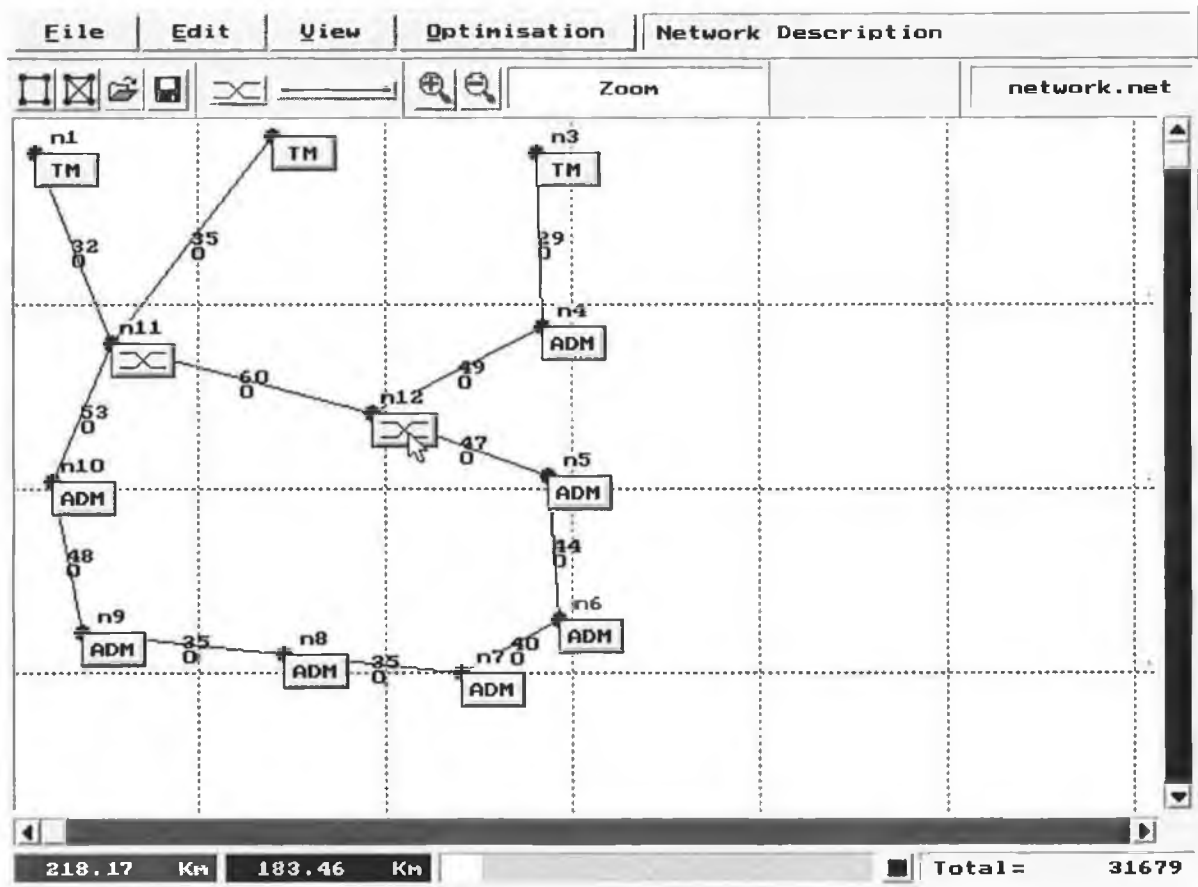


Figure 5.1 NetOpt application

The utilities and design features associated with the active design field will now be considered in more detail.

### 5.1.1.1 Network representation

It is appropriate at this point to recall how the input network infrastructure is initially defined. As stated in Chapter 3, the input network is denoted as an undirected Graph  $G = (N, L)$ , where  $N = (1, \dots, n)$  represents the set of all nodes, and  $L = (L_{i,j})_{i,j \in N}$  represents the set of all inter-nodal links. Since it is assumed that the geographical location of each node will be known in advance, the user must therefore define their locations within a predefined network domain\*. In conjunction with other utilities, the active design field facilitates this initial design step by enabling the user to define

\* The Access and Core network domains are given default settings of  $(100 \times 100) \text{ km}^2$  and  $(400 \times 400) \text{ km}^2$  respectively, although these values can be increased at any point during the design process to accommodate larger network domains.



the required network domain, and then based on this active domain, place each node at the desired geographical location. The  $(x_i, y_i)$  coordinates of each node location are then stored in the respective nodal arrays,  $X = (x_i)_{i \in N}$  and  $Y = (y_i)_{i \in N}$ . Once the nodal infrastructure has been defined, the user must then initialise the various inter-nodal constraints. These operations form part of the overall interactive procedure used to generate and define a new input network (see Section 5.2).

#### **5.1.1.2 Zoom setting**

The *Zoom* setting relates to the resolution of the view, where the resolution represents a unit of distance. At each resolution, the active design field measures approximately  $(600 \times 350)$  units<sup>2</sup>. At the lowest resolution (Zoom 1:1), NetOpt can display the active network to the nearest meter. In other words, the user can view any area of the active network domain measuring approximately  $(600 \times 350)$  m<sup>2</sup>. For each subsequent level, the resolution increases by a factor of 10. In other words, the user can zoom out and view increasing areas of the active network domain measuring approximately  $(6 \times 3.5)$  km<sup>2</sup>,  $(60 \times 35)$  km<sup>2</sup> and  $(600 \times 350)$  km<sup>2</sup>, respectively.

#### **5.1.1.3 Gridlines**

At each zoom level, the active network domain is divided into a matrix of cells measuring  $(100 \times 100)$  units<sup>2</sup>. Hence, there are approximately  $6 \times 3.5$  cells in the active display field at any one time. A gridline marking scheme is then used to indicate the boundaries of these cells.

In Figure 5.1, the zoom setting is at its second highest resolution and the display field therefore represents an area measuring  $(60 \times 35)$  km<sup>2</sup>. This in turn implies that each cell (grid) represents an area of  $(10 \times 10)$  km<sup>2</sup>. At this resolution, the gridlines provide a visual aid for determining the approximate location of each node in relation to the overall network domain. In addition, the grid scheme enables the user to zoom in on a particular region of the active network domain (see Section 5.1.1.7).

#### **5.1.1.4 Network coordinates**

While the gridlines provide the initial visual reference (to within 100 units of the current view), the arrow pointer can be used to determine the exact location (coordinates) of any given node to the nearest unit. To facilitate this, the  $(x, y)$  coordinates of the arrow pointer within the active display field are displayed on the status bar (see Section 5.1.4). As the active display area may represent any scrollable region of the active network domain, these coordinates are translated into absolute values in relation to the overall network domain. Since the resolution, gridlines and network coordinate






display features are all functions of the zoom setting, the following table provides a summary of the effective display characteristics at each level.

Zoom Setting	Resolution	Display Area	Cell Area
<b>Zoom [1] = 1:1</b>	0.001 km	(0.600 x 0.350) km <sup>2</sup>	(0.100 x 0.100) km <sup>2</sup>
<b>Zoom [2] = 1:10</b>	0.01 km	(6.00 x 3.50) km <sup>2</sup>	(1.00 x 1.00) km <sup>2</sup>
<b>Zoom [3] = 1:100</b>	0.1 km	(60.0 x 35.0) km <sup>2</sup>	(10.0 x 10.0) km <sup>2</sup>
<b>Zoom [4] = 1:1000</b>	1 km	(600 x 350) km <sup>2</sup>	(100 x 100) km <sup>2</sup>

**Table 5.1** Active display characteristics for each zoom level

### 5.1.1.5 Scroll bars





Displayed along the right and bottom edges of the active design field, the scroll bars enable the user to move anywhere within the active network domain (see Figure 5.1). The floating boxes inside the scroll bars indicate the vertical and horizontal position of the current view in relation to the overall network domain. The scroll bars can be activated either by mouse or keyboard. To illustrate, consider the example of the vertical scroll bar as shown below. It is noted that all scroll operations are in proportion to the current zoom setting.

-  ← Clicking on the arrow button or pressing the up arrow key scrolls up one cell
-  ← Clicking on the scroll bar or pressing the page up key scrolls up one full screen
-  ← The floating box indicates the vertical location within the network design field
-  ← Clicking on the scroll bar or pressing the page down key scrolls down one full screen
-  ← Clicking on the arrow button or pressing the down arrow key scrolls down one cell

The horizontal scroll bar is used in the same manner to move left or right. In addition, to provide a visual reference to the user, the scroll bars are colour coordinated with the gridlines and the running display of the network coordinates.

### 5.1.1.6 Network rendering

The active network is displayed in terms of the nodal infrastructure and the current network solution  $s(A, WC)^*$ . That is, the nodes, the inter-nodal links, and the corresponding capacity assignments on these links are displayed. This is illustrated by the network example shown in Figure 5.1. Referring to this example, the following object notation is used to denote the various network elements:

-  ← denotes a DXC network element (node)
-  ← denotes an ADM network element (node)
-  ← denotes a TM network element (node)
-  ← denotes a sub-network\*\* (group of nodes)

The connectivity characteristic of each node dictates which network element is rendered. In addition, a four character name tag is also associated and displayed with each node. These tags are defined during the initialisation phase of the input network (see Section 5.2.3). As stated above, the user can interact with the network solution and click on all active network elements displayed.

### 5.1.1.7 Changing the zoom setting

The user can zoom in or out of the current view by selecting the relevant zoom command from either the *View* menu or the tool bar (see Sections 5.1.2 and 5.1.3). In the case of the zoom out command, since this increases the current view by a factor of 10, the application automatically takes this view as the reference point and displays all regions around it. In the case of the zoom in command, the user must specify the region of the current view he wishes to zoom in on. That is, the user must position the arrow pointer and click on the region he wishes to zoom in on. In both cases, the gridlines and display coordinates are re-scaled in accordance with the new zoom setting. A sliding bar located on the tool bar is also used to indicate the current zoom setting.

---

\* If the optimal spare capacity solution  $s(A, WC, SC)$  has been evaluated, then the corresponding capacity assignments are also displayed.

\*\* Due to the area a rendered node occupies, which is independent of the zoom setting, a number of nodes may overlap within a certain region of the display. If this is the case, then this network element is used to denote a sub-network of at least two nodes exists in this particular region of the network. The sub-network can be viewed by clicking on the network element

### 5.1.2 Menu Bar

Located along the top of the application window, the pull-down menu bar provides the network designer with a comprehensive range of commands and associated functions to generate, edit, optimise, and view potential network configurations. All commands are grouped by functional type under the appropriate menu button. To activate a menu:

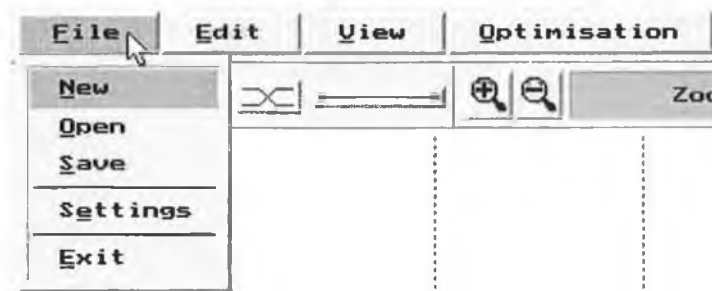
1. Click on the appropriate menu button using the mouse, or
2. Press **Ctrl+hotkey**, where the *hotkey* is the underlined letter in the menu name

Once the menu bar has been activated, the user may scroll right or left, to activate a new menu list. Alternatively, the user can simply click a different menu button. To select a menu command:

1. Click on the appropriate command, or
2. Press **Ctrl+hotkey**, where the *hotkey* is the underlined letter in the command name, or
3. Scroll up or down the menu list and hit return on the appropriate command

A brief description of the various menu commands is presented in following sub-sections.

#### 5.1.2.1 File menu



**New:** initiates the interactive process used to create a new network in NetOpt (see Section 5.2, Initialising a New Input Network).

**Open:** enables the user to open an existing network that has been created in NetOpt and saved to file. An input dialog box prompts the user to select the appropriate filename. When opened the design field is reconfigured to the saved settings (see **Save**).

**Save:** enables the user to save the active network to file. The following network data is stored:

- the various nodal arrays
- the inter-nodal data structures (matrices) used to define the input network

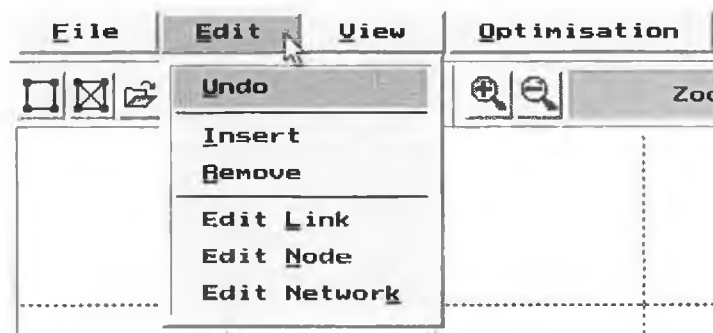
- the inter-nodal data structures used to define the current network solution
- the current design, display, and optimisation settings

Due to the format in which data is stored, the application adopts a .NET extension to denote compatible files. Since the current network configuration is saved, the application provides a *Save As* dialog box to enable the user to specify a new filename if required.

**Settings:** enables the user to access and update information relating to the active network, i.e. the current design area and network type.

**Exit:** enables the user to quit the application. Upon selection, the application will prompt the user to save the active network with unsaved changes.

### 5.1.2.2 Edit menu



**Undo:** enables the user to reverse the last editing action if possible. This only applies to the insertion or removal of either a link or node. As with any command or resultant operation that modifies the network characteristics, an algorithmic procedure is called to update the network solution and refresh the display (see Section 5.3.3).

**Insert:** provides the user with the option to insert a network element into the current network configuration. Upon selection, a dialog box is presented to the user offering him the choice of network elements. In the case of link insertion, the user is prompted to click on the nodal pair he wishes to inter-connect. In the case of node insertion, the user is provided with a drag and drop node which he can place anywhere within the design field (see Section 5.2.3). It is noted that these options are also available on the tool bar (see Section 5.1.3).

**Remove:** provides the user with the option to remove a network element from the current network configuration. Upon selection, the user is prompted to click on the relevant network element. In the

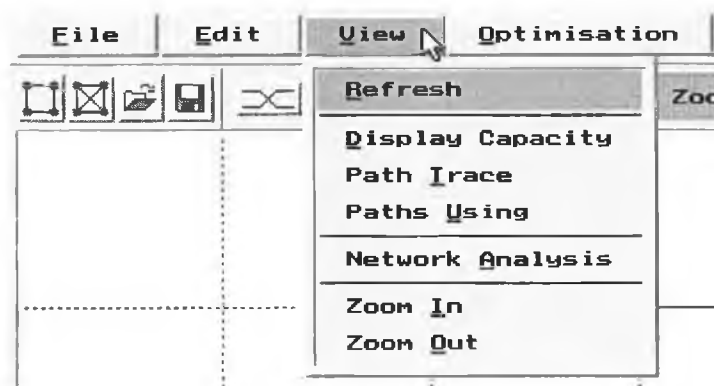
case of node removal, the various nodal arrays and inter-nodal matrices must also be updated. These options are also available by clicking on the relevant network element (see Section 5.1.6).

**Edit Link:** provides the user with a cross reference of link editing features, i.e. insert and remove.

**Edit Node:** provides the user with a cross reference of node editing features. Besides the basic insert/remove option, the user can also edit the nodal characteristics and inter-nodal constraints (see Section 5.3, Network Editing Utilities). It is noted that most of these options are also available by clicking the relevant node (see section 5.1.6).

**Edit Network:** offers the facility to update the network description or, in the case of network simulation, vary the inter-nodal constraints on a global basis (see Section 5.3).

### 5.1.2.3 View menu



**Refresh:** redraws the active network within the active display field and is used to over-ride the path trace feature or any other procedure that may have highlighted segments of the active network.

**Display Capacity:** provides the user with the option of displaying inter-nodal capacity assignments of the current network solution.

**Path Trace:** enables the user to view (highlight) the optimal routes from a specified node to all other nodes in the network. Upon selection, the user is prompted to click on the relevant node. Alternatively, this option is available by clicking directly on the relevant node. It is noted that the various routes remain highlighted until the network is refreshed or another node is selected.

**Paths Using:** enables the user to view all paths being routed over a specified link. Upon selection, the user is prompted to click on the relevant link. Once the link has been selected, a list of all inter-

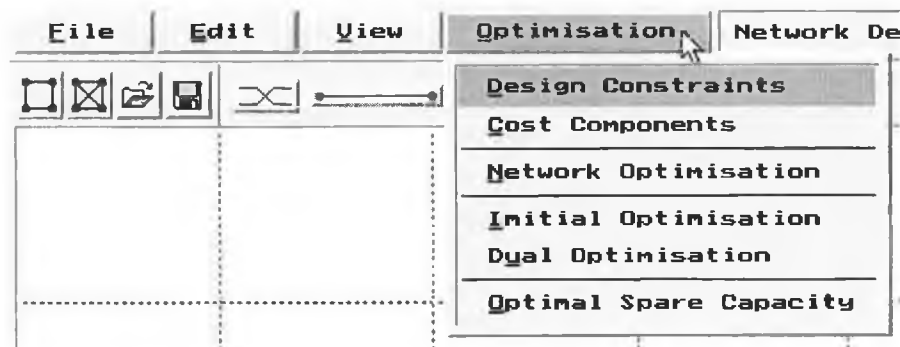
nodal paths (and corresponding demands) being routed over that link is presented in the form of a scrollable dialog box. Alternatively, this option is available by clicking directly on the relevant link.

**Network Analysis:** enables the user to obtain statistical information relating to the network characteristics for any given solution. This is discussed in Chapter 6, Results and Analysis.

**Zoom In:** enables the user to zoom in on a specified region of the design field. Upon selection, the user is prompted to click on the region he wishes to zoom in on.

**Zoom Out:** enables the user to zoom out of the current view.

#### 5.1.2.4 Optimisation menu



**Design Constraints:** enables the user to access and edit the design constraints, i.e. the optimal routing and restoration constraints (see Section 5.4, Network Optimisation).

**Cost Components:** enables the user to access and edit the current cost components settings (see Section 5.4).

**Network Optimisation:** enables the user to apply the NETWORK OPTIMISATION ALGORITHM to the active network (see Section 5.4).

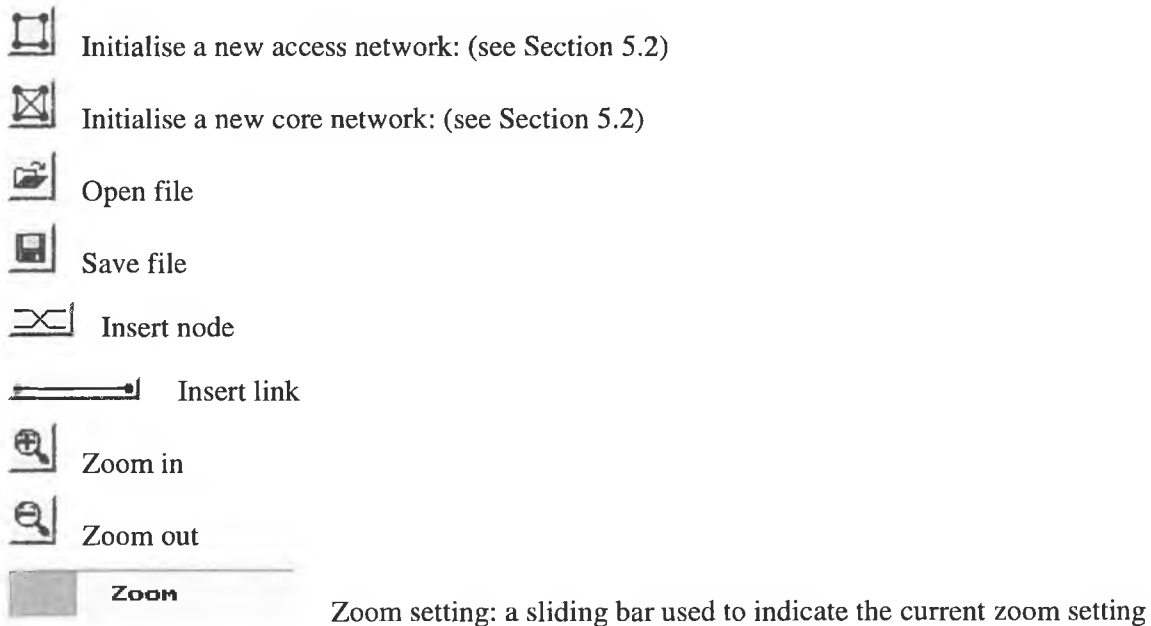
**Initial Optimisation:** enables the user to apply the INITIAL OPTIMISATION ALGORITHM to the current network configuration (see Section 5.4).

**Dual Optimisation:** enables the user to apply the DUAL OPTIMISATION ALGORITHM to the current network configuration (see Section 5.4).

**Opt. Spare Capacity:** enables the user to apply the OPTIMAL SPARE CAPACITY ASSIGNMENT ALGORITHM to the current network configuration (see Section 5.4).

### 5.1.3 Tool Bar

Located along the top of the application window and below the menu bar, the mouse driven tool bar provides quick access to many of the commands provided by the menu bar. The following is a list of the tool bar buttons:



Referring to Figure 5.1, two text fields are also located along the top of the application window. The first is used to display a 30-character description of the active network. This is provided by the user during the initialisation phase and can be edited at any point during the design process. The second field is used to display the filename of the active network.

### 5.1.4 Status Bar

The status bar is located along the bottom of the application window and displays information relating to the active network and various application processes. Referring to Figure 5.1, these information fields include:

*X, Y coordinates:* provide a running display of the arrow pointer coordinates within the active design field.

*Network connectivity status:* a sliding bar used to indicate the connectivity of the current network solution as a percentage of the overall network connectivity. During an active optimisation process, the sliding bar is updated dynamically to indicate that the process is running in the background.



*Solution status*: a status button which enables the user to quit an active process (see Section 4.2.1).

*Network cost*: used to indicate the cost of the current network solution. As with the sliding bar, this field is dynamically updated during an active optimisation process.

### **5.1.5 Dialog Boxes**

A dialog box takes the form of a pop-up window which enables the user to access or specify information relating to an active design procedure, e.g. opening files, editing inter-nodal constraints, etc. Used extensively throughout the application, they can contain control functionality, such as option buttons, information editing fields, and scroll bars. They are also used to display error messages and user prompts.

### **5.1.6 Interaction with the Network Solution**

As discussed in Section 5.1.1, the network designer can interact with the network solution by clicking on any active network element displayed. As a result, all relevant edit and view options can be provided to the user by clicking on a network element. When clicked, the network element is highlighted and a list of options is presented in the form of a pop-up menu. While some options, such as remove, are implemented upon selection, others, such as edit constraints, require additional user interaction. The following is the list of options for each element type.

#### **5.1.6.1 Link options**

**Remove**: (see Section 5.1.2.2).

**Paths Using**: (see Section 5.1.2.3).

#### **5.1.6.2 Node options**

**Remove**: (see Section 5.1.2.2).

**Move**: Upon selection, the user can drag and drop the node to a new location within the design field. It is noted that the inter-nodal distance matrix must also be updated (see Algorithm 5.1).

**Path Trace**: (see Section 5.1.2.3)

**Rename**: Upon selection, the user is presented with an input dialog box which enables him to edit the current name tag.

**Edit Constraints:** Upon selection, a special editing dialog box is opened. It contains a listing of all adjacent nodes and a choice of editing functions (see Section 5.3).

## 5.2 Initialising a New Input Network

This section presents an overview of the interactive procedure used to create and initialise a new input network in NetOpt. As discussed in the previous section, the user has a choice of two network types, i.e. Access or Core, which can be selected from either the *File* menu or tool bar. Once the network type has been selected, the user is guided through the following interactive steps to initialise and define the input network:

*Step 1.* Initialise the Active Network Domain

*Step 2.* Initialise the Number of Nodes

*Step 3.* Initialise the Nodal Characteristics

*Step 4.* Initialise the Inter-Nodal Constraints

*Step 5.* Initialise the Network Cost Components

*Step 6.* Initialise the Network Description

*Step 7.* Save the Network to File

### 5.2.1 Initialise the Active Network Domain

Although the Access and Core network domains are given default settings of (100 x 100) km<sup>2</sup> and (400 x 400) km<sup>2</sup>, respectively, these dimensions can be increased to accommodate larger network domains. An input dialog box, which prompts the user to confirm or update the default settings, i.e.  $x_{\max} \times y_{\max}$  km<sup>2</sup>, is therefore presented. For both network types, the dimensions are rounded up to the nearest 100 km. That is, for a network domain measuring (140 x 330) km<sup>2</sup>, the designer would specify a domain measuring (200 x 400) km<sup>2</sup>. The active design field is then configured according to these settings. Although fixed for the remainder of the initialisation process, these dimensions can be increased at any subsequent point in the design process if required.

### 5.2.2 Initialise the Number of Nodes

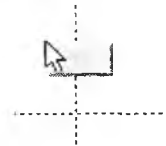
Once the active domain has been defined, the user is presented with a new input dialog box, prompting specification of the number of network nodes. The input value is saved as the active node variable  $n$ . This value is used to define the dimensions of the various nodal arrays and inter-nodal

matrices for the remainder of the process. While it is assumed that the number of nodes will remain constant over the entire design process, NetOpt does facilitate the option to insert or remove any number of nodes after the initialisation process has been completed.

### 5.2.3 Initialise the Nodal Characteristics

Once the number of nodes has been defined, the network designer must specify the location of each node based on the active network domain defined in Step 1. NetOpt provides two options for nodal placement: *User Defined* or *Random*.

In normal operation, since it is assumed that the nodal locations are known in advance, the network designer would select the *User Defined* option. In this mode of operation, the user must position the arrow pointer and click on the desired location within active network domain. As discussed in Section 5.1.1, the active design field and various display features, such as zoom, scroll and pointer coordinates, facilitate this procedure. In addition to these display features, a drag and drop node (as shown opposite) is tagged to the arrow pointer to provide a visual aid. For each nodal placement  $i$ , the  $(x_i, y_i)$  coordinates are translated and stored in the respective nodal arrays. At this point, an input dialog box is presented which enables the user to specify a four-character name tag for each node. As a default, the name tag is the letter  $N$  followed by the current nodal index number  $i$ . As with previous steps, once the initialisation process has been completed the user can edit these nodal characteristics.



The *Random* option for the placement of nodes was incorporated into NetOpt to facilitate the generation of random test case network infrastructures. On selecting this option, NetOpt will generate the coordinates for each node according to the following algorithmic procedure:

---

#### ALGORITHM 5.1: RANDOM NODE PLACEMENT ALGORITHM

*Step 1.* For all nodes  $i$ , let the  $(x_i, y_i)$  coordinates be uniformly distributed on the 2-dimensional space,  $[0, x_{\max}] \times [0, y_{\max}]$ . Note: all coordinates are given to the nearest meter.

*Step 2.* End.

---

### 5.2.4 Initialise the Inter-Nodal Constraints

This step of the process enables the network designer to initialise the various inter-nodal input constraints, i.e. demand requirements, fixed costs, and distances. As with the previous step, the user is given the option of specifying real input data or randomly generating this data.

If the user-defined option is selected, the inter-nodal constraints are initialised to default values according to the following algorithmic procedure:

---

**ALGORITHM 5.2: DEFAULT INITIALISATION ALGORITHM**

*Step 1.* For all inter-nodal links  $l_{i,j}$ , where  $i, j = 1, \dots, n$ , then:

- (i) Let  $r_{i,j} = 1 \text{ E-1}$  or  $\text{STM-1}$  (for Access or Core networks, respectively)
- (ii) Let  $f_{i,j}^{fixed} = 10,000 \text{ £/km}$
- (iii) Let  $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \text{ km}$

*Step 2.* End.

---

Once the initialisation process has been completed, the user can edit these constraints using the node editing utilities discussed in section 5.3.2.

For the random option, these constraints are initialised according to the following algorithmic procedure:

---

**ALGORITHM 5.3: RANDOM INITIALISATION ALGORITHM**

*Step 1.* Through a series of input dialog boxes, the user is requested to input the upper and lower bounds on the various inter-nodal constraints. The following input parameters are specified:

- (i)  $r_{min}$  and  $r_{max}$ , given in E-1s or STM-1s for Access or Core networks, respectively
- (ii)  $f_{min}^{fixed}$  and  $f_{max}^{fixed}$ , given in 1,000 £/Km
- (iii)  $d_{min}$  and  $d_{max}$ , which must be between the range [1, 1.5]

*Step 2.* For all inter-nodal links  $l_{i,j}$ , where  $i, j = 1, \dots, n$ , then:

- (i) Let  $r_{i,j}$  have a uniform distribution on  $[r_{min}, r_{max}]$
- (ii) Let  $f_{i,j}^{fixed}$  have a uniform distribution on  $[f_{min}^{fixed}, f_{max}^{fixed}]$
- (iii) Let  $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \cdot \delta$ , where  $\delta$  has a uniform distribution on  $[d_{min}, d_{max}]$

*Step 3.* End.

---

## 5.2.5 Initialise the Cost Components

Once the inter-nodal constraints have been initialised, the designer must specify the network cost components:  $f^{fibre}$  and  $f^{terminal}$ . Defined to the nearest £100/STM-1, the fibre and terminal cost components are given default settings of £1000/STM-1 and £5000/STM-1, respectively. For each cost component, an input dialog box is presented to the user which prompts him to confirm or

update these default settings. Again, these settings can be modified at any point after the initialisation process has been completed.

### 5.2.6 Initialise the Network Description

The user is provided with an input dialog box that enables him to enter the network description.

### 5.2.7 Save the Network to File

Before the user is requested to save the network to file, the following algorithmic procedure is called to generate the initial network solution:

---

**ALGORITHM 5.4: INITIAL NETWORK SOLUTION ALGORITHM**

*Step 1.* For all  $l_{i,j}$ , let  $a_{i,j} = y^l$ , i.e. initialise a fully connected network configuration.

*Step 2.* Apply the BASIC NETWORK COST ALGORITHM to generate the cost of the initial network solution,  $s^{cost} = F(A, WC)$ .

*Step 3.* Initialise the network cost and connectivity status fields, and refresh the display.

*Step 4.* End.

---

A Save As dialog box is then presented, which enables the user to save the newly defined network to file. Control is then returned to the user and the active network is displayed.

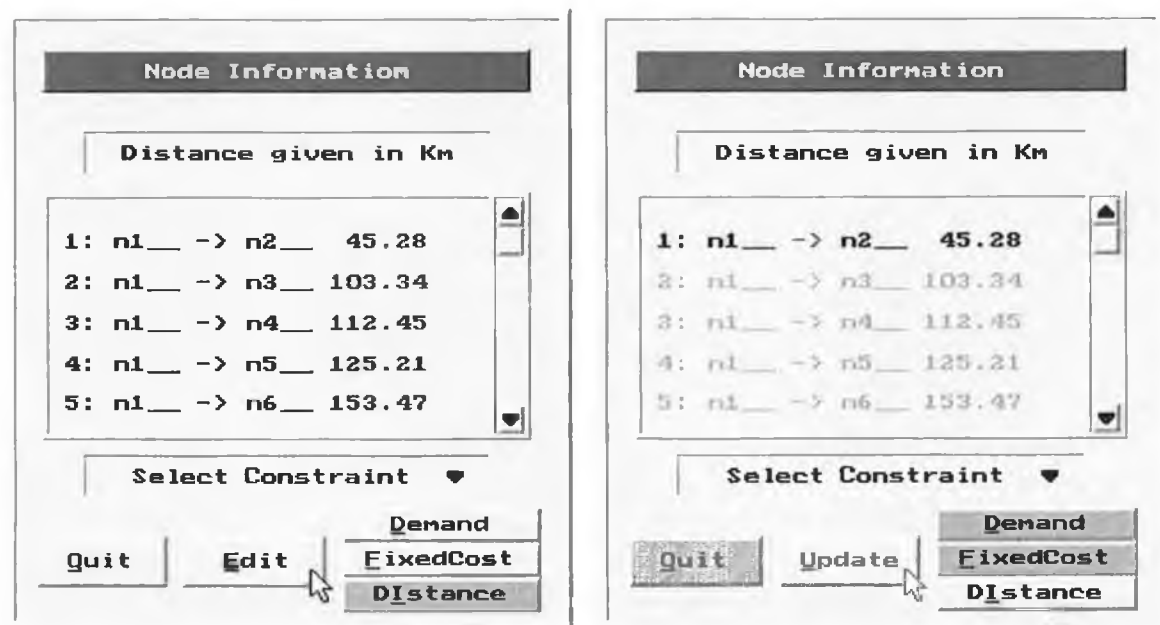
## 5.3 Network Editing Utilities

The various network editing utilities enable the network designer to interact with the overall optimisation procedure. The basic editing utilities, such as insert/remove, have been described in previous sections and are not repeated here. However, a number of enhanced editing utilities were also referred to in these sections. These offer the network designer the facility to access and modify the various inter-nodal constraints both locally and on a global basis.

### 5.3.1 Editing Inter-Nodal Constraints (Locally)

The following utility enables the user to edit the inter-nodal constraint on a node-by-node basis. It can be activated from the menu bar, in which case the user is prompted to click on the relevant node. Alternatively, the user can click on the relevant node and select the option from the pop-up menu presented. Once activated, the relevant node remains highlighted and a special editing dialog

box is opened. As shown in Figure 5.2 (a), this pop-up window contains a scrollable list of the inter-nodal links between the selected node and all other nodes in the network. Associated with each link is a constraint field. The various constraints are provided by way of buttons located below the scrollable region. The user can therefore choose which constraint he wishes to access and display by clicking on the relevant button. Once selected, the constraint button is highlighted and the constraint field associated with each link is updated. To edit these constraints the user must click the 'Edit' button. In edit mode, as shown in Figure 5.2 (b), the 'Edit' option is replaced by an 'Update' option, the constraint and 'Quit' options are disabled, and the constraint field is made active. The user can then scroll up or down the list and select the inter-nodal constraint(s) he wishes to edit. Once the required modifications have been made, the user simply selects 'Update' and the relevant matrix entries are updated. If no further changes are required the user can 'Quit' the process and close the editing window.



(a) Select mode

(b) Edit mode

**Figure 5.2** Node editing window

In practical network design problems, this is an important utility. As discussed in section 5.2.4, when initialising a network based on real input data, the initialisation process simply sets the various inter-nodal constraints to default values. This utility can then be used to access these constraints and set them to their proper values. It also provides the network designer with the facility to update these constraints at a later stage if required.

### 5.3.2 Editing Inter-Nodal Constraints (Globally)

The global editing utility was incorporated into the design tool to facilitate the generation of random test case networks and evaluate the effect these constraints have on the various cost components and network characteristics. This utility was used extensively for generating the test case networks presented in Chapter 6, Results and Analysis.

This network editing option is available as a command on the menu bar. Upon selection, the user is provided with the list of constraints. Once the relevant constraint has been selected, the user is requested to set the range on which the new inter-nodal constraints will be uniformly distributed. The algorithmic procedure is similar to that used in initialising random input data, see ALGORITHM 5.3.

### 5.3.3 Updating the Network Solution

As was mentioned earlier, for any editing operation that modifies the network constraints or cost characteristics, an algorithmic procedure is called to update and display the new network solution. This algorithm is stated as follows:

---

#### ALGORITHM 5.5: UPDATED NETWORK SOLUTION ALGORITHM

- Step 1.* Apply the BASIC NETWORK COST ALGORITHM to generate the cost of the new network solution,  $s^{cost} = F(A, WC)$ .
- Step 2.* Update the network cost and connectivity status fields, and refresh the display.
- Step 3.* End.
- 

## 5.4 Network Optimisation Utilities

This section provides an overview of the network optimisation related menu commands supported by the application, see Section 5.1.2.

### 5.4.1 Design Constraints

This command enables the network designer to access and edit the design constraints, i.e. the optimal routing and restoration constraints. Upon selection, an editing dialog box is presented to the user. It consists of two text fields which show the current constraint settings. Located next to each text field is an 'Edit' button. Located below these is a button prompting the user to 'Confirm' the

current settings. To edit a constraint, the user simply clicks on the relevant *'Edit'* button. Depending on which constraint is selected, an additional choice layer is presented. This enables the user to select from the alternative options associated with each constraint. If a constraint is edited, then the constraint field is automatically updated. Once satisfied with the current settings, the user simply selects *'Confirm'*.

The alternative options associated with each constraint are as follows:

- Optimal Routing Strategy
  - (a) Minimum-hop/shortest-path routing
  - (b) Standard shortest-path routing
- Optimal Restoration Scheme
  - (a) Line-level restoration
  - (b) Path-level restoration

#### **5.4.2 Cost Components**

This command enables the network designer to access and edit the current cost components, i.e.  $f^{fibre}$  and  $f^{terminal}$ . Upon selection, an interactive procedure similar to that used to initialise the cost components is followed (see Section 5.2.5). In the case of inter-nodal fixed costs, these constraints can be edited using the network editing utilities described in Section 5.3.

#### **5.4.3 Network Optimisation**

This command enables the network designer to apply the NETWORK OPTIMISATION ALGORITHM to the active network. Upon selection, the user is asked to confirm the optimisation constraints and cost components. The user is then asked to initialise the process constraints, i.e. the Tabu list size and the number of non-improving moves (iterations) allowed. Once these constraints have been confirmed, the application calls the NETWORK OPTIMISATION ALGORITHM. It is noted that, irrespective of the current network configuration, the input to this process will always be a fully connected network configuration. As such, if the network designer wishes to refine an optimal solution and evaluate the effect of manual changes, the stand-alone DUAL OPTIMISATION ALGORITHM would be selected (see Section 5.4.5). As with all optimisation processes, the user can quit any active process by clicking on the process indicator button located on the status bar.



#### **5.4.4 Initial Optimisation**

This command enables the network designer to apply the INITIAL OPTIMISATION ALGORITHM to the active network. As defined in Chapter 4, this modified Accelerated Greedy algorithm is used to find optimal network solutions that satisfy a required level of connectivity. As such, the user is requested to specify this constraint. The user is also requested to select the starting solution. In regard to this, the user has the option of "Continuing" with the current network solution, or to "Reset" the network as being a fully connected network. While this stand-alone process does not form part of the optimisation procedure, it was incorporated into the package to simulate and analyse the initial phase of this procedure (see next chapter).

#### **5.4.5 Dual Optimisation**

This command enables the network designer to apply the DUAL OPTIMISATION ALGORITHM to the current network solution. As discussed in Section 4.2.2, this design feature facilitates user interaction with the refinement phase of the optimisation procedure. Once the local optimum has been found, the network designer can manually edit the corresponding solution by inserting or removing any number of links. The DUAL OPTIMISATION ALGORITHM can then be applied to this new configuration to see if an improved solution can be found. As will be shown in the next chapter, this forms the basis for the highly effective interactive procedure used in the refinement phase of the overall optimisation procedure.

#### **5.4.6 Optimal Spare Capacity**

This command enables the user to apply the OPTIMAL SPARE CAPACITY ASSIGNMENT ALGORITHM to the current network configuration. As defined in Chapter 4, this includes the relatively exhaustive adaptive minimisation step and is offered as a stand-alone process to evaluate the exact spare capacity assignment costs for any given network configuration.

*Note:* a timing mechanism has also been incorporated into each optimisation process, to evaluate and analyse the computational times.

---

## Chapter 6 - Results and Analysis

---

This chapter concerns an evaluation of the optimal design procedure facilitated by the design tool, based on the analysis of a set of test case networks that model variations in the input constraints. These networks are defined in Section 6.1. Section 6.2 presents a detailed analysis of the impact these input conditions have on the solution cost characteristics for each network, and how these in turn have a residual effect on connectivity characteristics and so the optimal solution cost characteristics. Section 6.3 concerns an evaluation of the optimisation process used to realise the respective solutions presented in Section 6.2. It will be shown how variations in the input constraints and the optimal solution characteristics impact on the computational complexity of each phase of the respective optimisation processes. In particular, it will be shown how the inherent characteristics of the optimal solutions observed in Section 6.2, can be used to substantially improve the efficiency and effectiveness of the computationally intensive refinement phase. Finally, a cost analysis of the alternative routing strategies and restoration techniques is presented in Sections 6.4 and 6.5, respectively.

### 6.1 Generation of Test Case Networks

The objective is to model a range of potential network conditions by varying the input constraints. Two base networks are initially defined, i.e. Networks 1.0 and 2.0. Applying the interactive process used to create and initialise new input networks (see Section 5.2), the following input constraints were defined for each network:

- Domain area: (100 x 100) km<sup>2</sup> and (200 x 200) km<sup>2</sup>, respectively.
- Number of nodes: 12 and 24, respectively.
- Node locations: The RANDOM NODE PLACEMENT ALGORITHM [5.1] was used to generate the respective network infrastructures, i.e. the set of nodal locations within the respective network domains.
- Inter-nodal constraints: The RANDOM INITIALISATION ALGORITHM [5.3] was used to generate the inter-nodal constraints based on the following parameter settings:
  - i) Demands:  $[r_{\min}, r_{\max}] = [0, 1]$ , given in STM-1s
  - ii) Fixed costs:  $[f_{\min}^{\text{fixed}}, f_{\max}^{\text{fixed}}] = [10, 20]$ , given in 1,000 £/km

iii) Distances:  $[\delta_{\min}, \delta_{\max}] = [1.1 - 1.3]$

- Cost components:  $f^{fibre} = 1,000 \text{ £/km}$  and  $f^{terminal} = 10,000 \text{ £/km}$

The network infrastructure and inter-nodal demand requirement settings for the respective networks are typical representations of small to medium-sized metropolitan area networks [9, 10, 17]. The variations in these networks are used to analyse the effect that network size and domain area have on the results, and to show how these results can be extended to other network infrastructures. It is also noted that the cost component settings, including those defined for the inter-nodal fixed costs, represent a correlation of realistic cost values [9, 10, 12, 13, 33].

While these networks model variations in the network infrastructure, it is also important to analyse the effect that variations in the inter-nodal constraints have on the respective results. As such, two alternative versions of each base network, i.e. Networks *x.1* and *x.2*, were generated using the network editing utility described in Section 5.3.2. These networks model increases in the average inter-nodal demand and the fixed costs, respectively. The following parameters were specified for each:

- Demands:  $[r_{\min}, r_{\max}] = [0, 5]$ , given in STM-1s. As the demand requirements contribute to the variable capacity dependent costs, these increases can also model increases in the fibre and terminal costs, or, alternatively, decreases in the fixed costs.
- Fixed costs:  $[f_{\min}^{fixed}, f_{\max}^{fixed}] = [20, 40]$ , given in 1,000 £/km. Using the same reasoning as above, these increases can also model decreases in the fibre and terminal costs, or, alternatively, decreases in the demand requirements that contribute to these costs.

A set of six test case networks have therefore been defined. These are summarised in Table 6.1.

	<b>Nodes</b> $n$	<b>Domain</b> $[x_{\max}, y_{\max}]$	<b>Demands</b> $[r_{\min}, r_{\max}]$	<b>Fixed Costs</b> $[f_{\min}^{fixed}, f_{\max}^{fixed}]$
<b>Network 1.0</b>	12	[100,100]	[0,1]	[10,20]
<b>Network 1.1</b>	12	[100,100]	[0,5]	[10,20]
<b>Network 1.2</b>	12	[100,100]	[0,1]	[20,40]
<b>Network 2.0</b>	24	[200,200]	[0,1]	[10,20]
<b>Network 2.1</b>	24	[200,200]	[0,5]	[20,40]
<b>Network 2.2</b>	24	[200,200]	[0,1]	[10,20]

**Table 6.1** Test case networks

## 6.2 Solution Characteristics

In this section, the correlation between network connectivity and the various network cost characteristics are examined. These correlations are experimentally verified through the analysis of the key benchmark solutions generated for each test case network. In particular, it will be shown how variations in the input constraints impact on the network cost characteristics and how these have a residual effect on the connectivity characteristics of the respective benchmark solutions. Based on this detailed analysis, the effectiveness of the initial optimisation phase shall be verified. To conclude this analysis, the optimal solutions characteristics are considered.

### 6.2.1 Network Connectivity

Before analysing the test results, this section provides some insight into the correlation between the connectivity characteristic associated with each benchmark solution and some general network characteristics. As defined previously, a benchmark topology represents the minimum cost solution (in terms of working capacity assignment costs) that satisfies a required level of connectivity. The connectivity requirement itself defines the minimum number of node disjoint paths that must exist between any pair of nodes in the solution. In terms of survivability, this requirement also defines the minimum number of links that must be removed before any set of nodes in the network become disconnected. The minimum number of links needed to satisfy a required level of connectivity is given by:

$$\begin{aligned} l_{num} &= n-1, & \text{if } c_{req} &= 1 \\ &= \frac{n(c_{req})}{2}, & \text{otherwise} \end{aligned} \tag{6.1}$$

As can be seen from this equation, the connectivity characteristic (number of links) is a function of network size (number of nodes). The proof is quite straight forward. A fully connected network satisfies a connectivity requirement of  $n-1$ . That is, a minimum of  $n-1$  links must be removed before any two nodes become disconnected. Inserting this into equation (6.1) gives:  $n(n-1)/2$ , which is, as expected, the number of links in a fully connected network.

Applying this to the base network infrastructures defined in Section 6.1, the minimum number of links in each benchmark solution is shown below in Table 6.2.

	12-Nodes	24-Nodes
$c_{req}$	$l_{num}$	$l_{num}$
4	24	48
3	18	36
2	12	24
1	11	23

**Table 6.2** Minimum number of links for each benchmark solution

In practice, as the initial optimisation process only considers moves (link removals) that improve the solution cost, the actual number of links in each benchmark solution will tend to exceed the minimum requirement. The extent by which these values are exceeded depends on the required level of connectivity, network size and the solution cost characteristics. We can analyse the correlation between these characteristics by calculating the average connectivity requirement (see Section 6.2.2). This is given by the following equation:

$$\bar{c}_{req} = \frac{2 \cdot l_{num}}{n} \quad (6.2)$$

In terms of the solution cost characteristics, as network connectivity decreases, the fixed cost component will decrease, while the variable capacity dependent costs will increase. To understand the correlation between network connectivity and these cost characteristics, consider the following: Given any network solution, if a link is removed, i.e. the connectivity characteristic is decreased, the fixed cost associated with that link will no longer contribute to the new solution cost, i.e. the fixed cost component will decrease. However, the demand requirements being routed over that link must then be routed over alternative paths. Since the original paths represented the minimum-hop/shortest-path routes, the average hop-count and distance of these new paths will increase. As a result, both the working capacity switched and routing costs will increase. The spare capacity switched and routing costs will also increase, since the number of alternative paths for rerouting the failed working capacity on each link has decreased and the average failed working capacity on each of the respective links has itself increased. These issues form the basis for the results and analysis presented in the following sections.

### 6.2.2 Working Capacity Solution Characteristics

In this section, the correlation between network connectivity and the working capacity solution cost characteristics is examined. The following network and cost characteristics will be analysed:

**Average connectivity requirement:**  $\bar{c}_{req} = \frac{2 \cdot l_{num}}{n}$

**Number of links in each benchmark solution:**  $l_{num} = \sum_{\forall l \in L} a_l$

Note: the minimum number of links for each benchmark solution will also be shown in parenthesis beside the actual solution value.

**Average link distance:**  $\bar{d}_l = \frac{\sum_{\forall l \in L} d_l \cdot a_l}{l_{num}}$

**Average hop-count:**  $\bar{h}_{count}$

Note: this solution characteristic represents the average number of links each inter-nodal path traverses, i.e. the sum of the hop counts for each inter-nodal path divided by the number of inter-nodal paths. It is also noted that, the average number of paths traversing each link in the solution could also be derived from this sum by dividing it by the number of links. While the working capacity switched and routed costs depend on the actual demands being routed and the link distances involved, the basic relationship between network connectivity and these cost characteristics can be extrapolated from these routing characteristics. More significantly, these routing characteristics can be used to evaluate the effect network size has on the key benchmark solutions.

**Working capacity solution cost:**  $F(A, WC) = F_{fixed} + F_{total}^{wc}$

**Total working capacity costs:**  $F_{total}^{wc} = \sum_{\forall l \in L} [d_l \cdot wc_l \cdot f_l^{fixed} + 2 \cdot wc_l \cdot f_l^{terminal}] \cdot a_l$

**Total fixed costs:**  $F_{fixed} = \sum_{\forall l \in L} a_l \cdot d_l \cdot f_l^{fixed}$

We shall initially consider the 12-node network infrastructure, i.e. Networks 1.x, and analyse the effect that variations in the input demand and fixed cost constraints have on these benchmark solution characteristics. These results are presented in Tables 6.3 (a) - (c). Graphs of the benchmark cost characteristics for the respective networks are also shown in Figures 6.1 (a) - (c).

Network 1.0							
$c_{req}$	$\bar{c}_{req}$	$l_{num}$	$\bar{d}_l$	$\bar{h}_{count}$	$F_{fixed}$	$F_{total}^{wc}$	$F(A, WC)$
4	4.167	25 (24)	45.3	1.76	14 288	4 138	18 426
3	3.000	18 (18)	40.6	2.11	9 338	4 834	14 172
2	2.167	13 (12)	32.2	2.77	5 803	5 246	11 049
1	2.000	12 (11)	32.0	2.83	4 844	5 425	11 888

(a)  $[r_{min}, r_{max}] = [0, 1]$ ,  $[f_{min}^{fixed}, f_{max}^{fixed}] = [10, 20]$

Network 1.1							
$c_{req}$	$\bar{c}_{req}$	$l_{num}$	$\bar{d}_l$	$\bar{h}_{count}$	$F_{fixed}$	$F_{total}^{wc}$	$F(A, WC)$
4	4.333	26 (24)	46.2	1.67	15 806	19 051	34 857
3	3.500	21 (18)	41.9	1.98	11 172	20 448	31 620
2	2.667	16 (12)	35.8	2.26	7 513	22 145	29 658
1	2.500	15 (11)	35.7	2.29	6 814	22 505	29 319

(b)  $[r_{min}, r_{max}] = [0, 5]$ ,  $[f_{min}^{fixed}, f_{max}^{fixed}] = [10, 20]$

Network 1.2							
$c_{req}$	$\bar{c}_{req}$	$l_{num}$	$\bar{d}_l$	$\bar{h}_{count}$	$F_{fixed}$	$F_{total}^{wc}$	$F(A, WC)$
4	4.167	25 (24)	45.1	1.71	30 679	4 139	34 818
3	3.167	19 (18)	40.8	2.11	21 963	4 409	26 248
2	2.167	13 (12)	31.8	2.85	12 773	5 148	17 980
1	1.000	11 (11)	30.9	2.93	10 612	5 659	16 257

(c)  $[r_{min}, r_{max}] = [0, 1]$ ,  $[f_{min}^{fixed}, f_{max}^{fixed}] = [20, 40]$

Table 6.3 Benchmark solution characteristics

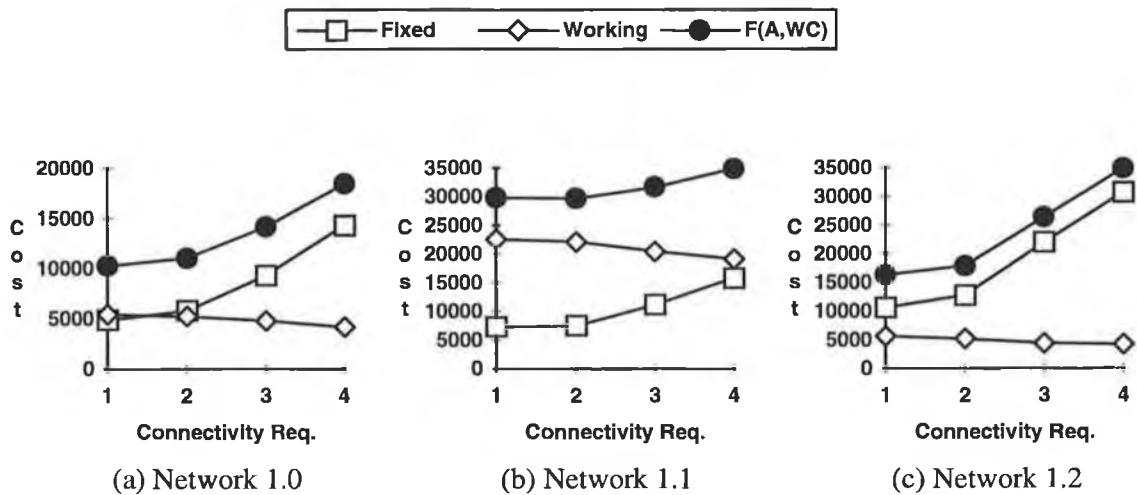


Figure 6.1 Benchmark cost characteristics

As expected, the solution cost,  $F(A, WC)$ , is a decreasing function of network connectivity in all cases. In other words, without the imposition of the connectivity requirement constraint, the local optima for this optimisation problem would tend toward a connectivity characteristic of one. While this was expected, it is of more interest to analyse the effect that variations in the input constraints

have on the average connectivity requirement of these local-optima and the preceding benchmark solutions. To understand this relationship, it is more appropriate to consider the effect these constraints have on the individual cost components of the respective solutions.

In all cases, as network connectivity decreases, the fixed cost component decreases, while the working capacity cost component increases. However, due to the variations in the input constraints, the relative contribution each cost component makes to the respective benchmark solutions varies dramatically. By comparing the effect each constraint has on these cost characteristics, the residual effect on the overall costs and the connectivity characteristics can be derived. Taking Network 1.0 as the basis for this comparison, it can be seen that the fixed cost component dominates at the higher connectivity levels. It can also be seen that the rate of change in this cost component is much greater than that of the working capacity cost component. In other words, as network connectivity decreases, the cost benefit of removing a link, in terms of its fixed cost component, is far greater than the additional cost of rerouting the original link demands. As a result, the number of links in each benchmark solution tends toward the minimum requirement. Finally, as the fixed costs are steadily decreasing and the working capacity costs are gradually increasing, there is a point where the latter becomes the dominant cost component. The average connectivity requirement for this transition is approximately two.

Referring to the results for Network 1.1, analyse the effect that an increase in the average demand requirements has on these characteristics can now be analysed. It can be seen that the working capacity cost component dominates over the entire range of solutions and, since the removal of any link increases this cost component and reduces the fixed costs, this dominance increases as network connectivity decreases. This has a residual effect on the actual connectivity characteristic of the respective solutions. It can be seen that the number of links in each solution exceeds the minimum requirement quite substantially, and as the connectivity requirement decreases, this characteristic increases. As a result, the average connectivity requirement for the local optima is much greater than that of the base network.

Considering to the results for Network 1.2, the effect that an increase in the average fixed costs has on these characteristics can now be considered. As expected, in comparison to the base network, the fixed cost component of each benchmark solution has increased substantially and is therefore dominant over the entire range of solutions. In other words, the cost of removing a link always represents a cost improvement. As a result, the connectivity characteristic for the local optima also represents the minimum requirement. However, it is interesting to note that the connectivity characteristic for the preceding benchmark solutions does not converge to the minimum requirement. This is due to the feasible layout of the network. To illustrate this, consider the



network solutions shown in Figure 6.2. Figure 6.2 (a) shows the actual dual-connected benchmark solution for Network 1.2, while Figure 6.2 (b) represents the minimum connectivity requirement solution. Due to the dominance of the fixed cost component, it would be assumed that this would represent the optimal solution. However, due to the exaggeration of its link configuration, i.e. trying to adopt a ring topology, the former solution actually represents the optimal configuration. Moreover, as network size increases this characteristic becomes even more pronounced. This is illustrated in Figure 6.3, where the corresponding solutions for the 24-node network are shown, i.e. Network 2.2.

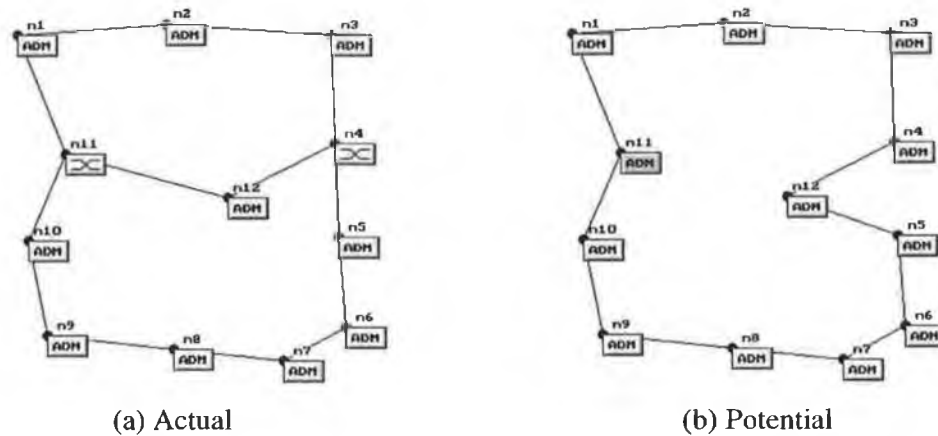


Figure 6.2 Dual-connected benchmark solutions for Network 1.2

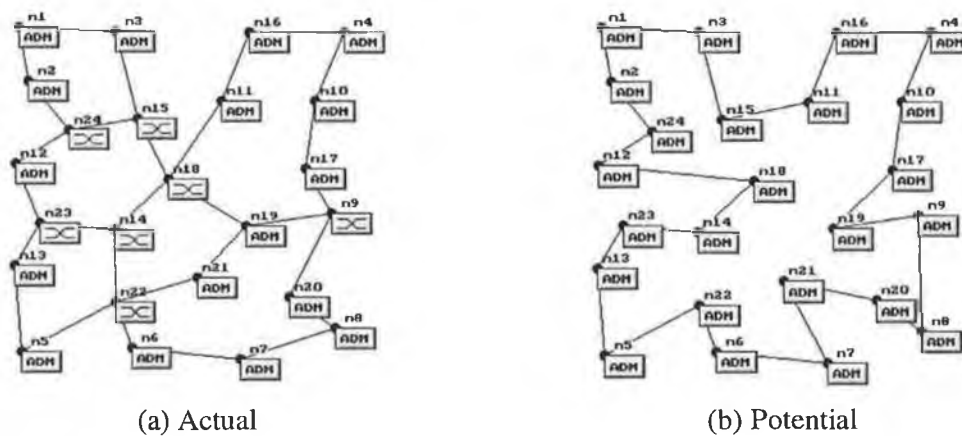


Figure 6.3 Dual-connected benchmark solutions for Network 2.2 (see results below)

Finally, in terms of the average hop-count and average link distance characteristics, the following observations can be made: As expected, the average hop-count is an increasing function of network connectivity in all cases (see Section 6.2.1). In regard to the average link distances, in general, for each move, links representing longer span lengths will tend to be removed. Hence the average link

distance is a decreasing function of connectivity. For a fixed network infrastructure, it is important to note that these characteristics are independent of variations in the input demand and fixed cost constraints. That is, given any link configuration, the routing characteristics and link distances will be the same irrespective of the demands being routed over these links or the fixed costs involved. As such, these characteristics are more useful in analysing the effect network size and domain size have on the solution characteristics. With this in mind, the 24-node network infrastructure will now be considered, i.e. Networks 2.x. These results are presented in Tables 6.4 (a) - (c), with graphs of the benchmark cost characteristics for the respective networks shown in Figures 6.4 (a) - (c).

<b>Network 2.0</b>							
$c_{req}$	$\bar{c}_{req}$	$l_{num}$	$\bar{d}_l$	$\bar{h}_{count}$	$F_{fixed}$	$F_{total}^{wc}$	$F(A, WC)$
<b>4</b>	4.167	50 (48)	64.4	2.49	47 221	28 834	<b>76 055</b>
<b>3</b>	3.417	41 (36)	60.8	2.75	33 939	30 594	<b>64 533</b>
<b>2</b>	2.583	31 (24)	54.6	3.26	23 117	33 268	<b>56 384</b>
<b>1</b>	2.000	24 (23)	51.1	3.47	17 485	34 285	<b>51 770</b>

$$(a) [r_{min}, r_{max}] = [0, 1], [f_{min}^{fixed}, f_{max}^{fixed}] = [10, 20]$$

<b>Network 2.1</b>							
$c_{req}$	$\bar{c}_{req}$	$l_{num}$	$\bar{d}_l$	$\bar{h}_{count}$	$F_{fixed}$	$F_{total}^{wc}$	$F(A, WC)$
<b>4</b>	4.500	52 (48)	65.3	2.47	49 810	134 792	<b>184 602</b>
<b>3</b>	3.750	44 (36)	62.2	2.67	38 321	139 377	<b>177 698</b>
<b>2</b>	3.083	37 (24)	58.3	2.97	30 523	145 105	<b>175 628</b>
<b>1</b>	3.083	37 (24)	58.3	2.97	30 523	145 105	<b>175 628</b>

$$(b) [r_{min}, r_{max}] = [0, 5], [f_{min}^{fixed}, f_{max}^{fixed}] = [10, 20]$$

<b>Network 2.2</b>							
$c_{req}$	$\bar{c}_{req}$	$l_{num}$	$\bar{d}_l$	$\bar{h}_{count}$	$F_{fixed}$	$F_{total}^{wc}$	$F(A, WC)$
<b>4</b>	4.083	49 (48)	59.1	2.75	85 264	27 951	<b>113 215</b>
<b>3</b>	3.250	39 (36)	58.8	2.94	65 965	30 752	<b>96 717</b>
<b>2</b>	2.417	29 (24)	53.4	3.46	43 262	34 138	<b>77 525</b>
<b>1</b>	1.000	23 (23)	50.8	3.47	64 948	36 281	<b>69 352</b>

$$(c) [r_{min}, r_{max}] = [0, 1], [f_{min}^{fixed}, f_{max}^{fixed}] = [20, 40]$$

**Table 6.4** Benchmark solution characteristics

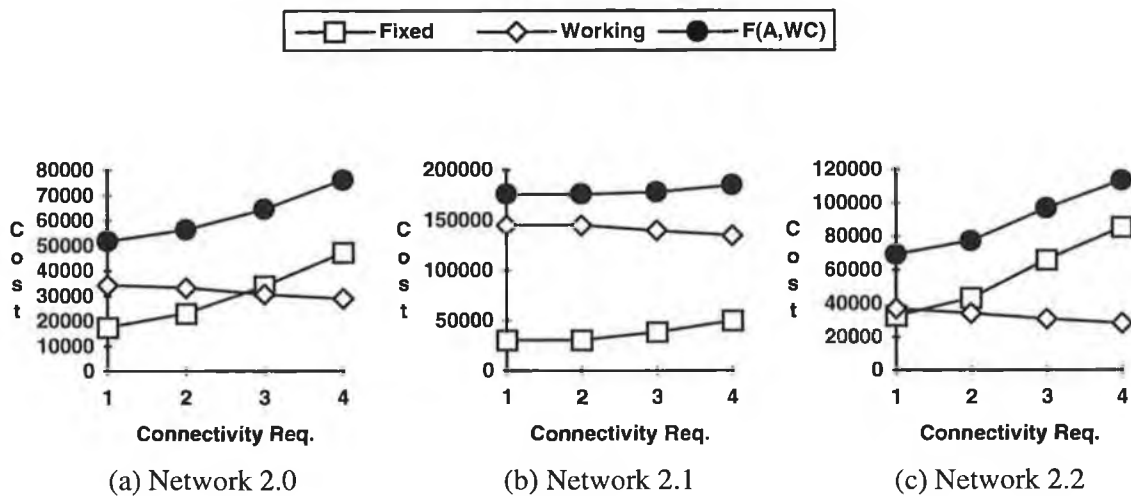


Figure 6.4 Benchmark cost characteristics

The main observation is that, in comparison to the previous network infrastructure, the relative contribution of the working capacity costs has increased in all cases. As a result, the connectivity characteristics of the respective benchmark solutions has increased, i.e. comparing Network 1.0 with Network 2.0, etc. This is due to the effect network size has on the routing characteristics. To illustrate this, consider the benchmark solutions that satisfy the initial connectivity requirement for the inter-related Networks 1.0 and 2.0. In both solutions, the average connectivity requirement is the same, i.e. 4.167. In terms of the routing characteristics, the average hop-count for each solution is 1.76 and 2.49, respectively. However, these values are relative to the network, i.e. the number of inter-nodal paths<sup>27</sup>. The total number of capacity channels being routed in each case is therefore 116 and 694, respectively. This represents a ratio of 1:6. Since the actual working capacity cost ratio is 1:7, the relative increase can be related to the increase in the routing characteristic. In addition, since the actual costs depend on the average link distances involved, the variance in these values can be attributed to this characteristic.

The correlation between the respective fixed cost components can also be derived from the average link distance characteristics. Taking the above two solutions again, it can be seen that the number of links in them is 25 and 50 respectively. Assuming that the uniform distribution of fixed costs associated with these links is approximately the same, the fixed cost ratio would be at least 1:2. However, the network domain size and the distribution of the network infrastructure within the

<sup>27</sup>The number of inter-nodal paths for the 12 and 24 node networks are 66 and 276 respectively, i.e. the number of links in the fully connected networks.

respective domains must also be considered. As the domain size of the 24-node network is four times that of the 12-node network, and since twice the number of nodes are being distributed within this domain, the average link distance ratio would be approximately 1:1.5. Augmenting this with the link ratio, the fixed cost ratio would be approximately 1:3, which is verified by the results. Finally, as the corresponding working capacity cost ratio is approximately 1:6, it can be seen how the relative contribution of the working capacity costs has increased over the range of benchmark solutions.

**Implementation:** All benchmark solutions were generated using the stand-alone initial optimisation process (see Section 5.4.4), while the network analysis utility was used to obtain all the relevant network and cost characteristics (see Section 5.1.2.3).

### 6.2.3 Working and Spare Capacity Cost Characteristics

In this section, the analysis of the benchmark solutions is extended, and the residual effect that network connectivity and the working capacity cost characteristics have on the spare capacity cost characteristics under varying network conditions is considered. The following network and cost characteristics will be analysed:

**Average connectivity requirement:**  $\bar{c}_{req} = \frac{2 \cdot l_{num}}{n}$

**Working capacity routed costs:**  $F_{routed}^{wc} = \sum_{\forall l \in L} a_l \cdot d_l \cdot wc_l \cdot f^{fiber}$

**Working capacity switched costs:**  $F_{switched}^{wc} = \sum_{\forall l \in L} 2 \cdot a_l \cdot wc_l \cdot f^{terminal}$

**Total working capacity switched costs:**  $F_{total}^{wc} = F_{routed}^{wc} + F_{switched}^{wc}$

**Spare capacity routed costs:**  $F_{routed}^{sc} = \sum_{\forall l \in L} a_l \cdot d_l \cdot sc_l \cdot f^{fiber}$

**Spare capacity switched costs:**  $F_{switched}^{sc} = \sum_{\forall l \in L} 2 \cdot a_l \cdot sc_l \cdot f^{terminal}$

**Total spare capacity switched costs:**  $F_{total}^{sc} = F_{routed}^{sc} + F_{switched}^{sc}$

**Percentage of spare capacity redundancy:**  $R_{sc}^{\%} = 100 \cdot F_{total}^{wc} / F_{total}^{sc}$

Note: this solution characteristic is used to express the amount of redundant spare capacity needed to be placed in the network as a percentage of the working capacity requirement.

As with the previous analysis, the 12-node network infrastructure will be considered initially and the effect that variations in the inter-nodal constraints have on these solution characteristics will be analysed. These results are presented in Tables 6.5 (a) - (c), with graphs of the benchmark capacity cost characteristics for the respective networks shown in Figures 6.5 (a) - (c).

Network 1.0		Working Capacity Costs			Spare Capacity Costs			
$c_{req}$	$\bar{c}_{req}$	$F_{routed}^{wc}$	$F_{switched}^{wc}$	$F_{total}^{wc}$	$F_{routed}^{sc}$	$F_{switched}^{sc}$	$F_{total}^{sc}$	$R_{sc}^{\%}$
4	4.167	2 818	1 320	<b>4 138</b>	1 521	660	<b>2 181</b>	<b>53 %</b>
3	3.000	3 234	1 600	<b>4 834</b>	2 308	1 120	<b>3 428</b>	<b>71 %</b>
2	2.167	3 206	2 040	<b>5 246</b>	4 089	2 580	<b>6 669</b>	<b>127 %</b>
1	2.000	3 345	2 080	<b>5 425</b>	4 178	2 720	<b>6 898</b>	<b>165 %</b>

$$(a) [r_{min}, r_{max}] = [0, 1], [f_{min}^{fixed}, f_{max}^{fixed}] = [10, 20]$$

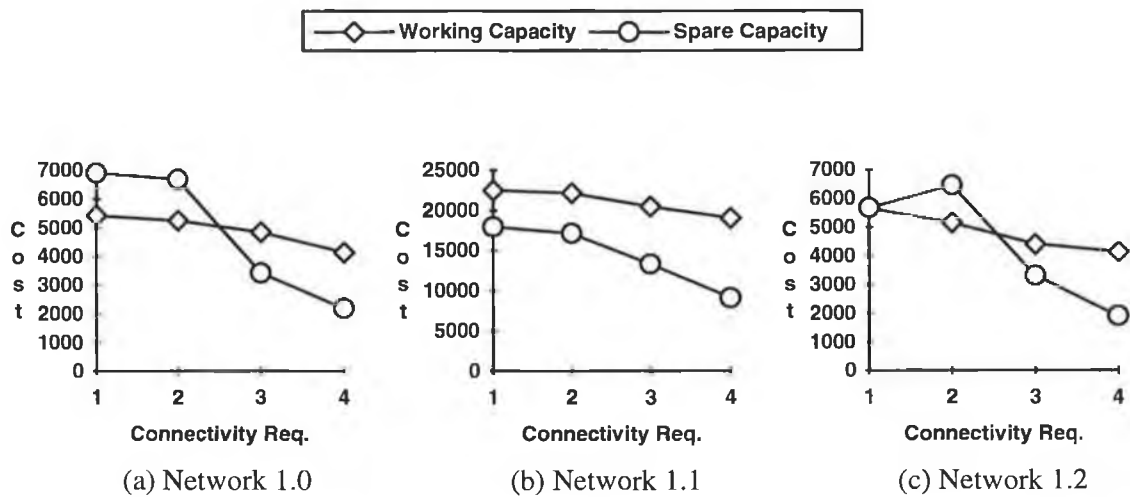
Network 1.1		Working Capacity Costs			Spare Capacity Costs			
$c_{req}$	$\bar{c}_{req}$	$F_{routed}^{wc}$	$F_{switched}^{wc}$	$F_{total}^{wc}$	$F_{routed}^{sc}$	$F_{switched}^{sc}$	$F_{total}^{sc}$	$R_{sc}^{\%}$
4	4.333	13 251	5 800	<b>19 051</b>	6 269	2 875	<b>9 144</b>	<b>48 %</b>
3	3.500	13 628	6 820	<b>20 448</b>	9 054	4 280	<b>13 334</b>	<b>65 %</b>
2	2.667	14 005	8 140	<b>22 145</b>	10 519	6 600	<b>17 119</b>	<b>77 %</b>
1	2.500	14 245	8 260	<b>22 505</b>	11 034	6 920	<b>17 954</b>	<b>80 %</b>

$$(b) [r_{min}, r_{max}] = [0, 5], [f_{min}^{fixed}, f_{max}^{fixed}] = [10, 20]$$

Network 1.2		Working Capacity Costs			Spare Capacity Costs			
$c_{req}$	$\bar{c}_{req}$	$F_{routed}^{wc}$	$F_{switched}^{wc}$	$F_{total}^{wc}$	$F_{routed}^{sc}$	$F_{switched}^{sc}$	$F_{total}^{sc}$	$R_{sc}^{\%}$
4	4.167	2 859	1 280	<b>4 139</b>	1 319	580	<b>1 899</b>	<b>45 %</b>
3	3.167	2 949	1 460	<b>4 404</b>	2 246	1 060	<b>3 306</b>	<b>75 %</b>
2	2.167	3 169	1 980	<b>5 149</b>	3 306	2 460	<b>6 450</b>	<b>125 %</b>
1	1.000	3 619	2 040	<b>5 659</b>	3 619	2 040	<b>5 659</b>	<b>100 %</b>

$$(c) [r_{min}, r_{max}] = [0, 1], [f_{min}^{fixed}, f_{max}^{fixed}] = [20, 40]$$

**Table 6.5** Benchmark capacity characteristics



**Figure 6.5** Benchmark working and spare capacity cost characteristics

In the previous section, the effect that variations in the input constraints had on the relative contribution the fixed and working capacity cost components made to the respective benchmark solutions, and how these variations had a residual effect on the connectivity characteristics of these solutions, was analysed. As these benchmark solutions provide the input to the optimal spare capacity assignment process, the residual effect of these characteristics on the spare capacity costs can be shown.

As expected, the spare capacity cost characteristics and the corresponding levels of redundancy are increasing functions of network connectivity in all cases. In other words, as network connectivity decreases, the spare capacity costs increase, since the amount of working capacity being routed over each link in the respective solutions has increased. Moreover, since the number of alternative paths for rerouting this failed capacity has decreased, the spare capacity is shared by a reduced number of failure scenarios and the corresponding level of redundancy has also increased.

In terms of network redundancy, it can be seen that the redundancy exceeds 100 %, as network connectivity tends toward two. In other words, the cost of protecting the network is greater than the normal cost of operation. This is a direct consequence of line-level restoration. To illustrate this, consider the dual-connected solution for Network 1.2 shown in Figure 6.6. This represents a logical-ring topology. It can be seen that  $wc_{3,4} = 9$  sets the upper bound on the spare capacity requirement on partial-ring (A), while the same is true of  $wc_{4,5} = wc_{10,11} = 11$  on partial-ring (B).

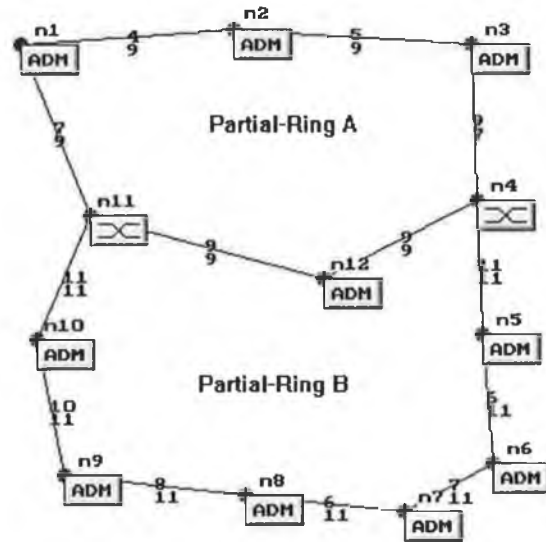


Figure 6.6 Dual-connected benchmark solution for Network 1.2

Finally, it can be seen that for an average connectivity requirement of exactly one, i.e. Network 1.2, the redundancy characteristic reverts back to a minimum of 100 %. In other words, this represents a tree-type topology where all links are protected on a point-to-point basis. The drawback to this is that, in order to protect against link cuts, spare capacity must be assigned to dedicated diversely installed links. As a result, the fixed cost component of this solution increases dramatically.

The 24-node network infrastructure will now be considered and the effect that network size and domain area have on these characteristics will be analysed. These results are presented in Tables 6.6 (a) - (c), with graphs of the benchmark capacity cost characteristics for the respective networks shown in Figures 6.7 (a) - (c).

Network 2.0		Working Capacity Costs			Spare Capacity Costs			
$C_{req}$	$\bar{C}_{req}$	$F_{routed}^{wc}$	$F_{switched}^{wc}$	$F_{total}^{wc}$	$F_{routed}^{sc}$	$F_{switched}^{sc}$	$F_{total}^{sc}$	$R_{sc}^{\%}$
4	4.167	21 914	6 920	<b>28 834</b>	10 349	3 360	<b>13 709</b>	<b>47 %</b>
3	3.417	22 894	7 700	<b>30 594</b>	15 164	4 940	<b>20 104</b>	<b>66 %</b>
2	2.583	24 328	8 940	<b>33 268</b>	23 333	8 600	<b>31 933</b>	<b>96 %</b>
1	2.000	25 045	9 240	<b>34 285</b>	29 179	10 940	<b>40 119</b>	<b>117 %</b>

(a)  $[r_{min}, r_{max}] = [0, 1], [f_{min}^{fixed}, f_{max}^{fixed}] = [10, 20]$

Network 2.1		Working Capacity Costs			Spare Capacity Costs			
$C_{req}$	$\bar{C}_{req}$	$F_{routed}^{wc}$	$F_{switched}^{wc}$	$F_{total}^{wc}$	$F_{routed}^{sc}$	$F_{switched}^{sc}$	$F_{total}^{sc}$	$R_{sc}^{\%}$
4	4.500	101 692	33 100	134 792	58 382	20 140	78 522	58 %
3	3.750	103 197	36 180	139 377	70 799	22 100	92 899	66 %
2	3.083	105 665	39 440	145 105	76 123	28 100	104 223	72 %
1	3.083	105 665	39 440	145 105	76 123	28 100	104 223	72 %

(b)  $[r_{min}, r_{max}] = [0, 5]$ ,  $[f_{min}^{fixed}, f_{max}^{fixed}] = [10, 20]$

Network 2.2		Working Capacity Costs			Spare Capacity Costs			
$C_{req}$	$\bar{C}_{req}$	$F_{routed}^{wc}$	$F_{switched}^{wc}$	$F_{total}^{wc}$	$F_{routed}^{sc}$	$F_{switched}^{sc}$	$F_{total}^{sc}$	$R_{sc}^{\%}$
4	4.083	20 711	7 240	27 951	11 005	3 740	14 745	53 %
3	3.250	22 812	7 940	30 752	14 560	4 960	19 520	63 %
2	2.417	24 485	9 680	34 138	25 440	9 920	35 360	103 %
1	1.000	26 578	10 300	36 878	26 578	10 300	36 878	100 %

(a)  $[r_{min}, r_{max}] = [0, 1]$ ,  $[f_{min}^{fixed}, f_{max}^{fixed}] = [20, 40]$

Table 6.6 Benchmark capacity characteristics

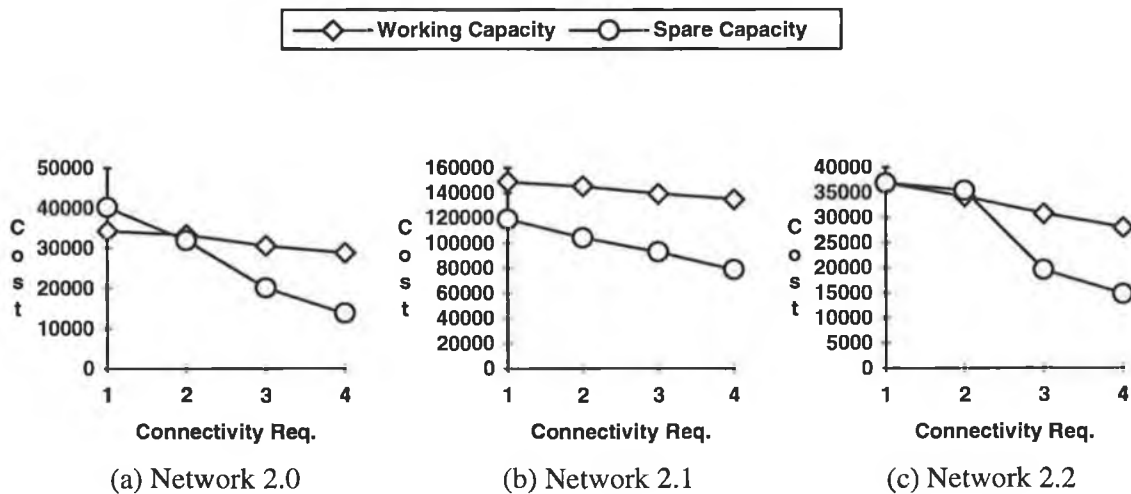


Figure 6.7 Benchmark working and spare capacity costs characteristics

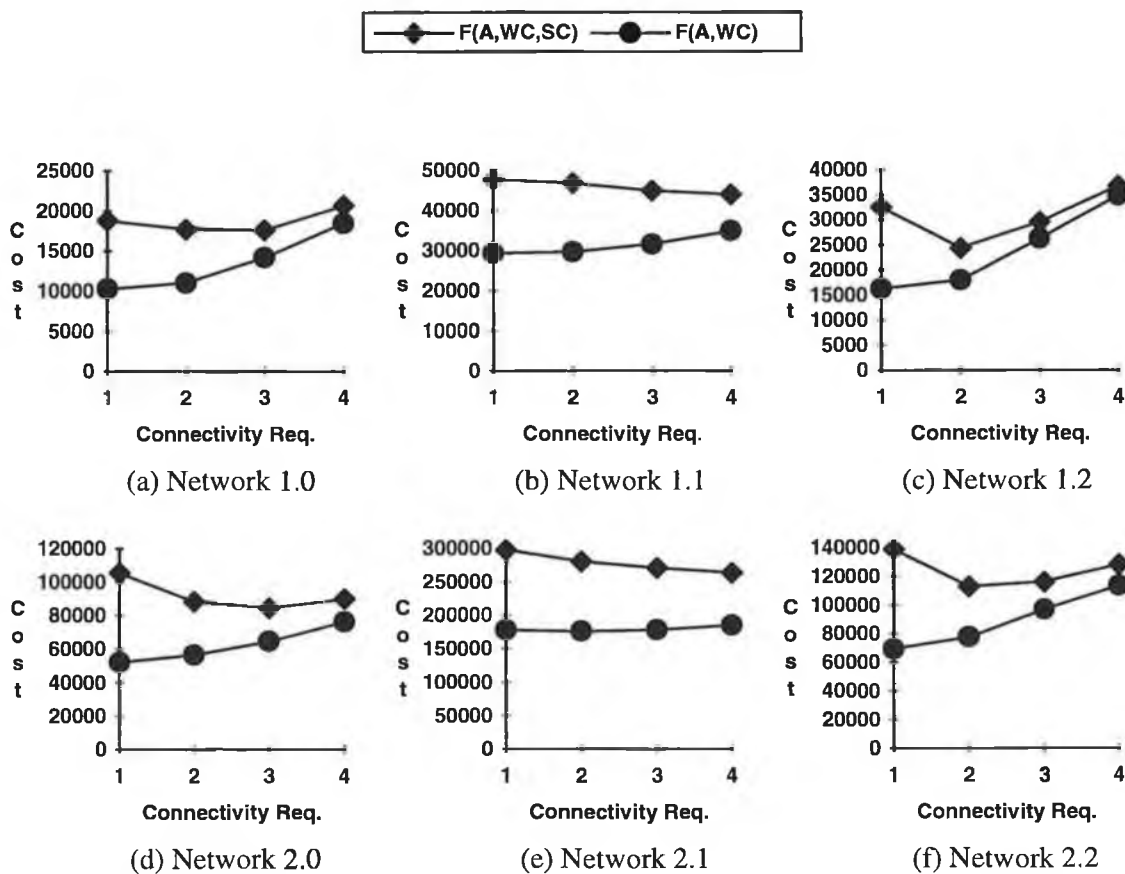
It can be seen that the spare capacity cost characteristics are similar to those obtained for the 12-node network infrastructure. However, due to the increase in the connectivity characteristics of the respective benchmark solutions (see previous section), the corresponding redundancy levels have decreased proportionally.



**Implementation:** The stand-alone optimal spare capacity assignment process was used to evaluate the spare capacity assignment costs for each benchmark topology (see Section 5.4.6).

### 6.2.4 Validation of Initial Optimisation Phase

To validate the initial optimisation phase, a comparison of the working capacity and total network solution cost characteristics for each test case network is presented. These are shown in Figures 6.8 (a) - (f).



**Figure 6.8** Solution costs characteristics

The objective of the initial optimisation phase is to find a more practical starting solution for the more computationally intensive refinement phase. As discussed previously, without the imposition of the connectivity requirement constraint, the local optima for the working capacity optimisation problem would tend toward a connectivity characteristic of one in all cases. It is this condition that enables the initialisation optimisation process to generate the set of benchmark topologies and evaluate the corresponding total network cost characteristics. As a result, the benchmark topology

representing the minimum reference cost can be determined and used as the input to the refinement phase. Depending on the input constraints, it can be seen how the average connectivity characteristic of the respective reference solutions can vary.

The main observation is that the characteristic cost curves for each pair of inter-related networks are quite similar. For both base networks, i.e. Networks 1.0 and 2.0, it can be seen that benchmark topology-3 represents the minimum reference solution. However, it is noted that, due to the relative increase in network size, the average connectivity characteristic of these solutions is 3.000 and 3.417, respectively. In the case of Networks 1.1 and 2.1, due to the increase in the demand requirements, benchmark topology-4 represents the minimum reference solution, while the average connectivity characteristics are 4.333 and 4.500, respectively. Finally, in the case of Networks 1.2 and 2.2, due to the increase in fixed costs, benchmark topology-2 represents the minimum reference solution, while the average connectivity characteristics are 2.167 and 2.417, respectively. A summary of these results is presented in Table 6.7.

Essentially, in terms of the solution space, the reference solutions provide a good indication as to the approximate location of the optimal solution, i.e. the objective of the initial optimisation phase is satisfied.

	$\bar{c}_{req}$	$l_{num}$	$F(s)$
<b>Network 1.0</b>	3.000	18	17 718
<b>Network 2.0</b>	3.417	41	84 637
<b>Network 1.1</b>	4.333	26	44 001
<b>Network 2.1</b>	4.500	54	263 124
<b>Network 1.2</b>	2.167	13	24 430
<b>Network 2.2</b>	2.417	29	112 885

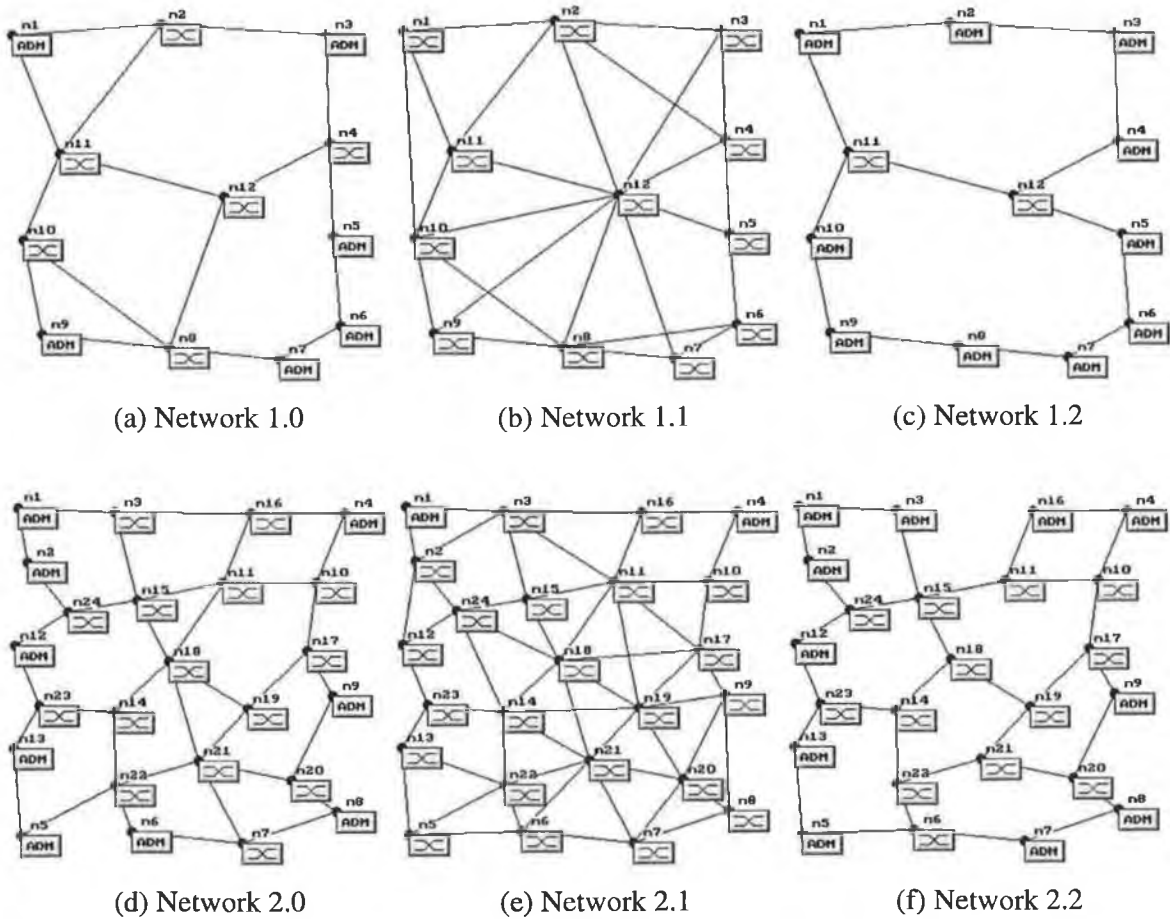
**Table 6.7** Minimum reference solution characteristics

### 6.2.5 Optimal Solution Characteristics

To conclude this analysis section, the optimal solution characteristics are now considered. The average connectivity and cost characteristics for the respective networks are presented in Table 6.8, with graphical representations of the corresponding solution topologies depicted in Figures 6.9 (a) - (f). An evaluation of the refinement process used to realise these solutions is presented in Section 6.3.2.

	$\bar{c}_{req}$	$l_{num}$	$F(s)$
<b>Network 1.0</b>	2.667	16	16 421
<b>Network 2.0</b>	2.917	35	82 125
<b>Network 1.1</b>	4.167	25	40 229
<b>Network 2.1</b>	4.333	52	243 422
<b>Network 1.2</b>	2.333	14	24 068
<b>Network 2.2</b>	2.583	31	110 363

**Table 6.8** Optimal solution characteristics



**Figure 6.9** Optimal network topologies

As in the previous section, due to the correlation between the respective solution characteristics, it is appropriate to consider each pair of inter-related networks separately.

**Networks 1.0 and 2.0:** In comparison to the minimum reference solutions, the average connectivity requirements for both base networks have decreased proportionally. In terms of the optimal configurations, it can be seen that both solutions represent hybrid networks of interconnected ring topologies. While this could have been deduced from the connectivity characteristics, it is interesting to note that the link arrangements within the respective topologies are well balanced. This is consistent for all optimal solutions found, and is related to the optimal assignment and utilisation of spare capacity. As discussed in Section 3.2.3, as a network becomes more balanced, spare capacity is assigned and shared more optimally across the network. In Section 6.3.2, it will be shown how this condition can be used to improve the efficiency and effectiveness of the refinement process.

**Networks 1.1 and 2.1:** As was the case with the base networks, comparing the optimal and reference solutions, the average connectivity requirements have decreased proportionally. However, due to the relative increase in the capacity dependent costs, the connectivity characteristics are still substantially greater than those of the base networks. As a result, both optimal solutions represent a more meshed network topology.

**Networks 1.2 and 2.2:** In comparison to the reference solution characteristics, the average connectivity requirements for both solutions have increased this time. However, due to the relative increase in the fixed costs, these characteristics are still less than those of the respective base networks. As a result, the number of interconnected ring topologies has decreased in both cases.

A summary of these results and the overall conclusions drawn will be presented in the next chapter.

### **6.3 Optimisation Complexity and Efficiency**

This section concerns an evaluation of the overall optimisation procedure in terms of the efficiency and effectiveness of the individual solution phases. Taking the set of test case networks as the bases for this evaluation, it will be shown how variations in the input constraints impact on the complexity of each problem.

In regard to the computational times presented in the following sub-sections, all optimisation processes were performed on a 100 MHz Pentium™ PC.

### 6.3.1 Initial Optimisation Phase

The initial optimisation phase is an integral part of the overall optimisation process. Based on the highly efficient Accelerated Greedy algorithm, it is used to reduce the initial complexity of the problem by finding a more practical starting solution for the more computationally intensive refinement phase. Based on the results presented in Section 6.3.4, the effectiveness of this solution phase has been verified. In terms of the computational time complexity of this process, the average run times for the 12 and 24-node networks are presented in Table 6.9 below. It is noted that variations in the inter-nodal constraints had a negligible impact on these times and are not considered. Comparing these times to the refinement times (see Table 6.11) illustrates the efficiency of this solution phase.

	Networks 1.x	Networks 2.x
Average time (s)	1.8	41.1

**Table 6.9** Initial optimisation process times

### 6.3.2 Refinement Phase

The objective of the refinement phase is to guide the optimisation process out of local optima in order to find a better cost solution. The following steps have been identified:

*Step 1.* As part of the network optimisation algorithm, the dual optimisation process is applied to the minimum reference solution. Based on the process constraints initialised by the user, i.e. the Tabu list size and the number of non-improving moves allowed, an optimal solution is returned.

*Step 2.* Once the local optimal has been found, the user can edit the corresponding solution by inserting or removing any number of links. The stand-alone dual optimisation process can then be applied to this new configuration to see if a better cost solution can be found. This can be applied any number of times, with new process constraints initialised for each step.

It will be shown that the computational time requirement and complexity of this phase is highly dependent on network size and the connectivity characteristics of the optimal solutions. That is, the choice of process constraints selected for each refinement step, and the number of steps required, depends on these network characteristics. The criteria used to select which links to edit is another important factor. However, it will be shown that this is relatively consistent for all networks, and is based on the balanced network condition observed in Section 6.2.5.

The basic unit of complexity is considered first. For each potential move, i.e. iteration of the dual optimisation process, the cost of every solution in the neighbourhood of the current solution must be evaluated. This represents a computational time complexity of order  $O(n^4).l_{num}$ , where the number of solutions is proportional to  $n^2$  and the time complexity for evaluating each solution is of order  $O(n^2).l_{num}$ . The latter is attributed to the complexity of the maximum flow algorithm (see Section 3.3.2). The time complexity for each iteration is therefore a function of network size and connectivity of the current solution. As the search of the solution space will be concentrated around the optimal solution, the average time complexity will be a function of this solution's connectivity characteristic. As shown in Table 6.10, this is consistent with the average iteration times found for the respective networks. As the total computational time is a function of the number of process iterations (moves), the objective would be to minimise this constraint without reducing the effectiveness of the process. This relates to the choice of process constraints and the perturbation techniques used to accelerate the process.

Network Index	1.0	1.1	1.2	2.0	2.1	2.2
Average time (s)	0.6	0.7	0.6	26.4	31.2	24.1

**Table 6.10** Average iteration times

Consider *Step 1* of the refinement phase. As the interactive procedure proved highly effective, the objective of this step was to apply a basic refinement of the reference solution. As such, the number of non-improving moves was set to zero and a local optima closer to the global optimal solution was returned. The computational time for each network process can, therefore, be derived from the respective differences between the reference and optimal solutions' connectivity characteristics multiplied by the average iteration time, e.g.  $(3 \times 0.4 = 1.2)$  s for Network 1.0.

*Step 2* of the refinement phase is now considered. To provide a better understanding of this step, the interactive procedure and the choice of process constraint are dealt with separately.

*Link selection criteria:* Based on the balanced network condition, the initial objective is to detect and remove sub-optimal structures that exist within the solution topology resulting from *Step 1*. For each subsequent step, as the basic sub-optimal structures that violate the balance condition will no longer exist, the objective is to provide some form of perturbation to accelerate the process. The most effective technique involved increasing the connectivity between locally adjacent nodes while maintaining the balance condition. For example, Network 1.0 would be edited in such a way that a new configuration similar to that represented by the optimal solution for Network 1.1 would result. This enabled the process to search a wider range of the solution space, while reducing the

computational time considerably. Moreover, better cost solutions were consistently found in comparison to any other technique considered, i.e. the random selection criterion.

*Tabu list size:* In all cases, best results were obtained for a Tabu list of 5 elements.

*Number of non-improving moves allowed:* Due to the effectiveness of the interactive procedure, it was found that the number of additional iterations for the process could be reduced significantly. Depending on the relative connectivity characteristics, the optimum settings varied between 10 and 30 iterations. It must be emphasised that these are the number of additional iterations and not the total number of iterations for each process. In other words, if 10 links are inserted during the interactive procedure, then approximately this number of moves (iterations) will be performed before any non-improving move is initially found.

*Number of interactive refinement steps:* It was found that the number of interactive refinement steps required for each network depended on the size of the network and the connectivity characteristics of the solution space around the respective optimal solutions. In the case of Networks 1.0 and 1.2, the optimal solution was found after only one interactive step. In the case of Network 1.1, since the connectivity of the optimal solution is greater, two interactive steps were required. In other words, due to the increased connectivity characteristic, there are a greater number of local optima based around the optimal solution to be considered. This condition also impacts on the total number of iterations per process and the average time per iteration, i.e. both increase. In the case of Networks 2.0 and 2.2, although the connectivity characteristics, i.e. values, of the optimal solutions are comparable to their 12-node counterparts, due to the increase in network size, these values represent a relative increase in inter-nodal connectivity and at least two interactive steps were required. Finally, in the case of Network 2.1, as the connectivity characteristic represents a highly meshed network, at least three interactive steps were required. A summary of these results, showing the refinement phase process times for each network, is presented in Table 6.11.

<b>Network Index</b>	<b>1.0</b>	<b>1.1</b>	<b>1.2</b>	<b>2.0</b>	<b>2.1</b>	<b>2.2</b>
<b>Average time (s)</b>	47.1	78.8	191.2	3970.4	7103.9	3048.5

**Table 6.11** Refinement process times

A discussion of these results and overall conclusions drawn will be presented in the next chapter.

## 6.4 Optimal Routing Strategies

In this section, an analysis of the alternative routing strategies is presented. As discussed in Section 3.2.3, the standard shortest-path algorithm does not take into account the distance independent working capacity switched component of the solution costs or the residual effect of the spare capacity costs. It was shown that the former could be minimised by imposing a minimum-hop routing constraint, which minimises the average number of paths being routed over each link and results in a more balanced distribution of the capacity assignments. Consequently, the overall spare capacity requirement is also minimised. As these observations were based on a single instance of a network, this analysis is now extended to consider the effect network connectivity has on these solution characteristics.

### 6.4.1 Working Capacity Cost Characteristics

The differences in the working capacity cost characteristics will first be considered. The set of benchmark solutions for Network 2.0 are used for this analysis. These results are presented in Table 6.12. It is noted that although the effect of variations in the input constraints will also be referred to, it is impractical to show all relevant results.

Network 2.0					
$C_{req}$	Routing Strategy	$\bar{h}_{count}$	$F_{routed}^{wc}$	$F_{switched}^{wc}$	$F_{total}^{wc}$
4	Minimum-Hop/Shortest-Path	2.49	21 914	6 920	28 834
	Standard Shortest-Path	2.62	21 736	7 300	29 036
	% Difference	+ 5.2 %	- 0.7 %	+ 5.5 %	+ 0.7 %
3	Minimum-Hop/Shortest-Path	2.75	22 894	7 700	30 598
	Standard Shortest-Path	2.87	22 750	8 040	30 790
	% Difference	+ 4.4 %	- 0.6 %	+ 4.4 %	+ 0.6 %
2	Minimum-Hop/Shortest-Path	3.26	24 328	8 940	33 268
	Standard Shortest-Path	3.26	24 320	8 960	33 280
	% Difference	+ 0.0	- 0.0	+ 0.0	+ 0.0
1	Minimum-Hop/Shortest-Path	3.47	25 045	9 240	34 285
	Standard Shortest-Path	3.47	25 045	9 240	34 285
	% Difference	0.0	0.0	0.0	0.0

Table 6.12 Working capacity cost characteristics



The standard shortest-path routing strategy selects paths of minimum accumulative link distance irrespective of the number of individual links (hops) involved. In comparison to the minimum-hop routing strategy, this results in an increase in the capacity switched costs and a decrease in the capacity routed costs. The former can be directly related to the increase in the average hop-count. It can be seen that, the relative differences in the routing and cost characteristic are a decreasing function of network connectivity. In effect, the higher the connectivity characteristic, the greater the percentage difference, since there are a greater number of alternative paths for routing the inter-nodal demands. However, as this characteristic decreases, both routing methods will tend toward the same solution, since the optimal path will generally represent the minimum-hop path.

In terms of variations in the input constraints, it was found that the percentage difference in the cost characteristics were proportional to the connectivity characteristic of the respective solutions.

#### 6.4.2 Spare Capacity Cost Characteristics

The differences in the spare capacity cost characteristics shall now be considered. Again, taking Network 2.0 as the basis for this analysis, the relevant results are presented in Table 6.13.

Network 2.0					
$C_{req}$	Routing Strategy	$F_{routed}^{sc}$	$F_{switched}^{sc}$	$F_{total}^{sc}$	$F(s)$
4	Minimum-Hop/Shortest-Path	10 349	3 360	13 709	89 764
	Standard Shortest-Path	11 357	3 680	15 037	90 993
	<b>% Difference</b>	+ 9.7 %	+ 9.5 %	+ 9.7 %	+ 1.4 %
3	Minimum-Hop/Shortest-Path	15 164	4 940	20 104	84 637
	Standard Shortest-Path	15 778	5 060	20 838	85 568
	<b>% Difference</b>	+ 4.0 %	+ 2.4 %	+ 3.6 %	+ 1.1 %
2	Minimum-Hop/Shortest-Path	23 333	8 600	31 933	88 317
	Standard Shortest-Path	23 747	8 720	32 467	88 864
	<b>% Difference</b>	+ 1.8 %	+ 1.4 %	+ 1.7 %	+ 0.6 %
1	Minimum-Hop/Shortest-Path	29 179	10 940	40 119	105 408
	Standard Shortest-Path	29 179	10 940	40 119	105 408
	<b>% Difference</b>	0.0 %	0.0 %	0.0 %	0.0 %

**Table 6.13** Spare capacity cost characteristics

As shown in Section 3.2.3, the standard shortest-path routing strategy results in an unbalanced distribution of working capacity assignments. In comparison to the minimum-hop routing strategy,

this results in an increase in both the spare capacity switched and routed costs. As was the case with working capacity costs, the percentage difference in the respective spare capacity costs is a decreasing function of network connectivity. While the standard shortest-path routing method can result in the same solution costs as network connectivity converges to one, by default it can be concluded that minimum-hop routing represents the optimal routing strategy.

In terms of the total solution costs, the overall cost benefits will depend on the contribution of the capacity dependent cost components, and the relative connectivity characteristics. For example, in the case of Network 2.1, where the working and spare capacity costs dominate the entire range of solutions, the overall cost benefits are substantially greater even at the lower connectivity levels.

## **6.5 Spare Capacity Assignment Algorithm**

### **6.5.1 Quality of Near-Optimal Solution**

One of the most important aspects of this work was the development of an innovative and highly efficient solution method for solving the optimal spare capacity assignment problem. In its basic form, this solution method is capable of generating feasible near-optimal cost solutions. As this is used to generate the set (neighbourhood) of feasible solutions for each iteration of the refinement phase, it is important to verify that these cost solutions are proportional representations of the optimal cost solutions.

Consider the minimum cost solution for Network 1.0, as shown in Figure 6.9 (a). For this analysis, the set of optimal and near-optimal spare capacity solutions within the neighbourhood of this solution were evaluated. Based on these results, a graph of the neighbourhood solution costs worthy of consideration, i.e. within a feasible range of the minimum solution cost, is shown in Figure 6.10. It can be seen that the near-optimal costs are, in fact, a proportional representation of the optimal costs. In other words, the order of potential moves is the same for both sets of solutions.

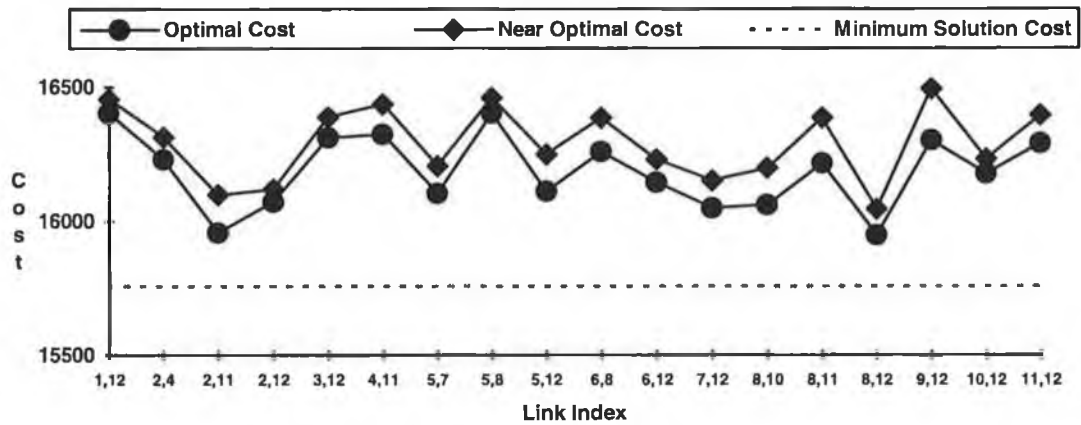


Figure 6.10 Set of neighbourhood solution cost characteristics

### 6.5.2 Path-level and Line-level Restoration

In this section, an analysis of the alternative restoration techniques is presented. As discussed in Chapters 2 and 3, in comparison to line-level restoration, path-level restoration can protect against a greater range of network failures while minimising the spare capacity requirement. However, the overall cost benefits must be such that they justify the increase in operational costs. It would be of interest to examine the effect network connectivity has on these cost characteristics. Table 6.14 shows the respective spare capacity costs characteristics for Network 2.0.

Network 2.0					
$c_{req}$	Routing Strategy	$F_{routed}^{sc}$	$F_{switched}^{sc}$	$F_{total}^{sc}$	$F(s)$
4	Line-Level Restoration	10 349	3 360	13 709	89 764
	Path-Level Restoration	5 576	1 480	7 056	83 111
	% Difference	- 46 %	- 56 %	- 49 %	[6 653]
3	Line-Level Restoration	15 164	4 940	20 104	84 637
	Path-Level Restoration	7 633	2 420	10 053	74 587
	% Difference	- 50 %	- 51 %	- 50 %	[10 050]
2	Line-Level Restoration	23 333	8 600	31 933	88 317
	Path-Level Restoration	16 190	5 600	21 790	78 174
	% Difference	- 31 %	- 35 %	- 32 %	[10 143]
1	Line-Level Restoration	29 179	10 940	40 119	105 408
	Path-Level Restoration	27 831	10 380	38 211	103 500
	% Difference	- 5 %	- 5 %	- 5 %	[1 908]

Table 6.14 Spare capacity cost characteristics

It can be seen that the percentage difference between the spare capacity cost characteristics is a decreasing function of network connectivity, while the overall cost benefits tend to increase to some degree. This is due to the increase in the corresponding levels of redundancy (see Section 6.2.3). It is, however, noted that as network connectivity tends toward one, the relative cost benefits become negligible. A correlation between the cost benefits, network connectivity, and the corresponding levels of redundancy can therefore be made. In terms of the viability of path-level restoration, as research and development in this area is ongoing, it is outside the scope of this work to provide a definitive cost analysis, beyond quantifying the potential gains.

---

## Chapter 7 - Conclusions

---

### 7.1 Overall Discussion of Results

#### 7.1.1 Solution Characteristics

Based on the cost model defined for this problem, it was shown that the solution cost for any given network configuration could be represented by a link-dependent fixed cost component and a variable capacity-dependent-cost component. It was also shown how these cost components were converse functions of network connectivity (the number of links in the network solution). That is, as network connectivity decreased, the fixed costs decreased, while the variable capacity dependent costs increased. Consequently, the connectivity characteristic of the optimal solution depended on the relative contribution (i.e. significance) of these cost components. The correlation between these solution characteristics formed the basis for the highly detailed analysis presented in Section 6.2. In particular, it was shown how variations in the input constraints impacted on these characteristics, and how these in turn had a residual effect on the optimal solution characteristics. The following is a summary of the main observations:

- *Inter-nodal demand sensitivity:* Given the optimal solution for any input network, if the inter-nodal demands for that network are increased, then, due to the relative increase in the capacity dependent costs that result, the connectivity characteristic of the optimal solution will also increase. In other words, while the optimal paths remain unchanged, due to the increase in the demands being routed over these paths, both the working capacity switched and routed costs will increase. In addition, since the average working capacity being routed over each potential link failure has increased, the spare capacity switched and routed costs will also increase. As a result, the cost benefits of inserting a link, in terms of reducing the variable capacity dependent costs, will be greater than the additional fixed cost of installing that link. Obviously, the degree by which the connectivity characteristic increases depends on the relative increase in demands. In addition, since the demands impact on the variable capacity dependent costs, an increase in this input constraint also models the impact of increases in the fibre and terminal costs, or, alternatively, decreases in the fixed costs.
- *Inter-nodal fixed cost sensitivity:* Given the optimal solution for any input network, if the inter-nodal fixed costs for that network are increased, then, due to the relative increase in the overall fixed cost component, the connectivity characteristic of the optimal solution will decrease. That

is, the cost benefits of removing a link, in terms of reducing the fixed cost component, will be greater than the additional costs attributed to rerouting the original link demands. As before, the degree by which the connectivity characteristic decreases will depend on the relative increase in this input constraint. In addition, using the same reasoning as above, an increase in this input constraint also models the impact of decreases in the fibre and terminal costs, or, alternatively, decreases in the demand requirements that contribute to these costs.

- *Network infrastructure sensitivity:* Maintaining the same parameter settings for the inter-nodal demand requirements and fixed costs, if the number of input nodes and domain area are increased, then, due to the relative increase in the capacity dependent costs, the connectivity characteristic of the optimal solution also increases. This is attributed to the effect an increase in the number of nodes has on the routing characteristics. Basically, as the number of nodes increases, the number of the inter-nodal paths and the average hop-count per path increase proportionally. Consequently, both the working and spare capacity switched and routed costs will increase. While it is noted that the average link distances also increase, their relative impact on the fixed cost component is not as significant. Hence, the overall result is an increase in the connectivity characteristic. As before, the degree by which this characteristic increases depends on the relative increase in the input constraints under consideration.

Due to the variance in these input constraints, it was shown that the connectivity characteristics of the optimal working capacity assignment solutions varied between 1.0 and 3.0. That is, as the relative significance of the fixed cost component increased, the connectivity characteristic of the solution tended toward the lower bound, while the inverse was true of the capacity dependent cost component. In the case of the optimal network solutions, since the spare capacity assignment costs were also considered, the relative significance of the capacity dependent cost component increased. Consequently, the respective connectivity characteristics increased proportionally, i.e. they varied between 2.5 and 4.5. As discussed, it was this condition, i.e. the fact that the optimal working capacity assignment solution tended toward a lower bound in comparison to the optimal network solution, that enabled the initial optimisation process to generate the set of benchmark topologies and evaluate the corresponding reference solution costs (see Section 7.1.4.1).

In terms of the optimal network configurations, the topological representation of these solutions can be deduced from the corresponding connectivity characteristics, i.e. the solutions ranged from hybrid networks of interconnected ring topologies to highly meshed network topologies. However, as the design tool also provided a visual representation of these topologies, it was significant to note that, irrespective of the number of links in these optimal solutions, their arrangement within the respective topologies was highly balanced. As discussed, this formed the basis for the highly

effective selection criterion used in the interactive procedure of the refinement phase (see Section 7.1.4.2).

Finally, considerations concerning the practical realisation and economic viability of the various survivable topologies are summarised:

- *DXC-based mesh topologies*: While the results show that DXC-based mesh topologies become more economically viable as the relative significance of the capacity dependent cost component increases, it is noted that some consideration should be given to the practical realisation of these solutions. This concerns the relative increase in management costs and restoration times. For example, in terms of restoration times, as the network size increases, it may not be possible for the current restoration schemes to meet the 2-s service objective for all single link failures (see Appendix A.2). Moreover, due to the more sophisticated management systems required, an additional nodal cost component should be imposed on all DXCs within an optimal solution. This could be modelled by a fixed, distance independent, link cost component which would effectively reduce the number of switched termination points across the network, i.e. decrease the connectivity characteristic of the optimal solution. Although this cost characteristic was not analysed in the results, its effect can be correlated with the effect increases in the inter-nodal fixed costs have on the optimal solution.
- *TM-based point-to-point topologies*: It was interesting to note that TM-based point-to-point survivable topologies did not feature in any of the optimal solutions found. This was attributed to the fact that, in order to protect against link cuts, TM-based point-to-point topologies must be protected by routing dedicated spare capacity over diversely installed links. As a result, both the fixed cost and capacity dependent cost components were such that ADM ring topologies proved more economically viable.
- *Path-level restoration*: In path-level restoration, since failed capacity is rerouted between the terminating nodes of the originating path, it was shown that the spare capacity requirement can be reduced considerably. Moreover, this restoration scheme also provides protection against nodal failures. However, consideration should be given to the additional management costs and increase in restoration times attributed to this relatively more complex restoration scheme. As discussed, while current network management systems and associated equipment costs may not justify near-term deployment, it is expected that enhancements in the former and reductions in the latter will make this restoration scheme a viable alternative for future SDH networks.

### 7.1.2 Optimal Routing Strategies

It was shown that, in comparison to the standard shortest-path routing strategy, the imposition of the minimum-hop routing constraint resulted in the minimisation of the total capacity switched and routed costs. In particular, it was shown that, by imposing this constraint, the average number of paths being routed across the network was minimised and the distribution characteristic of working capacity assignments became more balanced. As a result, the overall spare capacity requirement was also minimised. As discussed, the overall cost benefits were dependent on the contribution of the capacity dependent cost components and the relative connectivity characteristics of the optimal network solutions under consideration.

In terms of the implementation of the algorithmic procedure, it was shown that a separate routing process, having a polynomial time complexity of order  $O(n^2)$ , was used to find the optimal path from any node in the network to all other nodes. As this was applied to all  $n$  nodes in the network, the overall time complexity was of order  $O(n^3)$ . This is comparable to the most efficient algorithms used for solving this problem. In addition, due to the iterative routine used, the existence of an unconnected network solution can be determined during the first iteration. This represents an improvement in performance over existing algorithms.

### 7.1.3 Optimal Spare Capacity Assignment Algorithm

To compare and derive an optimal network solution, the overall optimisation process must at least be capable of generating the optimal spare capacity assignment solution for any feasible network configuration considered. As discussed, due to the complexity of this problem, efficient and/or effective techniques were lacking in this area. This resulted in the development and implementation of an innovative and highly efficient solution method. It was shown that, in its basic form, this solution method was capable of finding feasible near-optimal cost solutions that were proportional representations of the optimal cost solutions. It was this process, which represents a polynomial time complexity of order  $O(n^2) \cdot l_{num}$ , that was used in the overall optimisation procedure.

An additional solution step was also developed to evaluate the optimal cost solution. Based on an adaptive minimisation technique, it was capable of minimising the capacity switched and routed costs. However, as this incurred a relatively large computational overhead, it was only incorporated as a stand-alone process.

Finally, it is noted that both the stand-alone and basic spare capacity assignment processes were also capable of finding optimal and near-optimal cost solutions based on path-level restoration.



## **7.1.4 Overall Optimisation Procedure**

While some work in the area of SDH network optimisation has considered the impact of network reliability and the evaluation of spare capacity assignment costs for a single instance of an optimal working capacity assignment solution, none has considered the survivability constraint explicitly as part of the overall optimisation process. As such, in regard to evaluating the work presented in this thesis, there was no real basis for comparison. However, it is noteworthy that, irrespective of this, considerable effort was made to refine the overall optimisation procedure with particular emphasis placed on efficiency and effectiveness.

### **7.1.4.1 Initial optimisation phase**

As the results verify, this phase of the optimisation procedure proved highly effective in reducing the initial complexity of the problem. Essentially, in terms of the solution space, it was used to find the approximate location of the global optimal solution, i.e. to narrow the search of the solution space to the region around the global optimal solution. The resulting solution was then used as the input to the more computationally intensive refinement phase.

For the initial optimisation process, the main emphasis was placed on efficiency. Based on a modified Accelerated Greedy algorithm, the process took advantage of the fact that the optimal working capacity assignment solution tended toward a lower bound in comparison to the optimal network solution, to generate the set of key benchmark topologies and evaluate the corresponding reference solution costs. The minimum reference cost solution was then used as the input to the refinement phase. In terms of the computational time requirement, since the working capacity assignment costs were only considered for each iteration of the process, and since the monotonicity property held, it was shown that the average run times were negligible in comparison to the run times of the overall optimisation procedures.

### **7.1.4.2 Refinement phase**

Having found the minimum reference solution, the objective of the refinement phase was to guide the optimisation process toward the global optimal solution. As discussed, due to the imposition of the spare capacity assignment problem and the solution cost characteristics that result, it was found that the solution space consisted of a large number of well-distributed sub-optimal solution states. Moreover, it was shown that the computational time complexity of this phase was highly dependent on the number of input nodes and the connectivity characteristic of the optimal solution. As such, in terms of its efficiency and effectiveness, the refinement procedure had to be adapted to the input network under consideration. This related to the link selection criteria used in the interactive

procedure, the choice of process constraints selected for each refinement step, and the number of refinement steps required. The objective was to minimise the latter constraints, while maximising the effectiveness of the process through the interactive procedure.

Due to the effectiveness of the interactive procedure, the first step concerned the basic refinement of the minimum reference solution. It involved finding a local optimal solution closer to the global optimal solution. This was achieved by setting the process constraints, i.e. the Tabu list size and the number of non-improving moves, to zero. As a result, the computational time requirement for this refinement step was kept to a minimum.

With the initial local optimal solution found, the interactive refinement procedure was then applied. It was shown that, depending on the input network under consideration, this procedure consisted of one or more interactive refinement steps with variations in the interactive procedure and choice of process constraints appropriate to that network. The following is a summary of these results:

- *Interactive procedure:* It was shown that the criteria used to select which links to edit was relatively consistent for all networks, and was based on the balanced network condition. For the initial interactive step, the objective was to detect and remove sub-optimal link structures. For each subsequent step, the objective was to provide the perturbation mechanism. As discussed, the most effective technique involved increasing the connectivity between locally adjacent nodes while maintaining the balanced network condition. Essentially, this enabled the process to search a wider range of the solution space close to the global optima, while reducing the computational time requirement considerably. This technique actually highlights the main limitation of random-based perturbation techniques. That is, since links are selected at random, their effectiveness in accelerating the process cannot be guaranteed. Hence, a greater number of iterations per process and process runs would be required. This applies equally to any solution method based on a random selection methodology, e.g. Simulated Annealing and Genetic algorithms.
- *Process constraints:* Due to the effectiveness of the interactive procedure it was found that the process constraints could be kept to a minimum.
- *Number of interactive refinement steps and computational time requirement:* The number of interactive refinement steps was highly dependent on the number of input nodes and the connectivity characteristic of the optimal solution. That is, as these network characteristics increased, the number of local optima based around the optimal solution increased significantly. As a result, a greater number of interactive refinement steps were required to guarantee that the best cost solution was found. It was also shown how increases in these network characteristics impacted on the number of iterations per process and the average run time per iteration, i.e. both

increased significantly. Consequently, the overall computational time requirements for the respective interactive refinement procedures varied quite dramatically, i.e. from one minute for a sparsely connected 12-node network to approximately two hours for a highly meshed 24-node network.

### **7.1.5 NetOpt and the Optimal Design Procedure**

As defined, the design tool NetOpt was developed to facilitate the overall optimal design procedure, i.e. to support the key functional requirements of this procedure. Based on this, the following is a summary of the main features and utilities supported by the tool.

- *Basic Features:* NetOpt is an event-driven GUI application with full mouse support.
- *Active Design Field:* A scrollable window which enables the user to initialise, view, and edit a topological representation of an active network at varied resolutions.
- *Dialog Boxes:* Used extensively, they can contain control functionality, such as option buttons, information editing fields, and scroll bars. They are also used to display user prompts and error messages.
- *Menu and Tool Bar:* Provide access to an extensive range of NetOpt commands and associated utilities which are grouped according to the functional area they support, i.e. File, Edit, View, and Optimisation.
- *Status Bar:* Displays information relating to the active network and application processes.
- *User Interaction:* The user can interact with the current network solution by clicking on any of the active network elements displayed, i.e. nodes and links. When clicked, the network element is highlighted and a list of relevant command options is presented in the form of a pop-up menu.

In conclusion, this work has shown that CAD-based tools, such as NetOpt, are essential for generating and analysing cost-effective survivable SDH networks. It is also noteworthy that, while the primary objective of this prototype design tool was to facilitate the optimal design procedure, it also provided the framework (i.e. environment) and impetus needed to develop, evaluate, and refine every aspect of this procedure.

## **7.2 Recommendations and Scope for Future Work**

### **7.2.1 Further Validation of Results**

As defined, the infrastructure and the inter-nodal demand requirement settings for the base test case networks were typical representations of small to medium-sized metropolitan area networks. It was also stated that the cost component settings, including those defined for the inter-nodal fixed costs, represented a correlation of realistic cost values. However, it was found that variances in the input constraints and, in particular, the cost components did exist. While the network analysis attempted to model these variances, it is recommended that further validation of results should be based on real input data and cost component values. However, as was found during the course of this work, due to the competitive nature of the telecommunications industry, it is difficult to obtain, let alone publish, such confidential information.

### **7.2.2 Further Development of the Optimal Design Procedure**

It was shown that the overall computational time requirement for the optimisation procedure was mainly attributed to the refinement phase, i.e. the time requirement of the initial optimisation phase was negligible in comparison. It was also shown that this time requirement increased dramatically as the number of input nodes and connectivity characteristic of the optimal solution increased. Given this dramatic increase, as network size increases, the optimisation problem soon becomes intractable. As such, further development of the design procedure should take into consideration techniques that could be used to make the problem more manageable.

The most effective of these techniques, and one which is actively employed for solving large-scale network design problems, is based on the concept of facility hubbing and nodal hierarchies [17]. Similar to the concept of the access, regional, and core network exchange hierarchies described in Section 2.4, the idea is to introduce further partitioning (or layering) of these network hierarchies by grouping nodes according to some predefined criteria, e.g. geographical location, community of interest, or traffic loads. The problem with this technique is that the nodal hierarchies (i.e. sub-network infrastructures) and their respective inter-nodal demands must be estimated in the pre-planning phase of the design problem and therefore given as input data. Consequently, while the design procedure can be applied to each sub-network to find the optimal solution, as these are processed independently of each other, they may actually represent sub-optimal structures within the overall network infrastructure. However, as was the case with the basic design procedure, due to the reduction in the computational time requirement, this more modular procedure could be applied a number of times and therefore used to refine the input constraints and the optimal solution states.

---

## References

---

- [1] R. Ballet and Y.C. Ching, "SONET: Now It's the Standard Optical Network," *IEEE Comm. Magazine*, pp 8-15, March 1989.
- [2] H. Kasai et al, "Synchronous Digital Transmission Systems Based on CCITT SDH Standards," *IEEE Comm. Magazine*, pp 50-59, August 1990.
- [3] CCITT Recommendation G.707, "Synchronous Digital Hierarchy Bit Rates," *Blue Book*, 1988.
- [4] CCITT Recommendation G.708, "Network Node Interface for the Synchronous Digital Hierarchy," *Blue Book*, 1988.
- [5] CCITT Recommendation G.709, "Synchronous Multiplexing Structure," *Blue Book*, 1988.
- [6] Balcer et al, "An Overview of Emerging CCITT Recommendations for the Synchronous Digital Hierarchy: Multiplexers, Line Systems, Management, and Network Aspects," *IEEE Comm. Magazine*, pp 21-25, August 1990.
- [7] J. Ash, "Planning an Effective Migration Strategy: The Methodology Behind Successful Interworking," *IRR Conference on Interworking of PDH and SDH Networks*, GPT Network Systems Group, December 1994.
- [8] S. Ferguson, "Implications of SONET and SDH," *Electronics & Communication Eng. J.*, pp 133-151, June 1994.
- [9] R. D. Doverspike et al, "Network Design Sensitivity Studies for the Use of Digital Cross-Connect Systems in Survivable Network Architectures," *IEEE J. Sel. Areas Comm.*, pp 69-78, January 1994.
- [10] S. Hasegawa et al, "Control Algorithms of SONET Integrated Self-Healing Networks," *IEEE J. Sel. Areas Comm.*, pp 110-119, January 1994.
- [11] Y. Kajiyama et al, "An ATM VP-Based Self-Healing Ring," *IEEE J. Sel. Areas Comm.*, pp 171-177, January 1994.
- [12] T. H. Wu et al, "An Economic Feasibility Study for a Broadband VP SONET/ATM Self-Healing Ring Architecture," *IEEE J. Sel. Areas Comm.*, pp 1459-1473, December 1992.

- [13] T. H. Wu and M. E. Burrowes, "Feasibility Study of a High-Speed SONET Self-Healing Ring Architecture in Future Interoffice Networks," *IEEE Comm. Magazine*, pp 33-51, November 1990.
- [14] J. Sosnosky, "Service Applications for SONET DCS Distributed Restoration," *IEEE J. Sel. Areas Comm.*, pp 59-68, January 1994.
- [15] T. H. Wu et al, "The Impact of SONET Digital Cross-Connect System Architecture on Distributed Restoration," *IEEE J. Sel. Areas Comm.*, pp 79-87, January 1994.
- [16] T. H. Wu et al, "SONET Implementation," *IEEE Comm. Magazine*, pp 34-40, September 1993.
- [17] T. H. Wu et al, "Survivable SONET Networks - Design Methodology," *IEEE J. Sel. Areas Comm.*, pp 205-212, January 1994.
- [18] I. Hawker and C.P. Botham, "Distributed Control in SDH Transport Networks," BT Laboratories, 1994.
- [19] R. Holter, "Managing SONET Equipment," *IEEE Networks Magazine*, pp 36-41, January 1992.
- [20] S. Allmis, "Implementing a Flexible Synchronous Network," *IEEE Comm. Magazine*, pp 52-55, September 1993.
- [21] U. Mazzei et al, "Evolution of the Italian Telecommunications Network Towards SDH," *IEEE Comm. Magazine*, pp 44-49, August 1990.
- [22] N.B. Sandesara et al, "Plans and Considerations for SONET Deployment," *IEEE Comm. Magazine*, pp 26-33, August 1990.
- [23] M. Shafi and B. Mortimer, "The Evolution of SDH: A View from Telecom New Zealand," *IEEE Comm. Magazine*, pp 60-66, August 1990.
- [24] B. G. Lee et al, "Synchronous Digital Transmission," *Broadband Telecommunications Technology*, Chapter 3, pp 105-232, Artech House Inc., 1993.
- [25] T. J. Aprille, "Introducing SONET into the Local Exchange Carrier Network," *IEEE Comm. Magazine*, pp 34-43, August 1990.
- [26] R. G. Gareiss and P. H. Heywood, "SONET/SDH: Tomorrow's Networks Today," *Data Comm. - on the Web*, September 1993.
- [27] J. McGibney, "Modern Global Optimisation Heuristics in the long term Planning of networks", *M.Eng. Thesis*, Dublin City University, June 1995.

- [28] T. H. Wu et al, "Computer-Aided Design Procedures for Survivable Fibre Networks," *IEEE J. Sel. Areas Comm.*, pp 1188-1197, October 1989.
- [29] M. Minoux, "Network Synthesis and Optimum Network Design Problems: Models, Solution Methods and Applications," *Networks*, Vol. 19, pp 313-360, 1989.
- [30] J. R. Evans and E. Minieka, *Optimization Algorithms for Networks and Graphs*, Chapters 1-6, Dekker, 1992.
- [31] G. A. Grover et al, "MENTOR: An Algorithm for Mesh Network Topological Optimization and Routing," *IEEE Trans. on Comm.*, Vol. 39, pp 503-513, April 1991.
- [32] A. Gersht and R. Weihmayer, "Joint Optimization of Data Network Design and Facility Selection," *IEEE J. Sel. Areas Comm.*, pp 1667-1681, December 1990.
- [33] M. Kerner et al, "An Analysis of Alternative Architectures for the Interoffice Network," *IEEE J. Sel. Areas Comm.*, pp 1404-1413, December 1986.
- [34] M. Gerla et al, "On the Topological Design of Distributed Computer Networks," *IEEE Trans. on Comm.*, pp 48-60, January 1977.
- [35] B. Gavish et al, "Fiberoptic Circuit Network Design Under Reliability Constraints," *IEEE J. Sel. Areas Comm.*, pp 1181-1187, October 1989.
- [36] D. Bertsekas and R. Gallager, "Routing in Data Networks," *Data Networks*, Chapter 5, pp 363-479, Prentice-Hall, 1992.
- [37] T.H. Cormen et al, *Introduction to Algorithms*, Chapters 17, 25, and 27, McGraw-Hill, 1990.
- [38] A. V. Goldberg and R. E. Tarjan, "A New Approach to the Maximum-Flow Problem," *J. for Computing Machinery*, Vol. 35, pp 921-940, October 1988.
- [39] M. S. Bazaraa et al, *Linear Programming and Network Flows*, Chapters 1-3, 8-9, and 12, Wiley, 1990.
- [40] M. Minoux, "Accelerated Greedy Algorithms for Maximizing Submodular Set Functions," *Proc. IFIP*, pp 234-243, 1977.
- [41] J.F. Mondou et al, "Shortest Path Algorithms: A Computational Study with the C Programming Language," *Computer Ops. Res.*, Vol. 18, pp 767-786, 1991.

## Appendix A1 - DXC-based Restoration Control

DXC-based self-healing is defined as a distributed restoration scheme whose control is initiated and executed locally at each DXC in an autonomous distributed fashion. An alternative restoration technique using centralised control could also be employed. Both techniques define the mechanism used in reconfiguring the DXCs in order to reroute failed capacity. In the case of centralised control, as depicted in Figure A1.1 (a), DXC reconfiguration is performed at a centralised controller (or Operating System), with a large database containing all the relevant network information. In this scheme, when a failure is detected, a fault notification message is sent from each of the affected DXCs to the centralised controller. The OS then processes these messages, extrapolates the fault location, computes an optimal routing table and then downloads reconfiguration requests to the relevant DXCs. While this scheme may optimally reroute the failed capacity, the overall process may take upwards of 10 m and cause unacceptable degradation of services for most customers affected by the failure<sup>1</sup> [9, 10, 14, 15, 18]. In fact, as the size and complexity of these networks increase, centralised control becomes even less attractive, incurring greater response times which in turn reduce the overall resilience of the network.

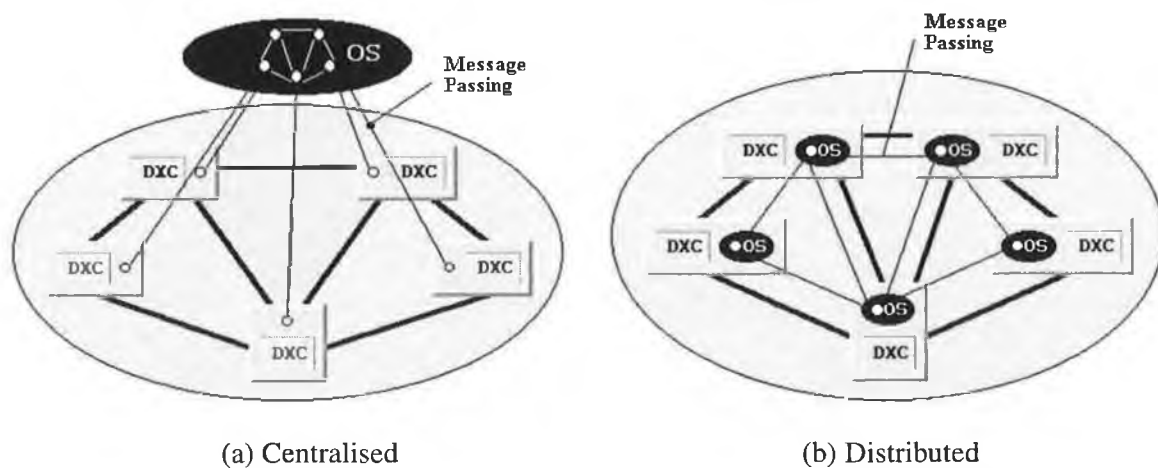


Figure A1.1 DXC-based restoration control

<sup>1</sup>However, this is still much faster than manual DXC reconfiguration (as is the case in PDH-based cross connects in the event of a failure) which may disrupt services for hours or more [14].



As most services are impacted by a service outage greater than 2 s, as discussed in [14, 15], the objective would be to reduce the restoration time down to this permitted level. This was the main motivation for considering alternative DXC reconfiguration schemes such as those based on distributed control. The viability of using DXC-based mesh topologies with distributed control (to meet the 2-s service restoration objective) has been extensively studied [15, 18, 19]. It has been shown that this alternative approach exhibits much faster restorational times and a higher degree of flexibility compared to the centralised control technique.

Distributed control, as depicted in Figure A1.1 (b), involves the exchange of messages between adjacent DXCs in response to a network request/event such as a link failure. This concept of request/event message passing can be applied to other forms of management processing, such as dynamic network reconfiguration (DNR) in response to significant changes in traffic patterns across the network [18]. With distributed control, the intelligence resides locally at each DXC. As such, the DXC controllers act like parallel processors computing alternative routes dynamically in real-time. Each DXC is also capable of storing local information such as the working and spare capacity assignments associated with each of its terminating links. Embedded in each controller is a set of rules/algorithms that interprets alarms or messages received from its adjacent nodes. In other words, there is no need for a large network database or control software as the network acts as its own database<sup>2</sup>. In addition, any new line system or node added to the network is automatically protected, subject to spare capacity availability, since there are no protection plans to modify.

---

<sup>2</sup>While each DXC operates in an autonomous and distributed manner, controlling its entire cross-connect system, interfaces with external operating systems will still exist. This will provide the necessary management facilities used to monitor and administrate the network remotely by the local operators.

---

## Appendix A2 - Distributed Restoration Time

---

The impact on network services (and customers) associated with service downtime is directly related to restoration time. Various parameters that can affect the total restoration time per STM-1 path have been highlighted. In accordance with the relevant studies, such as Bellcore's industry-wide standardisation initiative<sup>3</sup>, these parameters and their relative times are given below:

- Alarm Detection Time 10 ms
- CPU - Message Processing Time 1-10 ms
- Message Transfer Time 1-10 ms
- Cross-Connect Time 100 ms per STM-1 path

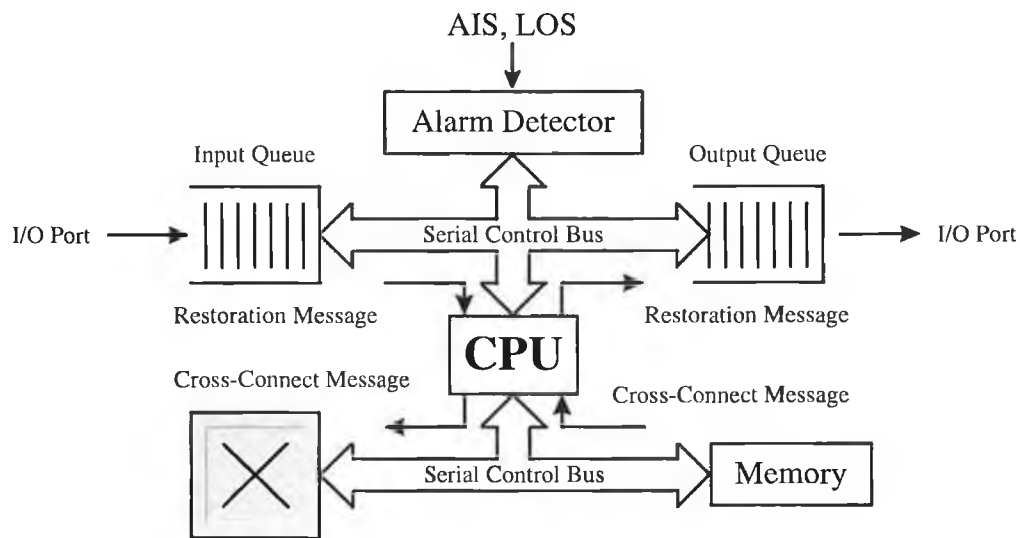
These parameter values relate to the current DXC system architectures and cross-connect technologies which only allow for serial message processing and serial cross-connection. A schematic representation of the DXC system is shown in Figure A2.1.

The alarm detection time is the duration between alarm occurrence and recognition by the CPU. A time constraint of 10 ms is generally assumed for this parameter. The message processing time is part of the restoration algorithm execution time and is largely dependent on the CPU and the required level of processing. According to the studies, the average CPU message processing time is 5 ms, i.e. 1-10 ms. Message transfer time through the DXC depends on the queue delay, the time from input queue to CPU memory and from CPU memory to output queue. This time is assumed to be in the same region as the CPU processing time. It is noted that, each DXC has a received message (input) queue and a sending message (output) queue for each connected link. Finally, the cross-connect time per STN-1 is 100 ms, where map transfer cross-connection<sup>4</sup> is assumed. Results from the various studies indicate that the cross-connect time is the most dominant factor and that mainly this parameter will dictate the total restoration time.

---

<sup>3</sup>This is consistent with other independent studies, such as those conducted by NEC [10].

<sup>4</sup>The cross-connect time includes two phases: 1) the time for computing the new configuration map within the CPU controller; and 2) the time to transfer the new map and update the current configuration map. The latter phase includes spare port verification and testing, and finally the reconfiguration of the cross-connect matrix based on this new map.



**Figure A2.1** Serial processing/cross-connect system

The objective of these studies was to analyse the viability of distributed control schemes to meet the 2-s restoration objective in DXC-based mesh networks. The network models were based on core networks, with between 11 and 15 nodes, an average connectivity of 3.5, and traffic demands ranging from 5 to 40 STM-1s per link. Spare capacity assignments were optimised to ensure 100% restoration against any single link failure. The studies conclude that, using the present serial processing/cross-connect DXC system, it is not possible to meet the 2-s restoration objective for all single link failure scenarios.

To improve the restoration time, two basic enhancements to the process have been identified. The first is to minimise the number of restoration messages or paths needed to be processed. The second is to enhance the DXC processing/cross-connect capabilities. In the first approach, there are two possible ways to reduce the number of paths or restoration messages needed to be processed: bundle restoration or priority restoration. Bundle and priority restoration options primarily impact the existing operation systems, that is, provisioning, administration, and bandwidth management, while the DXC system enhancements will impact on developments in DXC technology, i.e. it is vendor specific.

**Bundle Restoration:** The concept of bundle restoration is to restore a group of STM-1s that have the same path termination nodes with a single restoration message instead of one for each individual STM-1 path. Thus the number of restoration messages and paths needing to be processed within DXCs is significantly reduced. The idea is to create an STM- $N$  frame at the end nodes (for path restoration) to accommodate a maximum of  $N$  affected STM-1s, where  $N$  may be equal to 1, 4, or 16. However, information regarding which STM-1s should be bundled together during the

restoration process needs to be created during the service provisioning phase, and downloaded and stored in each DXC. Thus a database is needed in each DXC to store this information, which must also be updated continuously to ensure the correct paths are restored. As such, keeping track of these dynamic STM-1 assignments and correctly updating the relevant databases may put a strain on the network operations system.

**Priority restoration:** Priority restoration is a more realistic short term solution. The idea is to restore the STM-1 paths on priority basis. This reduces both message processing and DXC cross-connect delay and may provide 2-sec restoration to high-priority services. Unfortunately, while this concept may be simple to implement in a vendor-specific sub-network, it may prove more complex in larger multi-vendor networks. In turn, this may add significant complexity to existing operation systems and require enhanced bandwidth management capabilities. A number of studies have been proposed to determine the exact impact of using priority restoration on existing operations systems.

**DXC system architecture enhancements:** The proposed enhancements to the present DXC system for fast restoration include parallel message processing within the CPU, and the parallel path cross-connection and/or faster reconfiguration capabilities. It is expected that the next generation of DXCs will offer cross-connect times in the region of 10 ms per STM-1 path. Based on existing DXC technology, only small gains were actually achieved using parallel message processing. As expected, the cross-connect processing time is the dominant factor. As a result, increasing the batch size, i.e. the number of parallel processes, in the cross-connect processor results in a linear speed-up of cross-connect times. For example, a 40-channel STM-1 link could be restored within 2 if the batch size was greater than 6.

In simulations, it was also found that using next generation cross-connect technology (10 ms) there is a point beyond which no significant gains in restoration time can be made by increasing the batch size. At this point, the message processors become the dominant factor (or the bottleneck). To overcome this, more processors have to be added or the processing time reduced.

**Other short-to-medium term solutions:** Alternatively, the 2-s restoration objective may be met by overlaying hybrid network restoration topologies. These hybrid networks would be based on point-to-point APS or ADM rings topologies providing fast restoration to meet specific customer needs. The DXC-based self-healing network would then provide more flexible restoration with enhanced survivability against node failures, and adequate restoration time for most other customer requirements.

Another alternative would be to use preplanned DXC restoration. Preplanned restoration may result in faster algorithm execution and, more significantly, faster cross-connection since the cross-connect time may be reduced if the internal route for the cross-connect switching matrix is known in advance. However, the preplanned configuration maps in each DXC must be synchronised and complicated updating procedures applied, the latter putting a large overhead on the network administration and maintenance system. Each DXC would also require a large database in order to store all the relevant failure scenarios. Moreover, should the network state change significantly since the last update, then this approach cannot guarantee the required level of resilience.

One solution, as proposed in [18], would be to combine the dynamic and preplanned techniques in a way that would complement each other. However, it is possible that the additional overhead costs in equipment would not justify the expense. Which approach should be used depends on the cost for these system enhancements, and the revenue expected from services supported by the distributed control DXC network restoration systems.

---

## Appendix A3 - Optimal Routing Algorithms

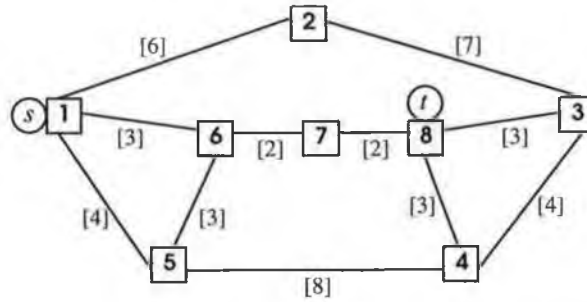
---

### A3.1 Implementation of Dijkstra's Shortest-Path Algorithm

---

**EXAMPLE 1: IMPLEMENTATION OF DIJKSTRA'S SHORTEST-PATH ALGORITHM**

[Must Modify] - To illustrate how this algorithm is implemented consider the network shown in Figure A3.1. The objective is to find the shortest-path between nodes 1 and 8, i.e.  $s = 1$  and  $t = 8$ .



**Figure A3.1** Example network (note: link distances are shown in square brackets)

*Step 1.* Start out by labelling node  $s$  as being checked. Set  $d_1^{path} = 0$ ,  $s_1^{checked} = true$ ,  $p_{1,1} = 1$ . At this point  $D^{path} = [0, \infty, \infty, \infty, \infty, \infty, \infty, \infty]$ ,  $P_1 = [1, x, x, x, x, x, x, x]$  and  $S^{checked} = [1, 0, 0, 0, 0, 0, 0, 0]$ . Let  $w_{node} = s = 1$ .

*Step 2.* Iteration ( $k = 1$ ):

Set  $w_{found} = false$  and  $d_{min} = \infty$ .  $\forall i, i \in N$ , where  $i \neq 1$  and  $s_i^{checked} = false$ :

$$d_2^{path} = \min\{d_2^{path}, d_1^{path} + d_{1,2}\} = \min\{\infty, 0 + 6\} = 6 \text{ and let } p_{1,2} = 1 \quad \textit{\{improved\}}$$

$$d_5^{path} = \min\{d_5^{path}, d_1^{path} + d_{1,5}\} = \min\{\infty, 0 + 4\} = 4 \text{ and let } p_{1,5} = 1 \quad \textit{\{improved\}}$$

$$d_6^{path} = \min\{d_6^{path}, d_1^{path} + d_{1,6}\} = \min\{\infty, 0 + 3\} = 3 \text{ and let } p_{1,6} = 1 \quad \textit{\{improved\}}$$

Result:  $D^{path} = [0, 6, \infty, \infty, 4, 3, \infty, \infty]$ ,  $P_1 = [1, 1, x, x, 1, 1, x, x]$ ,  $w_{found} = true$ ,

$$d_{min}^{path} = d_6^{path} = 3 \text{ and } n_{min} = 6.$$

*Step 3.* Since  $t \neq 6$  and  $w_{found} = true$ , set  $w_{node} = 6$  and  $s_6^{checked} = true$ , and return to *Step 2*.

*Step 2.* ( $k = 2$ ):

$$d_2^{path} = 6 \quad \textit{\{unchanged\}}$$

$$d_5^{path} = \min\{d_5^{path}, d_6^{path} + d_{6,5}\} = \min\{4, 3 + 3\} = 4 \quad \textit{\{unchanged\}}$$

$$d_7^{path} = \min\{d_7^{path}, d_6^{path} + d_{6,7}\} = \min\{\infty, 3 + 2\} = 5 \text{ and let } p_{1,7} = 6 \quad \textit{\{improved\}}$$

Result:  $D^{path} = [0, 6, \infty, \infty, 4, 3, 5, \infty]$ ,  $P_1 = [1, 1, x, x, 1, 1, 6, x]$ ,  $w_{found} = true$ ,  
 $d_{min}^{path} = d_5^{path} = 4$  and  $n_{min} = 5$ .

Step 3. Since  $t \neq 5$  and  $w_{found} = true$ , set  $w_{node} = 5$  and  $s_5^{checked} = true$ , and return to Step 2.

Step 2. ( $k = 3$ ):

$d_2^{path} = 6$  and  $d_7^{path} = 5$  {unchanged}

$d_4^{path} = \min\{d_4^{path}, d_5^{path} + d_{5,4}\} = \min\{\infty, 4 + 8\} = 12$  and let  $p_{1,4} = 5$  {improved}

Result:  $D^{path} = [0, 6, \infty, 12, 4, 3, 5, \infty]$ ,  $P_1 = [1, 1, x, 5, 1, 1, 6, x]$ ,  $w_{found} = true$ ,  
 $d_{min}^{path} = d_7^{path} = 5$  and  $n_{min} = 7$ .

Step 3. Since  $t \neq 7$  and  $w_{found} = true$ , set  $w_{node} = 7$  and  $s_7^{checked} = true$ , and return to Step 2.

Step 2. ( $k = 4$ ):

$d_2^{path} = 6$  and  $d_4^{path} = 12$  {unchanged}

$d_8^{path} = \min\{d_{1,8}^{path}, d_{1,7}^{path} + d_{7,8}\} = \min\{\infty, 5 + 2\} = 7$  and let  $p_{1,8} = 7$  {improved}

Result:  $D^{path} = [0, 6, \infty, 12, 4, 3, 5, 7]$ ,  $P_1 = [1, 1, x, 5, 1, 1, 6, 7]$ ,  $w_{found} = true$ ,  
 $d_{min}^{path} = d_2^{path} = 6$  and  $n_{min} = 2$ .

Step 3. Since  $t \neq 2$  and  $w_{found} = true$ , set  $w_{node} = 2$  and  $s_2^{checked} = true$ , and return to Step 2.

Step 2. ( $k = 5$ ):

$d_3^{path} = \min\{d_3^{path}, d_2^{path} + d_{2,3}\} = \min\{\infty, 6 + 7\} = 13$  and let  $p_{1,3} = 2$  {improved}

$d_4^{path} = 12$  and  $d_8^{path} = 7$  {unchanged}

Result:  $D^{path} = [0, 6, 13, 12, 4, 3, 5, 7]$ ,  $P_1 = [1, 1, 2, 5, 1, 1, 6, 7]$ ,  $w_{found} = true$ ,  
 $d_{min}^{path} = d_8^{path} = 7$  and  $n_{min} = 8$ .

Step 3. Since  $t = n_{min} = 8$ , then end. The shortest path between nodes 1 and 8 has thus been found (see Figure A3.2).

Step 4. Return  $connected = w_{found} = true$ .

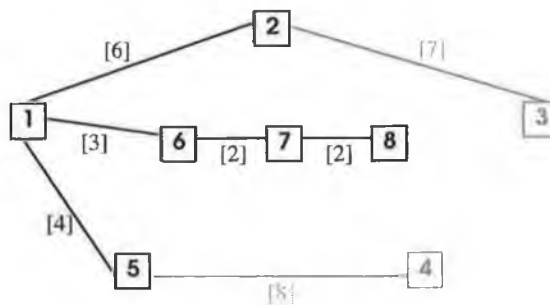


Figure A3.2 Graphical representation of optimal path

## A3.2 Implementation of Modified Shortest-Path Algorithm

### EXAMPLE 2: IMPLEMENTATION OF MODIFIED SHORTEST-PATH ALGORITHM

Continuing from Step 3 iteration ( $k = 5$ ) of EXAMPLE 1:

The status of the various inter-nodal variables and routing constraints up to this point are:  $w_{found} = true$ ,  $d_{min}^{path} = d_8^{path} = 7$ ,  $n_{min} = 8$ ,  $k = 5$ ,  $D^{path} = [0, 6, 13, 12, 4, 3, 5, 7]$  and  $P_1 = [1, 1, 2, 5, 1, 1, 6, 7]$ .

Step 3. Since  $k < n - 1$  and  $w_{found} = true$ , let  $k \leftarrow 6$ , set  $w_{node} = 8$  and  $s_8^{checked} = true$ , and return to Step 2.

Step 2. ( $k = 6$ ):

$$d_3^{path} = \min\{d_3^{path}, d_8^{path} + d_{8,3}\} = \min\{13, 7 + 3\} = 10 \text{ and let } p_{1,3} = 8 \quad \text{\textit{\{improved\}}}$$

$$d_4^{path} = \min\{d_4^{path}, d_8^{path} + d_{8,4}\} = \min\{12, 7 + 3\} = 10 \text{ and let } p_{1,4} = 8 \quad \text{\textit{\{improved\}}}$$

Result:  $D^{path} = [0, 6, 10, 10, 4, 3, 5, 7]$ ,  $P_1 = [1, 1, 8, 8, 1, 1, 6, 7]$ ,  $w_{found} = true$ ,  $d_{min}^{path} = d_3^{path} = 10$  and  $n_{min} = 3$ .

Step 3. Since  $k < n - 1$  and  $w_{found} = true$ , let  $k \leftarrow 7$ , set  $w_{node} = 3$  and  $s_3^{checked} = true$ , and return to Step 2.

Step 2. ( $k = 7$ ):

$$d_4^{path} = \min\{d_4^{path}, d_3^{path} + d_{3,4}\} = \min\{10, 10 + 4\} = 10 \quad \text{\textit{\{unchanged\}}}$$

Result:  $D^{path} = [0, 6, 10, 10, 4, 3, 5, 7]$ ,  $P_1 = [1, 1, 8, 8, 1, 1, 6, 7]$ ,  $w_{found} = true$ ,  $d_{min}^{path} = d_4^{path} = 10$  and  $n_{min} = 4$ .

Step 3. Since  $k = n - 1 = 7$  and  $w_{found} = true$ , then end. The shortest path from  $s = 1$ , to all other nodes in the network has been found (see Figure A3.3).

Step 4. Return  $connected = w_{found} = true$ .

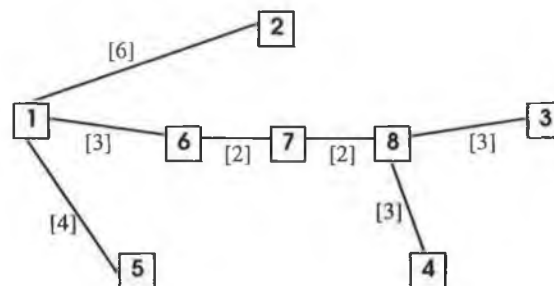


Figure A3.3 Graphical representation of solution



### A3.3 Implementation of Minimum-hop/Shortest-path Algorithm

---

#### EXAMPLE 3: IMPLEMENTATION OF MODIFIED SHORTEST-PATH ALGORITHM

Continuing from *Step 3* iteration ( $k = 5$ ) of EXAMPLE 1:

The status of the various inter-nodal sets and routing constraints up to this point are:  $w_{found} = true$ ,  $d_{min}^{path} = d_8^{path} = 7$ ,  $n_{min} = 8$ ,  $D^{path} = [0, 6, 13, 12, 4, 3, 5, 7]$  (same),  $P_1 = [1, 1, 2, 5, 1, 1, 6, 7]$  (same) and  $H^{count} = [0, 1, 2, 2, 1, 1, 2, 3]$ .

*Step 3.* Since  $k < n - 1$  and  $w_{found} = true$ , let  $k \leftarrow 6$ , set  $w_{node} = 8$ ,  $w_{hop} = h_8^{count} = 3$  and  $s_8^{checked} = true$ , and return to *Step 2*.

*Step 2.* ( $k = 6$ ):

$$d_3^{path} = 13 \text{ and } h_3 = 2 \quad \text{\textit{\{unchanged\}}}$$

$$d_4^{path} = 12 \text{ and } h_4 = 2 \quad \text{\textit{\{unchanged\}}}$$

It is noted that, using the standard shortest-path algorithm, optimal paths were routed via node 8. However, these improvements in the shortest-path distance are ignored by this algorithm, since the resultant paths would have twice the current hop-count in both cases.

Result:  $D^{path} = [0, 6, 2, 2, 4, 3, 5, 7]$ ,  $P_1 = [1, 1, 2, 5, 1, 1, 6, 7]$ ,  $w_{found} = true$ ,  $d_{min}^{path} = d_4^{path} = 12$  and  $n_{min} = 4$ .

*Step 3.* Since  $k < n - 1$  and  $w_{found} = true$ , let  $k \leftarrow 7$ , set  $w_{node} = 4$ ,  $w_{hop} = h_4^{count} = 2$  and  $s_4^{checked} = true$ , and return to *Step 2*.

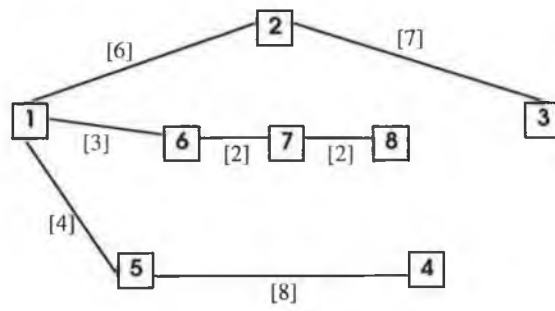
*Step 2.* ( $k = 7$ ):

$$d_3^{path} = 13 \text{ and } h_3 = 2 \quad \text{\textit{\{unchanged\}}}$$

Result:  $D^{path} = [0, 6, 2, 2, 4, 3, 5, 7]$ ,  $P_1 = [1, 1, 2, 5, 1, 1, 6, 7]$ ,  $w_{found} = true$ ,  $d_{min}^{path} = d_3^{path}$  and  $n_{min} = 3$ .

*Step 3.* Since  $k = n - 1 = 7$  and  $w_{found} = true$ , then end. The minimum-hop/shortest-path from node  $s = 1$  to all other nodes in the network has been found.

*Step 4.* Return  $connected = w_{found} = true$ .



**Figure A3.4** Graphical Representation of Solution

---