Survivable Network Design With Stepwise Incremental Cost Function

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

Author: Hai Wang, B.Sc.

Supervisor: Dr. T. Curran

Dublin City University

School of Electronic Engineering

Sept. 1999

Declaration

I hereby certify that the material, which I now submit for assessment on the programme of study leading to the award of Master 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 my work.

Signed: Hai Wang **ID No.:** 9697 0375

Date: 18/9/1999

Dedicated to my parent

ACKNOWLEDGMENTS

I would like to thank my academic supervisor Dr. Tommy Curran for his unceasing enthusiasm, interest, constructive suggestions, and for putting up with me over the years. I owe him much gratitude. I also thank him sincerely for facilitating my entrance to the field of telecommunication.

Thanks are very much owing to my colleague, Sean Murphy for his continual guidance and assistance during these years. Sean has been a constant source of encouragement to me and I wish to extend my sincere thanks to him for always making himself available to offer advice.

I am grateful to many other friends and colleagues at this time in no particular order: Dr. Feng Yu, Michael Collins, Liam Ward, Dr. Noel O'Connor, Eddie Cooke, Brendan Jennings and Yining Chen.

I would like to thank my special friend Dr. Shaestagir Chowdhury for his encouragement and loving support during these years.

I would like to convey my sincere thanks to my family, especially my mom and dad, without whose support and encouragement this thesis would not have been written. Thanks are due to my brothers, sister-in-law and niece, who continually inspired me from the family.

There are many, many individuals who have contributed in major and minor ways to this work. Thanks are due to them, whose names I forgot to mention.

Title: **Survivable Network Design with Stepwise Incremental Cost Function**

Author: **Hai Wang**

Abstract:

Modern society has become more and more dependent on information services, transferred in both public and private network, than ever before. The use of integration of computers with telecommunications has created a so-called "Information Age". The advent of high capacity digital telecommunication facilities has made it possible for the huge amount of traffic to be carried in an economical and efficient method, in recent years. These facilities, which are used to carry much higher capacities than the traditional ones, also result in the network's vulnerability to the failure of network facilities, i.e. a single link failure.

This thesis is concerned with the technology by which the spare capacity on the link of mesh networks is placed in order to protect the active traffic from network failure with a minimal cost. Although there have been many works to address the issue all of these works have been developed based on the assumption that the link cost with its capacity is linear. In fact, the linear cost functions does not reflect the reality that optic fiber cables with the specific amount of capacities are only available, in other words, the link cost function is stepwise rather than linear. Therefore, all existing algorithms developed for the linear assumption may not be applicable properly for the stepwise case.
A novel heuristic algorithm is proposed to solve the problem in this thesis. The algorithm is composed of two parts as follows. In part one, a maximum flow algorithm is employed to work out the maximal amount of feasible spare paths consisting of spare capacities in the network to re-route the disrupted traffic at the event of network failure. In part two, a newly proposed algorithm is used to find an alternative path on which to place the non-rerouted traffic on the failed link with the minimum network cost increment. The superiority of the algorithm is presented over other algorithms published in this area.

Table of Contents

Chapter 1 Introduction

1.1 Introduction

Modern society has become more and more dependent on information services, transferred in both public and private network, than ever before. The use of integration of computers with telecommunications has created a so-called "Information Age". The advent of high capacity digital telecommunication facilities has made it possible for the huge amount of traffic to be carried in an economical and efficient method, in recent years. These facilities, which are used to carry much higher capacities than the traditional ones, also result in the network's vulnerability to the failure of network facilities, i.e. a single link failure. A single multi-gigabit per second fiber optical cable can carry the capacity equivalent of tens of thousands of individual conversations and data connections. So the service disruption is no longer tolerated by industries if a fiber cable failed and there is no means in place to rapidly reroute the traffic which flowed on it. For example, a fiber cable cut in the AT&T network, which occurred at Newark in January 1991, interrupted 60 percent of voice and data coming and going out of New York, including three major commercial airports, for about 10 hours. The challenging issue for a network provider and designer is how to ensure the network continuity at an affordable cost and reasonable restoration time.

In the last decade, many approaches have been proposed to design the survivable network in the event of a network span failure. These restoration design technologies are basically divided into two categories: dedicated spare capacity routing and non-dedicated spare capacity routing. In dedicated spare capacity routing methods, such as automatic protection switching (APS) and self-healing rings (SHRs), a network will have spare capacity added to the network which is dedicated to rerouting the disrupted working traffic flows. The spare capacity is preset in the extra network facilities (for example idle fiber cable). When a network span fails the switching equipment automatically reroutes the working traffic by switching the working flow from the failed span to the preset spare alternate facilities. In non-dedicated spare capacity routing methods, spare capacity is placed on each network span, which the working capacities are located on. The spare capacity possibly contributes to the restoration of all possible network failures. It is worth noting that this type of network is donated mesh network because of the way they use spare capacity. In the event of a network failure, working

flows are rerouted over spans, on which spare capacities exist. In summery, in dedicated restoration technologies the working traffic is rerouted through predefined restoration paths in the extra network facilities, whereas in non-dedicated restoration technologies the network is recovered from a failure by using all the spare capacities available in the networks, as needed, to form restoration paths.

The primary advantage of dedicated capacity restoration methods is its speed. The spare capacity is effectively hardwired and used only in the event of network failure. The transport signals can be rapidly switched to the stand-by paths, which can result in restoration times of as little as 50 msec[1]. However, there are some disadvantages to this type of restoration, firstly, the total amount of spare capacity which is required to make such a design fully restorable, for all span failures, will be generally greater than the total amount of working capacity present in the network. The network will also be relatively inflexible in its configuration because all of its spare capacities are hardwired. The implication is that network traffic has to be accurately forecast, at the time of the network's construction, so that the network's fixed protection configuration will be able to support future traffic growth.

The network designed by the non-dedicated technologies, on the other hand, uses non-dedicated spare capacity and has the primary advantage of being fully restorable using an amount of spare capacity which can be 3 or 6 times less than that required in a dedicated network [1]. The reduction occurs because all the spare capacities in a mesh network are free to be re-used in the restoration of any span failure, whereas the spare capacity in a dedicated network can only be used in the restoration of a specific set of spans. An additional advantage which mesh networks offer is that both working and spare capacity is fully re-configurable to more easily accommodate future changes in offered traffic. The difference between working capacity and spare capacity, in a mesh network, is that one is committed to service and the other is simply sitting idle. If traffic offered to the network should suddenly increase in a certain area it may be carried by putting some of the spare capacity into service. However, this may result in the reduction of the network survivability. More usually, the benefit lies in placing new working capacity only where the growth is actually materialized this allowing a mesh network to be less dependent on the actual forecast ordering. Because of the advantages of the non-dedicated survivable network over the dedicated one, we will center on the non-dedicated survivable network in this thesis, namely, the survivable mesh network.

1.2 Research Objectives

The objective of this thesis is to find out how to place the spare capacity in mesh networks in order to protect the active traffic from network failure with a minimal cost. Due to the increasing interest in the survivable network design, there have been plenty of approaches proposed in recent years to address this problem [2] [3][4]. These approaches can basically be classified into two group, standard methods and heuristic methods. If standard methods, such as the Linear Programming (LP), or Integer Programming (IP), are applied to the problem of optimal spare capacity planning [3][4][5], then standard methods can be used to obtain the optimal solution to the Spare Capacity Placement (SCP) problem. In heuristic methods the greedy algorithm is employed to find the near optimal solution. Compared with the heuristic method, the standard method would get the optimal result at the expense of execution time. The standard method is more time consuming due to the larger amount of constraints required by LP (or IP) programming.

Almost all methods were developed based on the precondition that the link cost function with its capacity is linear. However, the fiber cables with the specific amount of capacities are only available in commercial markets, which makes it natural to consider solutions from the above methods may not be optimal although it is still giving a good approximation. In this thesis, we propose a heuristic approach to address the stepwise problem. The approach's effectiveness in terms of execution time and the costs required will be evaluated by comparing it with one standard method and one heuristic method respectively.

1.3 Thesis Structure

Chapter 2 is concerned with the introduction of the general background in the area of survivable network design, for example, some relevant terms and concepts, and then a brief description on the work detailed in current literatures is also given.

Chapter 3 contains an illustration of three algorithms that are used to obtain the feasible paths in the survivable network design, i.e. the Ford-Fulkerson's algorithm, the K-Shortest Paths algorithm (KSP) and the Matrix Maximum Flow algorithm (MMF). Then a comparison of their effectiveness in terms of the amount of paths found by these algorithms is provided.

Chapter 4 is concerned with the three algorithms which are applied to solve the SCP problem, IP formulation, max-latching algorithm and Stepwise Cost Heuristic algorithm (SCH). In addition, in order to solve the survivable network with stepwise cost function, the Addition Minimum Increment (AMI) algorithm is also proposed here. It is applied to obtain a path where the addition of the specified amount of traffic causes the minimum cost increment. IP formulation is a SCP solver that employs the IP programming to optimize the placement of spare capacity in 100% restorable network. The max-latching algorithm is a heuristic method that use the matrix knowledge to obtain a near optimal solution to SCP problem. The SCH algorithm is also a heuristic method that combines the maximal flow algorithm and the AMI algorithm to address SCP with the stepwise cost function.

Chapter 5 presents the results of the three SCP algorithms in terms of execution time and cost. Finally, an overall comparison of methods is presented.

Chapter 6 gives a discussion of the results. Finally, we suggestion some future research for mesh survivable network.

1.4 Algorithms proposed associated with the thesis

1. MMF algorithm, which is used to obtain the maximum flow between a given pair of nodes in a network.

2. AMI, which is used to find a path where adding the specified amount of traffic causes the minimum cost increment. The algorithm is developed based on Dijkstra's algorithm [8].

3. SCH algorithm, which is a heuristic algorithm to address mesh survivable network with stepwise cost function.

Chapter 2 Mesh Network Survivable Technologies

2.1 Introduction

As discussed above, mesh survivable networks are selected for study in this thesis because they have many advantages over dedicated survivable networks. The major advantages of mesh survivable network are as follows:

1. Flexibility:

Spare capacity is dedicated to almost all-possible failures of a network rather than any specific failure as dedicated survivable networks do. In case an unexpected failure occurs, restorable paths would be calculated dynamically according to the spare capacities available in order to protect as much interrupted traffic as possible from the network failure. Upon network failure, total spare capacities available are composed of both spare capacities placed before the failure, and released by traffic affected by the failure. All characteristics of mesh survivable networks contribute to its flexibility.

2. Extensibility:

Another attractive advantage of mesh survivable networks is its extensibility. Due to increasing reliance of our society on telecommunication, the amount of new services have been mounting rapidly day by day. Today's network with 100% survivability (introduced in Chapter 3) may no longer guarantee 100% survivability for the coming new services, tomorrow. Mesh survivable networks have more capabilities to meet the requirement of traffic growth than dedicated survivable networks. When new services are coming, spare capacities can temporally be used as working capacities to carry the new service although it reduces the network's survivability. To solve this, new spare capacities would be placed where new services occurred in order to keep 100% network survivability.

3. Affordable Time Complexity for real time services:

With the deployment of the powerful network equipment (i.e. DCS explained in the section 2.2.3 and ADM detailed in [6]), they have succeeded in speeding up the calculation time of SPC algorithm in each node so that the delay of services is affordable by users [7]. On the other hand, the fast restoration (50 msec) provided by

SHR and APS is not, strictly speaking, necessary in most instances. For example, in the telephone network, calling would not be dropped in 2 seconds, that means users can not perceive it. Therefore, 2 seconds is a very well reasoned and cost effective target for restoration times.

In summary, we will focus on studying mesh survivable networks in the following chapters. Before further discussion, we would like to illustrate a number of definitions regarding mesh survivable networks firstly.

2.2 Background to Mesh Survivable Networks

2.2.1 Centralized and Distributed Restoration

At the most level of abstraction, mesh restoration requires three conceptual steps: (a) accessing a network description, (b) computing a re-routing plan, and (c) deploying cross-connection actions to put the plan into effect. Centralized and distributed restoration can be differentiated by examining the steps of the restoration process.

In step one of the restoration process, centralized mesh restoration accesses a database at a central controller that stored information about all network nodes, connectivity maps, and spare facilities. In distributed restoration the network itself is the database; rather than accessing a central controller, each DCS obtains local network information from the links impinging on it.

To fulfil step two of the restoration process, centralized restoration computes the best re-routing paths for all failed signals based on the most recent network information available in the controller's database. Distributed restoration computes the re-routing plan is a distributed fashion across the entire network so that DCS only computes the part of the composite routing strategy which it is required to implement. The computed set of restoration paths form the re-routing plan in both cases.

In step three of the restoration process, centralized restoration requires the downloading of the re-routing plan to all DCS machines. However, distributed restoration leaves the computed set of restoration paths in place at each DCS node, obviating the need to download any re-routing plan.

While centralized and distributed restoration disperses information of the rerouting plan differently in step three of the restoration process, both centralized and distributed restoration may deploy the cross-connects required to implement a rerouting plan in the

same way. The ways in which cross-connection actions may be deployed at a DCS are explained in 2.2.3.

Centralized restoration is challenged with problems related to the size, cost, complexity, and vulnerability of the surveillance and control center needed for transport management. A centralized system is also dependent on the ability to maintain a complete, consistent, and accurate database image of the network which necessities redundant high-availability telemetry arrangements. As a result, centralized restoration is not only slower in real time than distributed restoration but runs the risk that a failure in the network will coincide with downtime at the central control site or a failure in the telemetry arrangement.

In a distributed approach there are no dependencies on telemetry or a central control site; the network is the computer on which the reconfiguration algorithm run. Therefore, the distributed approach is less vulnerable than the centralized approach. Furthermore, distributed mesh restoration algorithms have the potential to compute a rerouting plan much faster than centralized algorithms because they use the network as their database, and perform distributed processing over all DCSs. However, distributed restoration algorithms tend to be more complex than centralized algorithms because they must ensure that the routing decisions taken by all other nodes. The restoration algorithm presented in this thesis is a distributed mesh restoration technique.

2.2.2 SONET Network and its Signal Hierarchy

In this section, we provide background on the networks whose design will be optimized and into which the distributed dynamic path restoration algorithm will be deployed. The relevant network environment is that of the SONET transport network.

- Synchronous Digital Hierarchy (SDH)

Synchronous Digital Hierarchy (SDH) is a standard technology for synchronous data transmission on optical media. It is the international equivalent of SONET. Both technologies provide faster and less expensive network interconnection than traditional PDH (Plesiochronous Digital Hierarchy) equipment. In digital telephone transmission, "synchronous" means the bits from one call are carried within one transmission frame. "Plesiochronous" means "almost (but not) synchronous," or a call that must be extracted from more than one transmission frame. SDH uses the following Synchronous

Transport Modules (STM) and rates: STM-1 (155 megabits per second), STM-4 (622 Mbps), STM-16 (2.5 gigabits per second), and STM-64 (10 Gbps).

- Plesiochronous Digital Network (PDN) and Plesiochronous Digital Hierarchy (PDH)

A Plesiochronous Digital Network (PDN) uses point-to-point transmission systems and a layered multiplexing scheme to provide the physical connectivity, establishment, maintenance, and release of connections. In a PDN framing occurs at each multiplexing step, and each point-to-point transmission system is clocked independently. While all the clocks in a PDN are free running, they nominally operate at one of the standardised rate set in the Plesiochronous Digital Hierarchy (PDH) shown in Table 2.1

Digital Signal Level	Data rate (Mbps)
DS0	0.064
DS1	1.544
DS2	6.312
DS3	44.736
DS4	274.176

Table 2.1 Plesiochronous Digital Hierarchy

Within each rate in the multiplexing hierarchy, the various transmission systems in a PDN operate at slightly different frequencies. In order to multiplex signals with slightly different bit rates it is necessary to adjust the various input signals to a common rate by adding or deleting bits, may be eliminated or added without corrupting the transmitted information. While adding and deleting stuff bits in an input signal according to the rules stipulated in the PDN does not corrupt the information being transmitted, it does render the tributary signal inaccessible after multiplexing. In the PDH it is impossible to discern the difference between a stuff bit and an information bit in the payload of any digital carrier signal above the DS0 level with de-multiplexing the high speed into it constituent tributary signals.

- Synchronous Optical Network (SONET)

SONET is a standard in North America that defines both an optical interface, and rate and format specifications for optical signal transmission. It can support both broadband and narrow-band services. SONET Phase I specifies transmission rates, signal formats, optical interface parameters, and some payload mappings, however, it does not

standardize the operations and maintenance functions that must also be exchanged between Network Elements (NEs). Phase II of SONE defines the message set and protocols for using overhead channels for operations, administration, maintenance, and Provisioning (OAM&P). Phase II includes four major components: a protocol stack, a language, a message structure, and a common view of the data.

The SONET signal hierarchy plays a crucial role in SONET networks. Several factors determine the design of the SONET signal hierarchy. These factors include flexibility of supporting different services, simplicity in cross-connection, benefits from synchronous networks, facility maintenance, modularity for growth, and compatibility with existing networks. The basic building block (i.e., the first level) of the SONET signal hierarchy is called Synchronous Transport Signal-Level 1 (STS-1). The STS-1 has a bit rates of 51.84 Mbps. The traffic in SONET network is the combination of STSn (n = 1, 4, 9, 12...), currently the STSn is defined, as depicted in Table 2.2

Digital hierarchy	Line rate (Mbps)
STS-1	51.84
STS-3	155.52
STS-9	466.56
STS-12	622.08
STS-18	933.12
STS-24	1244.16
STS-36	1866.24
STS-48	2488.32

Table 2.2 SONET Signal Hierarchy with its Line Rate

It is worth noting that network traffic can only be represented by the integer number of STSn (for example, STS-1, STS-3 and so on) due to the modularity of the SONET signal hierarchy. The following network definitions also play a crucial role in design of mesh survivable networks.

In recent decades, increasing deployment of fiber facilities in telecommunications network raises concerns about service efficiency on the end-to-end basis due to the lack of signal standards for optical networks. This service efficiency concern, along with the need for supporting broadband service, which require bandwidth beyond the DC3 level (e.g. High Definition Television [HDTV]), led to the establishment of a national

standard signal format that supports present service and future broadband services. This optical signal format has been defined as SONET. So SONET network will become the major infrastructure network for the future broadband network (i.e. ATM network). Although SDH network is another major infrastructure for the future broadband network, the major principles derived from SONET network can also be applied to SDH network due to the similarity of SONET and SDH. Therefore, only terminology regarding SONET network is given in the following section.

2.2.3 Some Concepts regarding Network Structure

- A span is the collection of point-to-point STSn channels, working and spare, in parallel between two DCS nodes.
- A working channel is any channel that is part of a path bearing live traffic.
- A spare channel is an equipped-but-idle STSn channels terminated on DCSs;
- A working path is an end-to-end (source to sink) concatenation of channels (e.g. STS-1) from a pair of source and destination through the network.
- A spare path is a concatenation of spare channels through the surviving portion of a network that logically substitutes for one failed working path.
- A route is the sequence of spans followed by a working path or spare path; Note that each span is composed of both working and spare channels.

Adjacent nodes are those that are directly connected by a span. Feasible paths are defined as those that consist of spare paths between the nodes pair, where the failed span starts from and terminates in. Feasible paths can be used to reroute the disrupted traffic on the failed span in the event of network failure. In order to make these concepts clear we illustrate them in Figure 2.1.

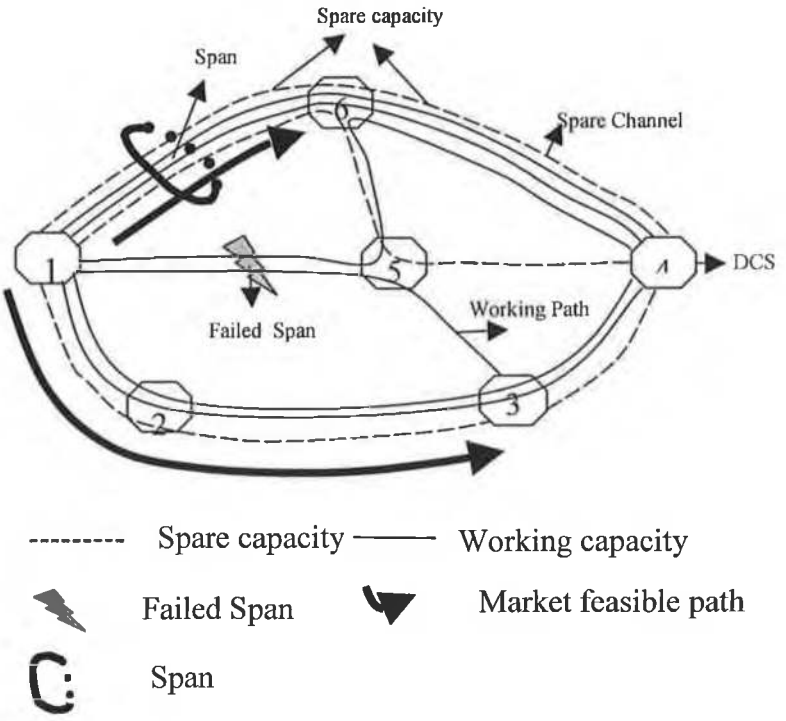


Figure 2.1 Mesh Survivable Network

In Figure 2.1, the span between DCS1 and DCS 4 fails, there are two working paths dropped by the failure, i.e. a working path (DCS1 → DCS5 → DCS6), the other (DCS1 → DCS5 →DCS3). Two feasible paths are found to restore the failed working path, i.e. the feasible path (DCS1 → DCS6) for the first failed working path, and the other (DCS1 → DCS2 → DCS3) for the second failed working path.

Digital Cross-Connect (DCS): Provides non-block connections between any of its ports. Offers cross-connection for SONET signal rates, through mapping and multiplexing to the various SONET STSn frames. Capable of monitoring allocated section/path overhead (management and status information accompanying the data), for enhanced flexibility to network management. Allows for the interconnection of various network topologies, i.e. ring and star, thus enhancing overall network flexibility. Its transparent switching characteristic offers extensive switching capability for network restoration and network re-configurability. In mesh survivable networks, DCS has three basic functions as follows:

- DCS can be used to switch the incoming traffic into its destination by identifying its header bit when networks stay in order.
- DCS can be used to cross-connect the disrupted traffic into spare paths by router table at the event of failure of network components.

- DCS can be used to work out the spare paths by using the knowledge about network topology and spare capacity layout. In distributed restoration scheme, the knowledge available to DCS is global. In centered restoration scheme, the knowledge comes from the neighbors of the DCS.

2.2.4 Some Important Parameters Regarding Survivable networks

There are a few major parameters that are used to judge the performance of mesh survivable networks, i.e. span survivability, network survivability and spare capacity redundancy.

Span survivability $Sv_{s,i}$ and network survivability Sv_n are defined as:

$$Sv_{s,i} = \frac{\min(w_i, k_i)}{w_i} \qquad Sv_n = \frac{\sum_{i=1}^{S} Sv_{s,i} w_i}{\sum_{i=1}^{S} w_i},$$

Where S is the number of spans in networks. Note that Sv_n is not the average of individual span survivability (unless all spans have the same values w_i). As defined, Sv_n weights the restorability of each span by the size of each span so that it expresses the total fraction of working capacity that is protected, not the average fraction protected on each span. This reflects the importance of large spans in overall network performance. The worse case survivability of a network (to span failure) is defined as the lowest span survivability level of any span in the network:

$$Sv_{n,wc} = \min_{i \in s}\{Sv_{s,i}\}$$

In the case of $Sv_n < 1.0$, $Sv_{n,wc}$ can be used as part of a determination of whether many spans are slightly below full restorability or, in contrast, one or a few spans are very under-protected.

Spare Capacity Redundancy is the ratio of spare capacities required to working capacities for the specified level of network restorability.

2.3 Representative of Network Topology in a Computer

In this section we will present two popular methods to represent a given network topology on a computer, Binary Adjacency Matrix (BAM) and its variation, namely, Capacity Matrix (CM). BAM is a simple and intuitive network representative where entries of this matrix are set 1 if the span exits between the corresponding nodes pairs, and 0 otherwise. The main disadvantage of BAM is that it may only be used to represent the topology of networks rather than the capacity on network spans, so an

additional matrix is required to store network capacity. Obviously, plenty of extra computer memory will be requested to store them. To solve this problem, we substitute BAM matrix by its Capacity Matrix (CM), thanks to the fact that capacity of a given network would be placed on a pair of nodes if and only if a span exist between them. We can easily convert the Capacity Matrix (CM) into the BAM if necessary. The BAM is therefore regarded as a special Capacity Matrix where the capacity in each span is 1. We will present the details of BAM and Capacity Matrix in the following section. Note that a Capacity Matrix (CM) is denoted by weight capacity in graph theory [13], where "weight" may represent capacity, distance and so on. Here, we will begin computer representative of networks with some graph theories.

2.3.1 Graphical Networks

A graph G = (V, E) consists of a set of vertices V = $\{v_1, v_2, \ldots\ldots, v_n\}$ with the finite number of elements and a finite set E of edges E = $\{e_1, e_2, \ldots\ldots, e_m\}$, as seen in Figure 2.2.To each edge, *e,* there corresponds a pair of distinct vertices (*u, v*) where *e* is said to be *incident* on. When drawing a graph we represent each vertex by a dot and each edge e by a line segment joining its two end vertices. A graph is said to be a directed graph (or digraph for short), as seen in Figure 2.2, if the vertex pair (*u, v*) associated with each edge *e* is an ordered pair. Edge *e* is then said to be directed from vertex *u* to vertex *v*, and the direction is shown by an arrowhead on the edge. A graph is undirected if the end vertices of all edges are unordered (i.e. edges have no direction). A network is a directed or undirected graph in which a real number is assigned to each edge. This number is often referred to as the weight of that edge. In a practical network this number (weight) may represent the driving distance, the construction cost, the transit time, the reliability, the transition probability, the carrying capacity, or any other such attribute of the edge.

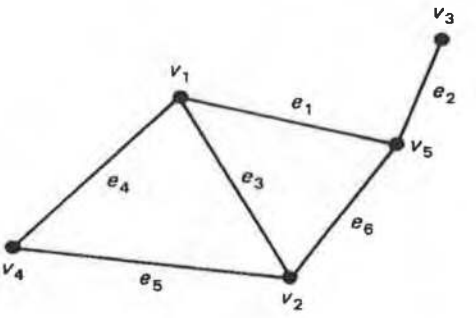


Figure 2.2 Undirected graph with 5 vertices and 6 edges.

Two edges are said to be parallel if they have the same pair of end vertices (and additionally, if they have the same direction in case of a directed graph). Throughout following chapters we assume that the network under consideration has no parallel edges. (This assumption gives us some simplicity without any cost in generality). Thus we can refer to each edge by its end vertices.

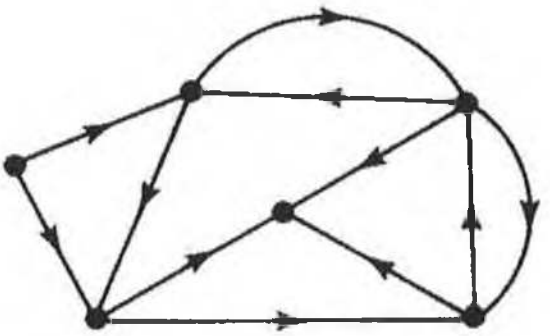
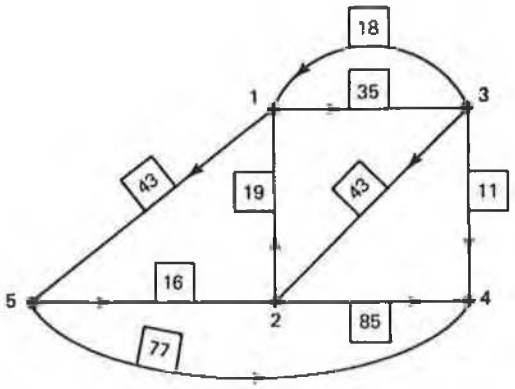


Figure **2.3** Digraph with vertices and 11 edges

We denote the letters n and m as the number of vertices and number of edges respectively in a network. A vertex will be referred to as a node (a term more popular in applied fields).

2.3.2 Weight (Capacity) Matrix and Binary Matrix of Networks

The simplest and perhaps the most popular computer representative of a network is the weight matrix (capacity matrix). The weight matrix of an n-node network is an $n \times n$ matrix $W = \{w_{i,j}\}$ in which the (i, j)th entry $w_{i,j}$ is the weight of (i, j). The edge from node i to node j in the network G. If there is no edge (i, j) in G, the corresponding element is set usually to be 0 (in practice, some very large number). The diagonal entries are usually set to zero (or to some other value depending on the application and algorithm). It is easy to see that the weight matrix of an undirected network is always symmetric. A network and its weight matrix are shown in **Figure** 2.4. Boxed numbers next to the edges are their weights.



$$\begin{pmatrix} 0 & 0 & 35 & 0 & 43 \\ 19 & 0 & 0 & 85 & 0 \\ 18 & 43 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 16 & 0 & 77 & 0 \end{pmatrix}$$

Figure **2.4** A network and its weight matrix

Based on the network weight matrix we obtain a Binary Adjacency Matrix (BAM) by converting non-zero entry to 1 because non-zero entry in the network weight matrix indicates that there is a link between the corresponding vertices. In **Figure** 2.5, the binary adjacency matrix of the network in **Figure** 2.4 is given.

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

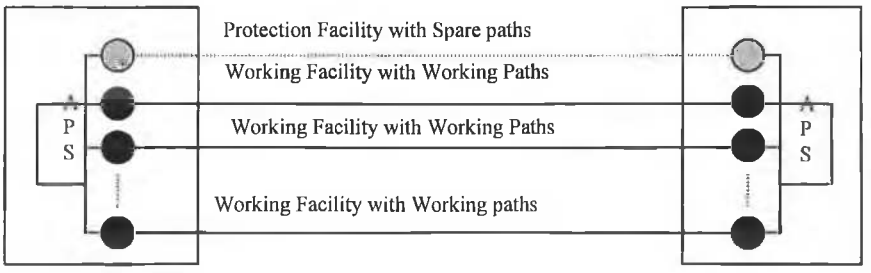
Figure **2.5** Binary Adjacency Matrix of a Network

All algorithms that will be illustrated in Chapter 3 and 4 works with weight matrix and binary adjacency matrix.

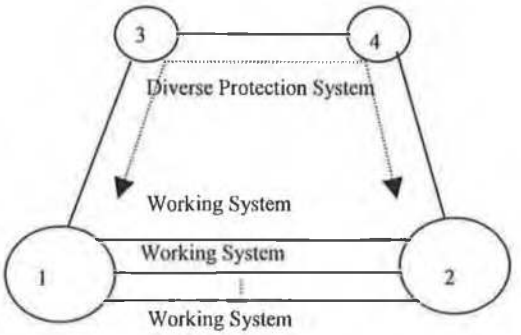
2.4 General Description of Mesh Survivable Networks

In general, the restoration technologies can fall into two categories, traffic restoration and facility restoration. Traffic restoration is the circuit level restoration where the disrupted circuit is re-routed around failure. A circuit switch, such as AT&T's 5ESS switch, perform traffic restoration by rerouting calls around failed circuit. Facility restoration is the transportation level restoration that reroutes the failed traffic in large units such as STSn upon failure of the network. As discussed in Chapter 1, facility restoration can be divided in two subcategories, dedicated spare capacity (facilities) restoration and non-dedicated spare capacity (facilities) restoration. Facility dedicated restoration is characterized by using the dedicated facilities for protection including Automatic Protection Switch (APS), dual homing, and Self-Healing Rings (SHRs). Non-dedicated facility restoration is defined by using DCSs (or ADM) to reroute a failure point.

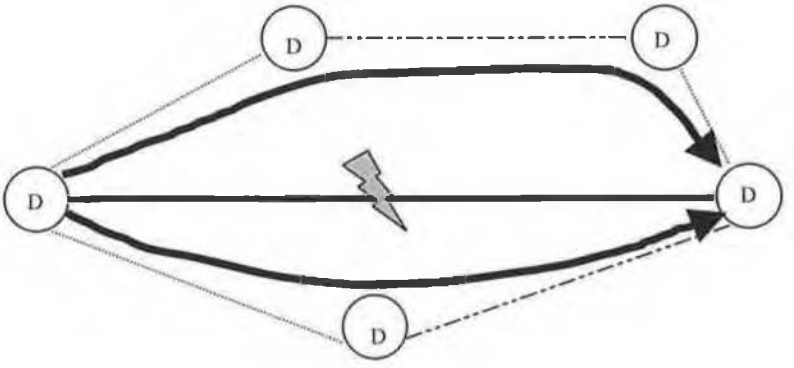
These techniques are also referred to as DCS restoration techniques. DCS restoration does not required the dedicated facilities to working systems for restoration. Alternatively, it uses spare capacities with working systems to restore disrupted traffic. Figure 2.6 shows several examples of facility-dedicated restoration and non-dedicated restoration in order to make the above definitions clearer.



(a) 1:N APS Architecture



(b) 1:N APS Architecture with/without Diverse



Work capacity

Spare Capacity

Link disrupted

DCS Digital Cross-connected Switch

Reroute Path

(c) DCS restoration

Figure 2.6 Facility Restoration: (a) and (b) is referred to as Facility Dedicated Restoration and (c) is referred to as Facility non-dedicated Restoration

- 1: N APS Diverse Protection

The APS approach is commonly used to facilitate maintenance and protect working services, and has the advantage of being totally automatic. The 1: N diverse protection structure is an alternative to the commonly used 1:N protection strategy, where N working fiber system share one common protection fiber system. The only difference between these structures is the location of the fiber protection system; the 1:N protection structure places the protection fiber in the same route as that of working systems, and the 1:N diverse protection structure places the protection fiber in a diverse

route. In a 1:N system, a cable cut occurs and a 1:N diverse protection scheme is used, part of the service is lost because only one of the N working systems can be restored. Figure 2.7 shows the difference between these structures. This diverse protection scheme is attractive because electronics cost dominate total cost and remain unchanged when attempting to achieve survivability. a 1:1 diverse protection arrangement, which provides 100% survivability for fiber cable cuts, required more facilities and equipment than 1:N diverse protection arrangement

In contrast with facility restoration, traffic restoration is more efficient with respect to utilizing network bandwidth at the expense of more complex switch control. Due to the popularity of SONET network in the future broadband, our attention will be put on facility restoration. Although facility dedicated restoration has several advantages over facility non-dedicated restoration, e.g., simpler control and quicker restoration, in the thesis facility non-dedicated restoration still has been exploited and studied due to its flexibility of coping with unexpected failure and traffic growth and higher effectiveness of utilizing network bandwidth. In fact, real time factor is no longer the most important for most of services provided by the network, e.g., data and even voice [7], since less than 2 seconds outage time is not perceived by users in practice. In this chapter, facility non-dedicated restoration will be studied as a major technology to produce a survivable network.

Non-dedicated facility restoration can be considered at two different levels: path restoration and span restoration. Path restoration restores the end-to-end logical path affected by the span cut, and span restoration restores all disrupted channels carried by a failed span. Figure 2.7 depicts examples of span and path restoration techniques. Table 2.3 shows a relative comparison between the span and path restoration method. The path restoration uses spare capacity more efficiently than span restoration [24]. However, span restoration requires a simpler routing decision. Thus, span restoration is expected to be faster than path restoration in terms of restoration time.

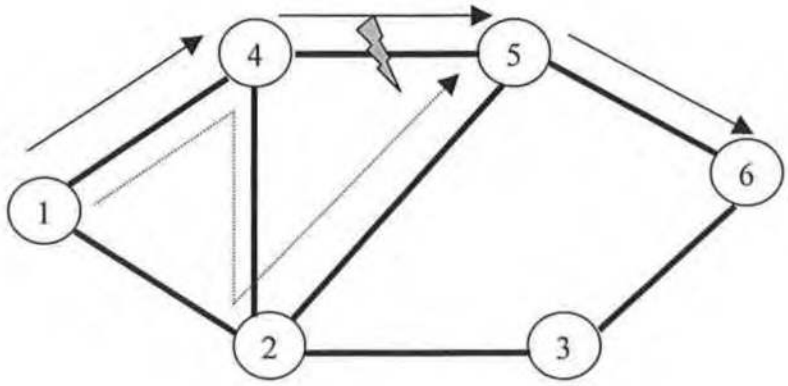


Figure 2.6 (a) Span Restoration

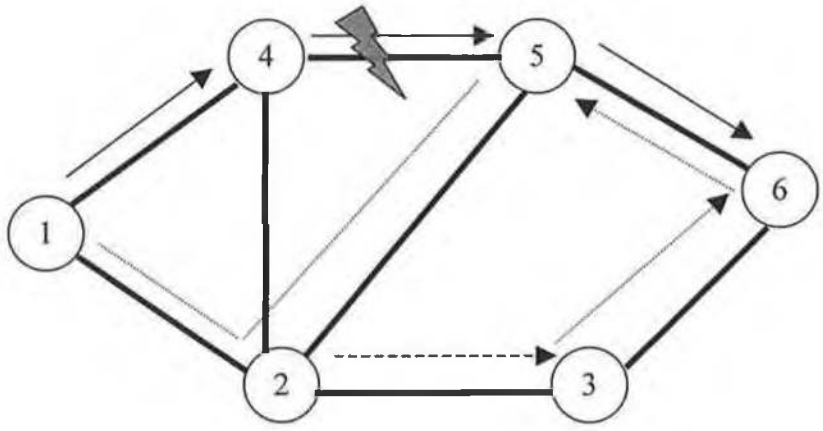


Figure 2.6 (b) Path Restoration

Original

Broken

Restored

Figure 2.7 Span Restoration and Path Restoration

Attribute	**Span Restoration**	**Path Restoration**
Time Complexity	lower	higher
Spare Capacity redundancy	higher	lower
Restoration Control	simpler	complex

Table 2.3 Relative comparison between spans restoration and path restoration

Technically, span restoration does not have any big difference from path restoration. In span restoration, the traffic affected by a failure of a single cable is the working channels between the nodes pair connected by the failed span, so the number of nodes pairs affected is only one for any particular single cable failure. In path restoration, the traffic affected by a failure of a single cable is the working paths through the failed span. The number of nodes pairs affected is the number of the demand pair affected by the failed span.

Due to the similarity of the two restoration technologies, all restoration algorithms for path restoration can be applied to span restoration with a small modification. Almost all algorithms developed from span restoration can also be applied to path restoration. A heuristic SCP algorithm [18] proposed by B.D.Venrable *et al* in 1993 is an exception. The algorithm just works for span restoration, we will discuss the algorithm in detail in Chapter 4. So span restoration can be considered as a special case of the path restoration. In the thesis, although the algorithm we proposed can works for both span restoration and path restoration, only its span restoration implementation will be given because the heuristic algorithm, a span restoration algorithm, will be selected to compare with our new algorithm.

As discussed above, the facility non-dedicated restoration is a transport level restoration technology, which take advantages of functionality of Digital Cross-Connect Switch (DCS) to re-route the failed traffics through spare capacities on other unaffected spans. It is not hard to realize 100% survivability if desired, but a challenging issue is to how to obtain a Spare Capacity Placement (SCP) algorithm, which meets not only network requirement for real time and minimal cost, but also 100% survivability or any target level of specified survivability less than 100%.

2.5 General Description of Spare Capacity Placement problem (SCP)

The aim of designing mesh survivable networks is to install the extra backup spare capacities to be utilized to reroute the disrupted traffics in the event of network facilities failure minimizing the cost. All these kinds of problems have been defined as the SPC problem, namely, Spare Capacity Placement (SCP) problem. Here, a general description on the problem of mesh survivable network is given.

Spare capacity placement problem can be stated as:

$$\text{Min} \sum_{i=1}^{S} f(s_i + w_i) \qquad \text{such that } R_n = 1,\ w_i \geq 0,\ s_i \geq 0 \quad i = 1,\ldots, S$$

Where s_i and w_i are the number of the spare channels and working channels on span i respectively, $f(s_i + w_i)$ is the link cost function with its capacities carried on span i. SCP problem considered requires following network information as input: the network topology, a vector of working paths, w_i, the target level of network survivability R(assumed to be 100 here), and the maximal restoration path length allowed (i.e. Hop-Limits). The output is a vector of spare paths, s_i, which meets the survivability target within hop-limits set by the maximum allowed restoration path length.

2.6 Review of SCP Approaches in the Literature

Due to the increasing concern of survivable network design, there has been lots of work to address this issue in the last decades. In general, all approaches proposed can fall into two groups, heuristic approach and non-heuristic approach. Heuristic approach is characterized as the near-optimal algorithm that is used to reduce the higher time complexity of non-heuristic approach at the expense of obtaining the near-optimal solution to SCP problem. Non-heuristic approach is the optimal algorithm that is employed to obtain the exact solution of SCP problem, but, taking more execution time, so it can only be applied to small-scale networks. Note that almost all non-heuristic approaches apply Linear Programming (LP) or Integer Programming (LP) to formulate SCP problems.

In a 1993 paper [14], Meir Herzberg suggested a decomposition approach for SCP problem in order to deal with a single cable failure. The approach is a span restoration algorithm that was developed based on a Linear Programming (LP) model. The paper assumes study networks can be decomposed into several simpler sub-networks model, and then formulate these sub-network SCP problems into Linear Programming model. The approach is committed to minimizing spare channels that are used to reroute disrupted working channels at the event of failure. For the same purpose, Meir Herberg, et al [15] proposed a new model to address SCP algorithm, and explored how hop-limits factor affects the optimal result, and eventually the principle to choose proper hop-limits was given. This algorithm is composed of two parts: Part1 – relies on a Linear Programming (LP) formulation (Min-Max) from which a lower bound solution is found for SCP problem; Part2 – rounds up the solution of Part 1 and uses a series of related LP, aimed to tightening the round-up assignment to a practical optimal solution. The

main disadvantage of the above two algorithms is that both just can cope with a single cable failure, so his 1995 paper [16], a new algorithm was proposed to address multi-cable failure and node failure. It is also based on a Linear Programming (LP) model.

Based on an Integer Programming mode, R.R.Iraschko *et al,*[21] developed an algorithm that can be used for span restoration and path restoration. The algorithm is aimed to minimize the total amount of spare capacities that contributes significantly to its costs. To reduce spare capacities further, the algorithm is also designed to re-use the working capacity released by affected working paths. So the algorithm is more effective in terms of minimizing spare capacities that is necessary to protect networks from failure. A detailed description about the algorithm is given in chapter 4 and its C++ codes implementation is done in the PC basis as a benchmark.

Due to the higher time consumption of LP or IP based SCP algorithm, recently several researchers have addressed the issue by introducing heuristic approaches. In 1991, W.D.Grover et al [19] developed an effective approach for this task which heuristically first finds a feasible solution ("Forward Synthesis") and then reduces redundancy while maintaining the restoration level achieved ("Design Tightening"). The approach provided a full tradeoff curve between survivability and redundancy for a network that IP (LP) based algorithm can not give. In practice, the curve is really important for network planners and designers because it can provide a guideline for network design. In 1997, W.D.Grover et al [18] proposed a more effective heuristic algorithm for SCP problem. The heuristic algorithm runs much faster times than the optimal Integer Programming (IP) while having big capacity redundancy. These characteristics can contribute significantly to the problem of finding the globally best single or multiple new span additions in the evolution of large transport networks. The algorithm will be also described and implemented on a PC that is benchmarked in Chapter 4.

Both two types of SCP algorithms assumed that the link cost with its capacity is linear, however, it is not always true, since only are fiber links with stepwise capacity available in commercial market. That means that link cost is no longer continuous but stepwise. So it is obvious that the SCP algorithms proposed with the assumption of linear link cost restrict the implementation issue although these algorithms are still a good approximation of the SCP solution. In Chapter 4, we proposed a new SCP algorithm to address the issue based on Matrix Maximum Flow (MMF) and Addition Minimum Increment (AMI) that will be described in Chapter 3 and in Chapter 4 respectively.

Chapter 3 Algorithms for Finding Feasible Paths In a Network

3.1 Introduction

This chapter focuses primarily on the investigation of several important approaches, called Feasible Path Algorithms (FPAs), which are designed to find feasible paths between a pair of nodes in networks. The FPAs are major functions in the Mesh Survivable Network (MSN) design. In fact, the MSN design algorithms employ the different FPAs to determine restorable paths to re-route the failed traffic upon network failure, so FPAs algorithms play a crucial role in improving the effectiveness of MSN design in terms of cost savings and execution time. In general, the FPAs algorithms employed by MSN design take almost 99% of execution time taken by MSN design [10], hence, the effectiveness of FPAs contributes significantly to the reduction of MSN's time complexity.

We will investigate two existing FPAs algorithms, i.e. the Ford-Fulkerson's algorithm, the K-Shortest Paths (KSP) algorithm. The Ford-Fulkerson's algorithm is a maximal flow algorithm that can be used to find the maximal flow between a given pair of nodes in a network. Based on the matrix theorem 40 in [11], we developed a new FPAs algorithm, which we have termed Matrix Maximum Flow (MMF) algorithm. Finally, the relative comparison of these FPAs algorithms is given in terms of total feasible paths.

We organize the rest of this chapter as follows. In Section 3.2, The description of two types of MSN design problems, i.e. pre-planned survivable network design and dynamic restorable network design, and their different requirement for FPAs. In section 3.3, we give two criteria for judging performance of MSN design, i.e. network survivability and restorability. In Section 3.4, we discuss three FPAs algorithms, i.e. the Maximum Flow algorithm (i.e. the Ford-Fulkerson's algorithm), the K-Shortest Path algorithm and MMF. In section 3.5, the test networks over which the FPAs algorithm will be applied are given. In section 3.6, Results are presented and discussed. Conclusions are presented in section 3.7.

3.2 Pre-planned Survivable Network Design and Dynamic Restorable Network Design

There is an increasing reliance by society on the timely and reliable transfer of large quantities of information (such as voice, data, and video) across high speed telecommunication networks. A network failure, such as the loss of a link or a node, can occur due to a variety of reasons causing service disruptions ranging in length from seconds to weeks. Typical network failures are attributed to accidental cable cuts, hardware malfunctions, software errors, natural disasters (i.e. fires), and human error (i.e. incorrect maintenance). With advent of high bandwidth optic fiber more and more services in networks are being carried on a few optical fiber cables bundle, which means even a single fiber outage can affect many services. In the last decades there has been an increasing interest in finding approaches to design networks that are resilient to failure [2][15][28].

In general, the survivable network design can fall into two steps, i.e. pre-planned survivable network design and dynamic restorable network design, preplanned networks have preset routes for restoration, dynamic restorable networks find routes dynamically based on all existing spare capacity in a network, hence, the dynamic restorable network design relies significantly on the pre-planned survivable network design. Both the preplanned survivable network design and the dynamic restorable network design work together to obtain a survivable network.

The aim of the pre-planned survivable network design is to work out an approach, called Spare Capacity Placement (SCP) algorithms, to place spare capacity in networks in order to enable the network to recover from network failure. It is impossible and unnecessary to predict all possible network failures that may happen. If all predictable failures were considered large amounts of extra capacity would be required resulting in highly over-engineered network. In fact, some network failures are so unlikely to happen, for example, multiple links failure, and some other network change is not predictable, for example, traffic growth, so only are the most likely network failures, e.g. a single link or node failure, can be taken into account in pre-planned survivable network design. The SCP algorithm is applied to place spare capacity for all considered network failures with minimal network cost. The re-routing tables is generated based on the result of the SCP algorithm, and stored in the network DCSs. When a pre-specified

failure occurs, the network switches to its outage state, where DCSs in each network node have the following functions to perform to restore the failure:

1. Identify the network failure, i.e. which span or node fails.
2. Change the original route table to that of Outage State corresponding to the occurred failure and then switch the affected working paths into the pre-set spare paths that were placed by SCP algorithm at design time.
3. Switch back to the normal state when the failure is fixed.

So the pre-planned survivable design has the capability to restore all pre-specified network failures but can not recover from failure in the case of unexpected network failures and traffics growth. The dynamic restorable network design can be employed to address the issue.

The aim of the dynamic restorable network design is to work out a network re-routing protocol, called the Restoration Scheme (RS), to extend the network's flexibility of coping with network traffic growth and unexpected network failures based on the existing network spare capacity placed by the preplanned survivable network design. The RS is a protocol that uses the existing spare capacity available to re-route the newly added traffic or failed traffic at the event of unexpected network failures. For example, when an unexpected network failure occurs, DCS will invoke the Restoration Scheme (RS) embedded in them, DCSs would perform following functions.

1. Obtain the network knowledge, i.e. network topology, spare capacity layout and so forth, as the input to Restoration Scheme
2. Execute the RS to work out the feasible paths for failed or added working paths.
3. Switch the failed or newly added traffics to the feasible paths found by the RS.

Both the pre-planned survivable network design and dynamic restorable network design take use of the FPAs to find the "suitable" paths. In the pre-planned survivable network design, the term "suitable" paths are those on which the spare capacity is placed to re-route the failed traffic upon all most-often-happened network failures with the minimal network cost. In the dynamic restorable network design, the "suitable" paths are those on which the newly added traffics or failed traffics at the unexpected network failure can be re-routed as many as possible while considering the real time requirement, obviously, two types of MSN design require the FPAs differently. The former is more

sensitive to network cost, the latter is more sensitive to real time. The difference determines the difference on the implementation of FPAs.

3.3 Network Survivability and Restorability

Network survivability and restorability are two important concepts for pre-planned survivable network design and dynamic restorable network respectively. The network survivability is a network design objective and the network restorability is a criteria to measure the flexibility of a network to deal with unexpected network failure or traffic growth.

3.3.1 Network Survivability

The definition of network survivability has been given in Chapter 2. Network survivability is a network design objective to be reached by network designers. The survivability has to be set as an input of a SCP algorithm that is applied to find out the placement of spare capacity to prevent the network from failure. The higher survivability networks are, more spare capacities are required. In practice, we can set up any level of network survivability according to the requirements of the network operator.

3.3.2 Network Restorability

Network restorability is a metric that can be used to measure the network' ability to use the existing spare capacity in the case of network failure. The layout and number of spare capacity found by SCP algorithm in pre-planned survivable network design and Restoration Scheme (RS) determine the level of network restorability. It reflects the capability of survivable networks to utilize the existing spare capacity to recover from network outages by using the restoration protocol (scheme) in the face of unexpected network events. The aim of the restorable network design is to find protocol (scheme) to serve as many traffics as possible in order to minimize impact on the network performance when an unexpected failure of network facilities occurs or new traffics need to be added.

Network restorability has the same mathematical definition as that of network survivability. Before we give the network restorability the network span restorability is introduced as follows:

$$R_{f,i} = \frac{\min(w_i, k_i)}{w_i},$$

and then the network restorability is defined based on the network span restorability.

$$R_n = \frac{\sum_{i=1}^{s} R_{f,i} w_i}{\sum_{i=1}^{s} w_i}$$

Where F represents all possible network failures and S network spans, including single span failure, Here, $R_{f,i}$ is the single span restorability in the event of the failure $f (\in F)$. R_n is the network restorability for all unexpected network failures. Because the unexpected network failure and traffic growth are not known beforehand the exact result of network restorability is not available. We can approximate it by the following methods.

In case the new traffic is assumed to take place, we can convert spare capacity in each network span to working capacity (one at a time), the Restoration Scheme (RS) is applied to re-route these new work capacity, and calculate its restorability. In case two-span failure is assumed to be an unexpected network failure, we choose all sets of two spans as failed (one set at a time) and calculate network restorability.

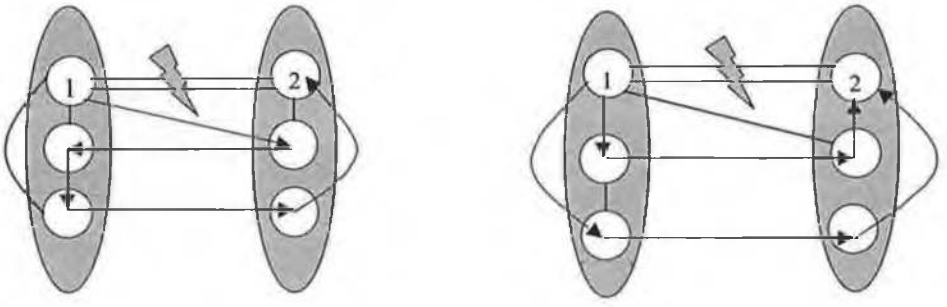
Since network survivability only applies in pre-planned survivable network design, we refer to pre-planned survivable network design techniques as survivability techniques for simplicity. Similarly, dynamic restorable network design techniques are referred to as restorability techniques.

3.3.3 Relationship between Survivability and Restorability

Although there is some similarity between network survivability and restorability they reflect the different performance of networks upon their failure. The former is a design time objective, the latter is a network performance measure.

100% of network survivability in design time can not guarantee 100% of network restorability. The reason is that the restoration schemes are executed in a distributed way, and do not always find desirable paths (optimum paths) which are expected by the

Spare Capacity Placement algorithm (SCP) in design time. This is illustrated in Figure3.1.



(a) Incorrect Route Assignment **(b)**Correct Route Assignment

Figure3.1 Network Route for a Single Span Failure

Where the circles are the network nodes and the lines represent the network span channels, for example, two span channels in the span between nodes 1 and 2. The arrows indicate the routing direction.

Assume the span between node 1 and node 2 fails, two units of capacity will be dropped. In design time, two feasible paths are assigned by a SCP algorithm for a span failure as depicted in Figure3.1(b). However, when the network outage occurs, the restoration scheme may chooses wrong feasible paths so that the wrong chosen paths as depicted in Figure3.1(a) exhaust all spare capacity assigned for the specified failure.

3.3.4 Different Requirement of Network Survivability and Restorability for Feasible Path Algorithms (FPAs)

Due to the difference between restorability and survivability techniques, the implementations of FPAs regarding network survivability and restorability are not always same.

The primary aim of survivability and restorability (S&R) techniques is to recover networks from the failure of network facilities taking account of their time complexity, and minimizing the cost of network construction. Demands for the design of survivable and robust networks have been increasing, so there has been plenty of work done in developing SR techniques [2][3][27][29]. As discussed above, finding feasible paths, through which the failed working channels are re-routed, is an important issue in SCP algorithm, so FPAs play a crucial role in improving the effectiveness of SR techniques in terms of cost savings and execution time. Because different FPAs techniques have

different advantages and disadvantages in practice, some may only be suitable for survivability techniques or for restorability techniques, while others can be applied to both cases. In some cases, some FPA approaches for survivability techniques that are not suitable for restorability techniques can be modified to work for restorability techniques. For example, the FPAs is employed in restorability techniques are more sensitive to execution time than ones for survivability because they must operate in a real network environment, and find solutions very quickly to minimize the disruption to the network caused by the network failure: such speed requirements may result in the use of non-optimal solutions. On the contrary, the FPAs algorithms employed by survivability techniques can take relatively longer time to get as many feasible paths as possible to place spare capacity on in order to minimize the cost of networks. In the following section several FPA algorithms, i.e. Ford-Fulkerson's algorithm, the K-shortest paths algorithm, and the Matrix Maximum Flow algorithms (MMF) will be presented and the advantages and disadvantages of these algorithms will also be compared in five test networks. Since Restorability technique is beyond the scope of the project, we will focus on the survivability techniques in the rest of this thesis. But, in order to distinguish the different FPA implementation for two types of network design, Two implementations of the K-Shortest Paths algorithm will also be discussed for both survivability and restorability techniques respectively.

3.4 Feasible Path Algorithms (FPAs)

In this section the FPAs are discussed: the Ford Fulkerson's algorithm, the K-Shortest Paths (KSP) algorithm and the Matrix Maximum Flow algorithm. We will discuss and compare these algorithms in terms of the number of paths found. In addition, we describe two types of implementation of K-Shortest Paths (KSP) algorithm, namely, non-disjoint KSP and disjoint KSP in order to show survivability and restorability techniques impose different restrictions on FPAs?

3.4.1 Ford-Fulkerson's Algorithm

3.4.1.1 General Theory of Maximal Flow

As the name suggests, the maximal flow algorithm is used to find the maximal flow between a given pair of nodes in a directed network N = G(V, E, C) where V is the set of nodes in the network, E is the set of edges in the network, C is a the network capacity matrix. In addition, the number of working channels (i.e. traffic flow) is the existing

traffic flow in the network, it is stored in matrix f. $C_{i,j}$ and $f_{i,j}$, the entries (i, j) of matrix C and matrix f, are the maximum traffic flow allowed through edge (i, j) and the existing traffic flow that exists in the edge (i, j) respectively. Note that $C_{i,j}$ is always greater than or equal to $f_{i,j}$ resulting from the stepwise link cost function.

$$0 \leq f_{i,j} \leq C_{i,j}, \qquad \text{for the edge } (i, j) \in E$$

Here, the difference of $C_{i,j}$ minus $f_{i,j}$ is the number of the installed-and-idle spare channels. In the following section we will describe how to find the maximum number of spare paths consisting of these existing spare channels on each of network edges between a given pair of network nodes which is represented by its source node s and destination node d.

For each node j, other than source node s and destination node d, The conservation law, i.e. the sum of flows into a node equals the sum of flows leaving the node, are satisfied,

$$\sum_{i}^{s(i)} f_{i,j} - \sum_{l}^{s(l)} f_{j,l} = 0 \qquad i \in s(i), j \in s(l),$$

Where $s(i)$ is a set of links for which node i acts as a source , $s(j)$ a set of links for which node l acts as a sink. Since the network traffic flow may be used by local users in node s and node d, the flow conservation law may not be held.

The sum of edge flows into the destination node d is $f(d)$,

$$f(d) = \sum_{i} f_{i,d}$$

Where i is the set of edges incoming to the destination node d.

The maximum-flow problem (max-flow) is to find the values of $f_{i,j}$ for all the edges such that $f(d)$ is maximised. To find the maximum flow a path augmentation technique can be used. An augmenting path is a sequence of pair-wise adjacent edges from the source node s to the destination node d, which allows us to increase the value of flow. If the (i, j)th edge orientation coincides with the direction of the path, then in order to push more flow through it, f_{ij} must be less than $C_{i,j}$. If the (i, j)th edge points in the opposite direction, then in order to push some additional flow through it, we must reduce its flow, and $f_{ij} > 0$ is required.

- **Labelling Algorithm**

To find an augmenting path from s to d, a labelling procedure is used. The labelling algorithm assumes that there exists an initial flow in the network. we may have, for instance, all $f_{ij} = 0$. Then labels of the form (j, ε) or $(j, -\varepsilon)$, where ε is a positive number or infinity, are assigned to each other, beginning with the source *s*. If it is possible to label the sink *d*, a change of the flow from *s* to *d* is made and the labelling is repeated. If it is impossible to label the sink, the flow is optimal. The procedure uses two routines, A and B. During each step of routine A, a node is in one of three states: unlabeled and unscanned, labelled and unscanned, or labelled and scanned. Initially all nodes are unlabelled and unscanned.

1. Routine A (Labeling)

Step 1. Label the source with $(-, \infty)$. Because the source is the node for labelling and is assumed to allow any number of flows to come in its predecessor node is none represented by -, and the flow through it is infinite ∞.

Step 2. For any labelled and unscanned node *j* with label $(i, \varepsilon_j$ (or $-\varepsilon_j))$, scan it by examining all unlabeled nodes *l*, adjacent to *j*.

a. If (j, l) is an edge and $f_{jl} < C_{jl}$, then label node *l* with (j, ε_j), where $\varepsilon_l = \min(\varepsilon_j, C_{jl} - f_{jl})$.

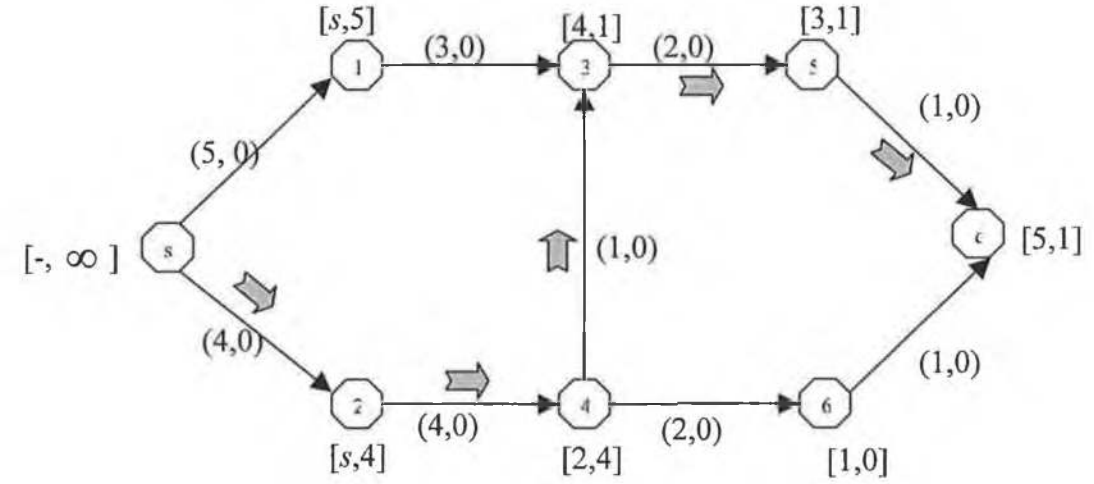
b. If (l, j) is an edge and $f_{lj} > 0$, then label node *l* with $(j, -\varepsilon_l)$, where $\varepsilon_l = \min(\varepsilon_j, f_{lj})$.

Step 3. The node *l* are now labelled and unscanned, and node *j* is labelled and scanned. Repeat step 2 until either the sink *t* is labelled or it is impossible to label the sink. In the first case we have a breakthrough and route B is initialised. In the second case we have a non-breakthrough and the algorithm is terminated, the flow is optimal.

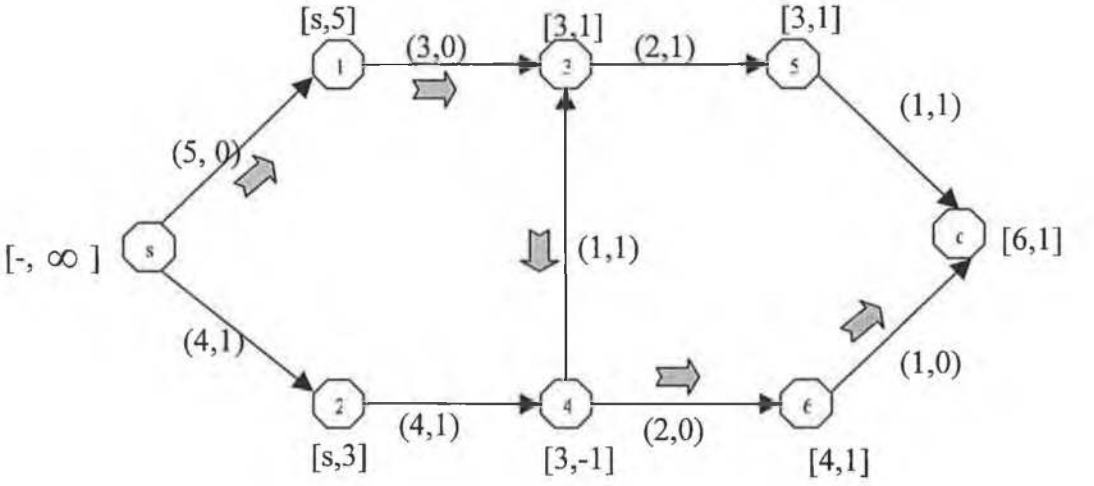
2. Routine B (Flow Change)

The sink d has been labelled with (l, ε_d). Therefore, the network with the current flow admits an augmenting path from *s* to *d*, which can increase the flow value by ε_d and *l* is the second last node on this path. Hence set $f_{ld} = f_{ld} + \varepsilon_d$. Now look at the node *l* labelled $(i, \varepsilon_j$ (or $-\varepsilon_j))$. If the second label is ε_l then *l* has been labelled from *j* by using the edge (j, l), therefore set $f_{ld} = f_{ld} + \varepsilon_d$. Otherwise, the edge (l, j) has been

used, and set $f_{lj} = f_{lj} - \varepsilon_d$. Continue the flow change indicated by the first element of the labels until the node s is reached. Discard the labels and return to routine A. In Figure3.2, the diagram for the above Maximum Flow Algorithm (MFA) is given as follows.



(a)



(b)

Figure3.2 Labelling Algorithm:
(a) Initial flow and first labelling; (b) Second Labelling

Where the number in the brackets are $f_{i,j}$ and $C_{i,j}$. Initial edge flows are zero. According to the above algorithm the following steps can be executed:

1. Label the source [-, ∞].

2. Label nodes 1 and 2 with [s, 5] and [s, 4], respectively.

3. Label node 4 with [2, 4].

4. Label nodes 3 and 6 with [4,1] and [4, 2], respectively.

5. Label node 5 with [3, 1].

6. Label the sink (destination) with [5, 1].

There is a breakthrough and the augmenting path is indicated by➡. The first labelling procedure the flow of 1 through the edges [s, 2], [2, 4], [4, 3], [3, 5], and [5, d]. The second labelling is shown in Figure3.2 (b). The third labelling is attempted but no breakthrough occurs.

The algorithm terminates and produces the final solution: $x_{s2} = x_{s1} = x_{13} = x_{24} = x_{35} = x_{46} = x_{5d} = x_{6d} = 1$, and the flow value is 2.

Assuming that the algorithm terminates, the last labelling does not reach the sink node *d*. Let S be the set of nodes labelled in the last labelling attempt and $\overline{S}$ the set of unlabelled nodes. If an edge (*i, j*) is directed from S to $\overline{S}$, it must be saturated, that is, $f_{ij} = C_{ij}$; otherwise, j would have been labelled when *i* was scanned. Also, all edges (*j, i*) from $\overline{S}$ to S must have zero flow; otherwise, *j* would have labelled when *i* was scanned. Observe that s belongs to S and t belongs to $\overline{S}$. It is fairly obvious that the flow value is not greater than the sum of capacities of any set of edges (called a cut) which contains at least one edge of every path from *s* to *d*. Hence the flow value *f*(*d*) is optimal and equals the sum of capacities of the edges between S and $\overline{S}$. We state this as the max-flow min-cut theorem.

3.4.1.2 Ford-Fulkerson's Algorithm

When the labelling algorithm terminates, the flow *f*(*d*) is optimal and equal to the capacity of the minimum cut.

The question of whether the algorithm always terminates also needs to be considered. To see that it does if all initial edge flows and capacities are integer, we need to make two observations. First, the algorithm adds and subtracts only and does not introduce fractional flows. Second, if *d* is labelled the flow value is increased by at least one unit. Since the flow value is bounded from above (e.g. by $\sum C_{i,d}$ which is finite) the labelling algorithm must terminate.

However, unless we better define the labelling process (process A) the algorithm can be inefficient in some pathological cases. Modify the capacities of the network in Figure3.2 as shown in **Figure3.3** and assume that M is a very large number. If the labelling algorithm starts with $f(d) = 0$ and alternatively uses the same augmenting paths as shown in Figure3.2, it will require 2M iterations of routes A and B to find the optimum flow value $f(d) = 2M$. Here the number of iterations depends on the problem capacities.

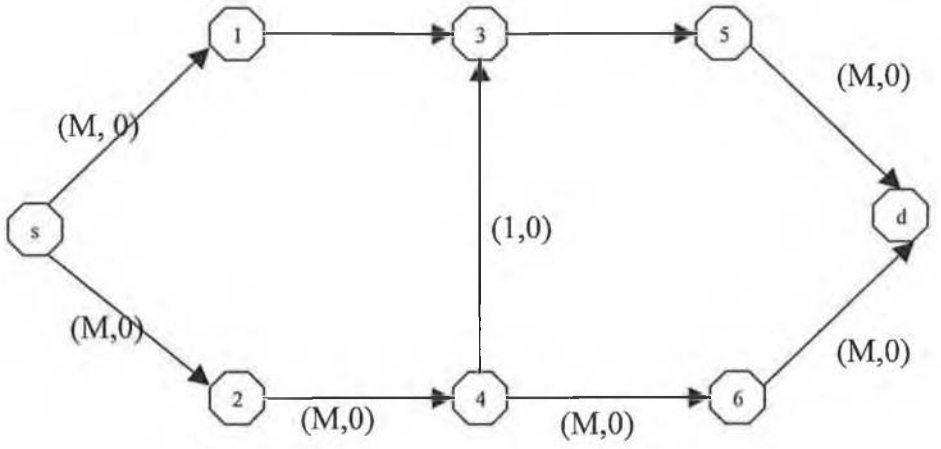


Figure3.3 Worst-Case Performance

Edmonds and Karp [1972] corrected this deficiency and showed that if the labelling procedure always uses the augmenting paths as short as possible, its time complexity is $O(nm^2)$, in an n-node m-edge network. Had we used the shortest augmenting path in the example shown in **Figure3.3** we would have used routines A and B only twice. The Ford-Fulkerson's algorithm will be implemented by C++ in a PC basis.

3.4.2 K-Shortest Path Algorithm

In [15], MacGregor, *et al* proposed the K-Shortest disjoint Paths (KSP) algorithm. The algorithm has two advantages over the Ford-Fulkerson's algorithm; it is much easier to control the length of paths found and that it is much easier to implement. However, the major disadvantage of the KSP algorithm is that it does not guarantee that the maximum amount of flow is found. The KSP was discussed in detail in [23]. The advantages and disadvantages of the KSP when compared with the Ford-Fulkerson's algorithm were discussed and two algorithms were compared in terms of performance. In [23], it has also been shown that the amount of paths found by the KSP is more than 99.9 % of that found by the Ford-Fulkerson's algorithm. Hence, the KSP can be used in most

circumstances that the Ford-Fulkerson's algorithm can be used and it offers the added advantages given above.

Here we introduce all the notation used in the KSP algorithm. In the given network G(V, E) where V is denoted a set of nodes and E a set of spans, we define two matrices R and C, where $R_{i,j}$ and $C_{i,j}$, the entries $(i, j)$of matrix R and matrix C, represent the maximum number of channels and the active number of channels carried on the span (i, j) respectively. $R_{i,j}$ is always greater than or equal to $C_{i,j}$. Note that there are $R_{i,j} - C_{i,j}$ spare channels on each span and these can be used to re-route the disrupted working paths without any increase in the network cost

The KSP can be implemented in two different methods, i.e. disjoint span KSP and non-disjoint span KSP. The disjoint span KSP algorithm finds disjoint paths between node pairs, i.e. paths do not share any common spans; the non-disjoint KSP algorithm does not have this restriction.

In the disjoint span KSP implementation, the basic idea is to perform Dijkstra's algorithm a number of times, removing spans between each iteration [8].

First, the Dijkstra's algorithm is used to get the shortest path in terms of the real distance between the specified nodes pair (say, s, d); Next, all spans on the shortest path are removed from the network topology. Based on the updated network, Dijkstra's algorithm is invoked again to find the second shortest path, and then all spans on the second shortest path are deleted from the network topology. The procedure continues until k shortest paths are found or no more paths can be found, i.e. the network is disconnected between nodes s and d. Pseudo-code for disjoint span KSP is given In Figure 3.4.

In the non-disjoint span KSP, Dijkstra's algorithm is again used to find the shortest path between nodes s and d.

The network topology is then updated by deleting the spans on which the capacity carried is minimal over all spans constitute. The capacity on all other spans that make up the shortest path is then reduced by this minimal capacity. Dijkstra's algorithm is again performed followed by the network topology and capacity update until either k shortest paths are found or no more path exists between nodes s and d. Pseudo-code for the non-disjoint form of the KSP algorithm is given in Figure 3.5.

Variables:

- $C_{i,j}$ Entry (i, j) of capacity matrix C storing the capacity of link connecting node i and j.
- $E_{i,j}$ Entry (i, j) of span adjacency matrix E, where it is a "1" if a link between nodes i and j exists and a 0 otherwise.

Given:

- s Source node.
- d Destination node.
- P An array to store the spans on the recent shortest path.
- M Number of paths found so far between nodes pair s and d.
- K Number of paths expected to find.

Procedure:

Step 1. Call Dijkstra's algorithm to determine the shortest path between nodes s and d in the network and store the nodes on the path in the array P.

Step 2. Update E by converting the entries in P from 1 to 0.

Step 3. If M < K and any path exists between nodes s and d, go to step 1, otherwise, the procedure stops.

Figure 3.4 Pseudo-code for the Disjoint Span KSP algorithm

Procedure:

Step 1. Perform Dijkstra's algorithm to obtain the shortest path between nodes (s, d) and store it in the array P.

Step 2. Compare all the capacity carried on the spans in the array P to find those spans with minimum capacity and delete these spans by updating the network adjacency matrix E.

Step 3. Update C by subtracting the above minimum capacity from the capacity on the other spans in the array P.

Step 4. If K > M and any path exists between nodes s and d, go to step1, otherwise, the

procedure stops.

Figure **3.5** Non-disjoint span KSP algorithm

Compared with the disjoint span KSP, the non-disjoint span KSP can find the more paths. However, the non-disjoint span version of the KSP takes longer to obtain solutions. For example, survivability techniques do not require very fast operation since they are design time technique, it is used to place the spare capacity in a network to meet requirement of restoration of disrupted working traffic upon the most-likely-happened network failure minimizing network cost at design time, so it is not very sensitive to execution time. Hence, the non-disjoint span KSP is most suitable method because it finds more paths to place spare capacity. Conversely, the objective of restorability techniques is to determine how to re-route the disrupted working paths utilizing the existing spare capacities as quick as possible, so, the disjoint span KSP is a better option. Note that the two types of KSP algorithms find almost as many paths as the Ford-Fulkerson's algorithm and offers some extra advantages. The comparison between two types of KSP algorithm will be given in terms of total paths and execution time below.

3.4.3 Matrix Maximal Flow (MMF) Algorithm

The Maximal Flow algorithm (Ford-Fulkerson's algorithm) can be used as an accurate approach to find the maximal flows between a pair of nodes in a network. However, the algorithm has the following disadvantages.

a) The maximum length of paths can not be controlled. It is often useful to be able to control the maximum length of the feasible path because long paths, in general, are wasteful of resources and result in losing the synchronization of traffics in different feasible paths.
b) It has high time complexity. Its time complexity restricts it from being applied to large-scale network [23].

The KSP algorithm was developed to overcome these problems to some extent, However, it does not guarantee the maximum account of the feasible path other than 99.9% on average [23].

Here, a new algorithm called the Matrix Maximal Flow (MMF) is proposed. The MMF algorithm is easier to implement than the Ford-Fulkerson's algorithm and it also provides a way to limit the length of the feasible path found in terms of hops. It has the

higher time complexity than the Ford-Fulkerson's algorithm in a single processor computer. However, it is derived to reduce its time complexity dramatically in a vector processor equipped computer from the conclusion of the paper [10] where the same matrix technique is employed. In this project the comparison of their time complexities will not be given because the vector processor is not available. But we will give an introduction of the Recursive Matrix (RM) algorithm [10], which uses the same basic technique, i.e. matrix multiplication, as the MMF algorithm does, in order to show their similarity in time complexity.

3.4.3.1 Recursive Matrix algorithm (RM)

In [10], the Recursive Matrix (RM) algorithm was proposed to find a disjoint shortest path subject to a particular hop-limit. The algorithm is based on the concept of the KSP algorithm, namely, it is used to find the specified number of the shortest paths, but replaces the shortest distance path as criterion with the minimum hops as criterion between a pair of nodes for further matrix calculation.

The theory of matrix on which the RM algorithm is based is described in [11], namely, if the matrix A is the binary adjacency matrix of a network with n nodes, the number of distinct path of length k in terms of hop between a pair of nodes (i, j) is the entry (i, j) of A^k for any positive integer k.

Consider that the number of disjoint paths between nodes (i, j) of a network will be determined by the RM algorithm. The RM works by first setting the entry (i, j) of the network binary adjacency matrix to zero. The matrix is then multiplied by itself until resulting matrix has a non-zero entry in the (i, j) position. The value of this entry represents the number of the distinct paths between nodes (i, j) having hops equal to the account of times the binary matrix has been multiplied by itself. Next, the algorithm can trace backward from destination j to source node i to mark one path between the source and destination nodes. All spans on this path are then removed by updating the binary matrix. These steps are repeated until the number of multiplication reaches the pre-set hop-limit or no more feasible paths can be found. Pseudo-code for the algorithm is given below.

A network G(V, E) V and E are a set of nodes and a set of spans in the network respectively.

Given:

- A Binary adjacency matrix.
- H Hop-limit of all searching paths.
- k Number of times the A has be multiplied by itself so far that is less than or equal to H.

Procedures:

Step 1.

Copy the adjacency matrix A to the temporary matrix B. Check the entry a_{ij} of matrix A, if a_{ij} is 1, set the entry b_{ij} of matrix B to 0.

Step 2.

Let $k = 2$.

Step 3.

Calculate B^k by multiplying B^{k-1} by B. If b_{ij}^k, the entry of B^k, is 0 and k is less than H, then there are not path from node i to j with k hops or less. Go to step 4. If b_{ij}^k is 0 and k is equal to H, the procedure ends because no path exists between nodes i and j with hops H or less. If b_{ij}^k is greater than or equal to 1, then b_{ij}^k paths between nodes i and j exist. Go to step 5.

Step 4.

Increase k by 1. If $k \leq$ H, go to step 3, otherwise, the procedure ends because no more paths with H hops can be found.

Step 5.

Decrease k by 1 in order to find a set of p such that the following equation is satisfied.

$$b_{ip}^k \bullet b_{pj} \neq 0 \qquad (1)$$

The above equation means that at least one path from node i to node p exist, as does the span (p, j) because Eq. (1) implies $b_{ip}^k \geq 1$ and $b_{pj} \geq 1$. Although there may be several ps that satisfy Eq (1), only one of them is selected because one path will only be determined during one iteration, the principle of the selection is First-Found-First-Select (FFFS). Since the span (p, j) is now used in one of the paths, b_{pj} is set to zero to prevent it being used in other paths and hence results in a set of link disjoint paths. Then j is replaced by p. If k is greater than or equal to 2, repeat step 5. Otherwise, a path with k hops is found and the temporary adjacency matrix B is updated by deleting all spans constitute the found path, then go to step 2 to search for the other possible paths.

3.4.3.2 Matrix Maximum Flow (MMF) algorithm

In [10], the RM algorithm has been shown to improve significantly its time complexity in a vector processor equipped PC over the Ford-Fulkerson's algorithm. Due to the crucial role of vector processor in reducing the time complexity of matrix calculation, we extend the matrix technique to be used to find maximum flow in a network . So we propose the Matrix Maximum Flow algorithm (MMF) based on the same technique. The MMF can be employed to find the maximum flow between a given pair of nodes with the specified hop-limit. The detailed algorithm is presented below.

A pseudo-code version of the MMF algorithm is below.

Given:

- A Binary adjacency matrix of the network.
- C Capacity matrix where $C_{i,j}$ is the amount of the available capacities on the span (i, j).
- T An array storing the path found.
- H Hop-limit of all searching paths.
- k Loop iterator.

Procedures:

Step 1.

Initialize the matrix A by setting its entry (i, j) to zero.

Step 2.

Increase k by 1 until a_{ij}^{k}, the entry of A^{k}, becomes nonzero. Which represents that k distinct paths exist between the node pair (i, j).

Step 3a.

Determine the set of p such that

$$A^{k-1}(i, p).A(p, j) \neq 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (2)$$

Then choose the p^0 such that the capacity on the span (p^0, j) is maximal over the set of span (p^w, j) for all p^w (is the element of p) and store p^0 to the array T.

Step 3b.

Repeat the following procedure:

$$j = p^0;$$

$$k = k\text{-}1;$$

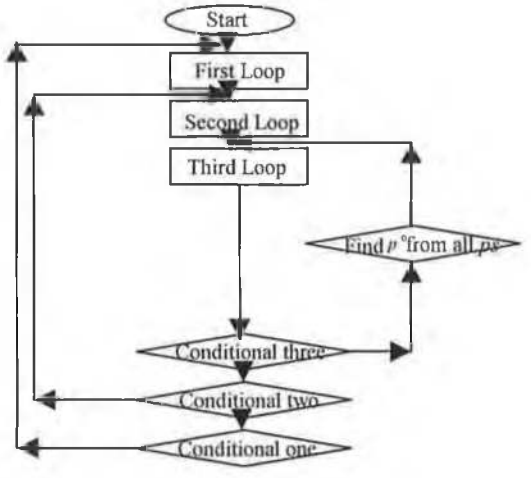
go to step 3a until k becomes 1. Then find span (n, m) on which the minimal

capacity D_{min} exists over all spans constituting the found path by tracing backward the array T. Finally update C by subtracting the capacities on the spans on the found path by the D_{min}, and A by set its entries, corresponding to these spans on the found path carrying the D_{min} capacity to 0. When step 3 ends go to step 5.

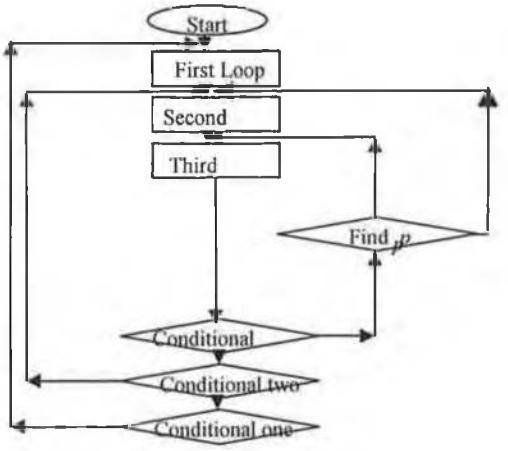
Step 5:

Go to step 2 until (k > H) or enough capacity found.

As discussed above the MMF algorithm is similar to the RM algorithm in terms of using the matrix. Figure 3.6(a) and Figure 3.6(b) give the simplified flowchart version of the RM algorithm and the MMF algorithm in order to compare theatrically their time complexities. Figure 3.6 (a) and Figure 3.6 (b) show the RM algorithm is almost the same flowchart as the MMF algorithm except their third loops. In the RM algorithm the procedure jumps out of the third loop once a p is found whereas the MMF algorithm jumps out of the third loop if and only if the third loop is completed in order to get the p^0 from a set of p, so in the worse case of the RM algorithm that p is found when its third loop is completed, the RM algorithm takes the same execution time as the MMF algorithm does. Here, the relative time complexities of the two algorithm can be derived from two flowchart. Assume that the time complexity of the RM algorithm is TC, and then the time complicity of the MMF algorithm is $2TC$, so the execution time of the RM algorithm is two times faster than that of the MMF algorithm.



(a) The MMF algorithm



(b) The RM algorithm

Figure 3.6 The comparison of two Flowcharts of the RM algorithm and the MMF algorithm

In [10] it has been proven that the RM algorithm is 10 times faster than that of the Ford-Fulkerson's algorithm does in a vector equipped super computer, i.e. Cray-2. It is reasonable to believe that the execution time of the MMF algorithm is shorter than that of the Ford-Fulkerson's algorithm in a vector equipped super computer. Our emphasis is not on the RM algorithm in this project the only MMF algorithm is implemented below.

3.5 Networks used to Investigate the above Algorithms

Five networks with associated demand matrices will be used to test the performances of the four algorithms described above, i.e. the Ford-Fulkerson's algorithm, the disjoint KSP algorithm, non-disjoint KSP algorithm and the MMF algorithm. The five test networks – both node location and interconnection capacities – have been described in [30]. The distance between a pair of nodes is given by our measurement in the diagram. In **Figure3.7**, the test network 1 is given. The other test networks are shown in appendix A.

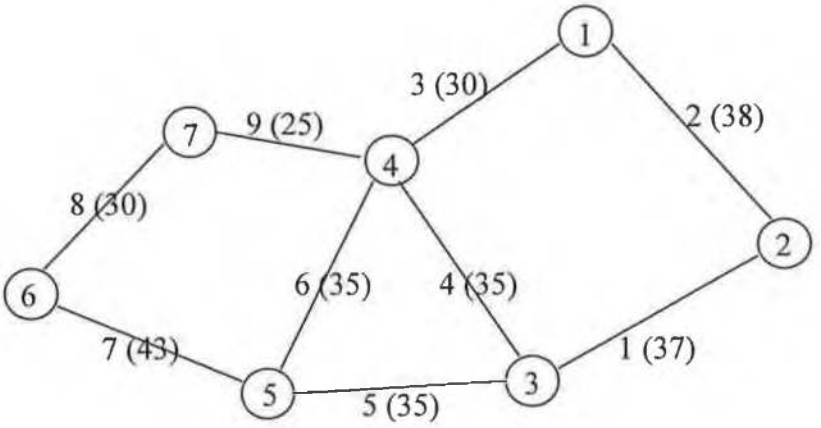


Figure3.7 Topology of Test Network 1

In **Figure3.7** each line represents a network link, and the two numbers next to each link represents that link capacity and the length of the span; the length of the span is in brackets. In Table 3.1, the detailed information about the five test networks is given.

	No. of Nodes	No. of Spans	Total Capacity	Link Length on Average
Network 1	7	9	45	32
Network 2	8	16	136	37
Network 3	16	28	397	32
Network 4	9	17	153	38
Network 5	9	16	136	37

Table 3.1 Information of the Five Test Networks

3.6 Results and Discussion

All four FPAs, i.e., Ford-Fulkerson's algorithm, non-disjoint span KSP algorithm and disjoint span KSP algorithm and the Matrix Maximum Flow algorithm, will be tested by using the five test networks mentioned above in order to determine their relative performance. The key performance indicator of interest here is the amount of feasible path found. The comparison of the execution time of two types of KSP algorithms is given as well. All algorithms described above are implemented in a standard PC with a Pentium II processor operating at 180Mhz. It is worth noting that for the purpose of comparison of the four FPAs algorithms the path hop-limit for the MMF algorithm and two types of the KSP algorithms is set to a very large number, say 7, in order that as many paths as possible can be found.

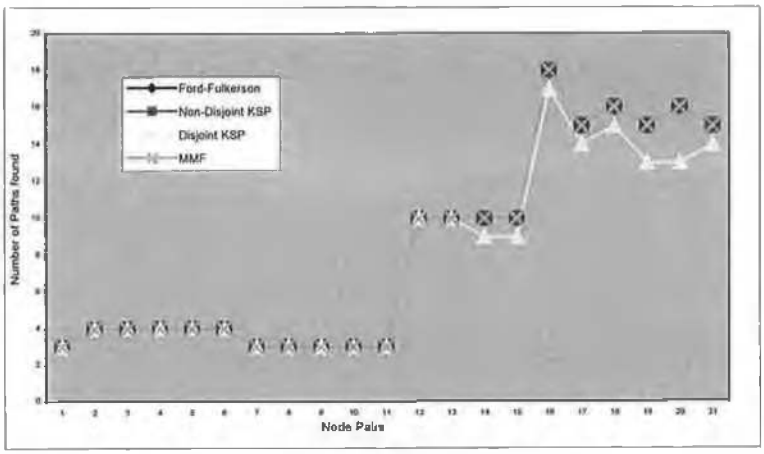


Figure 3.8 Results for Network 1

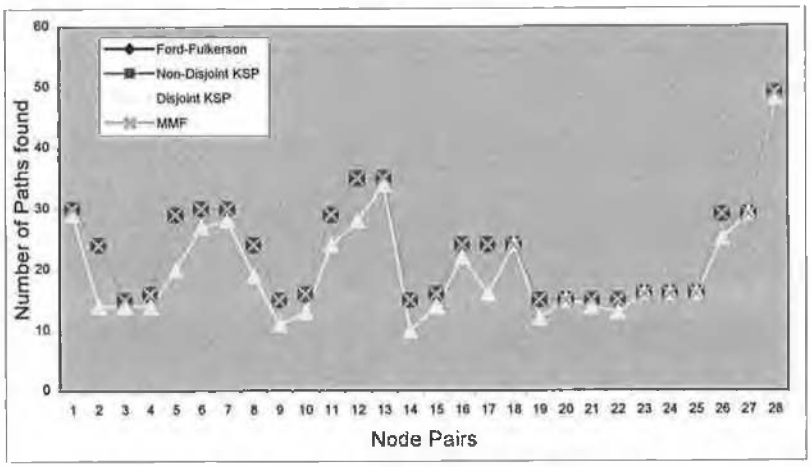


Figure 3.9 Results for Network 2

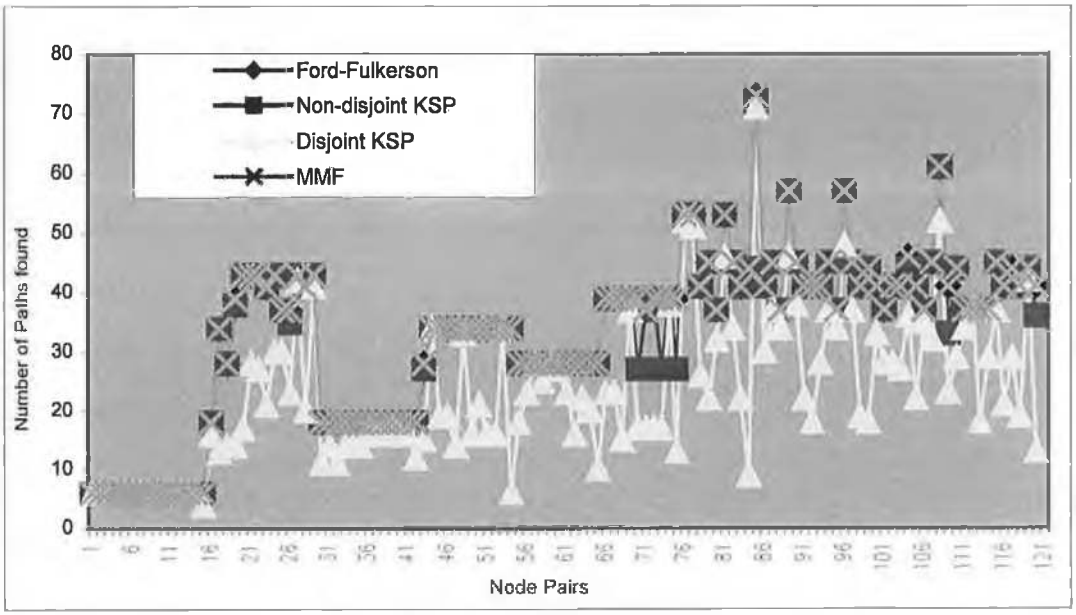


Figure 3.10 Results for Network 3

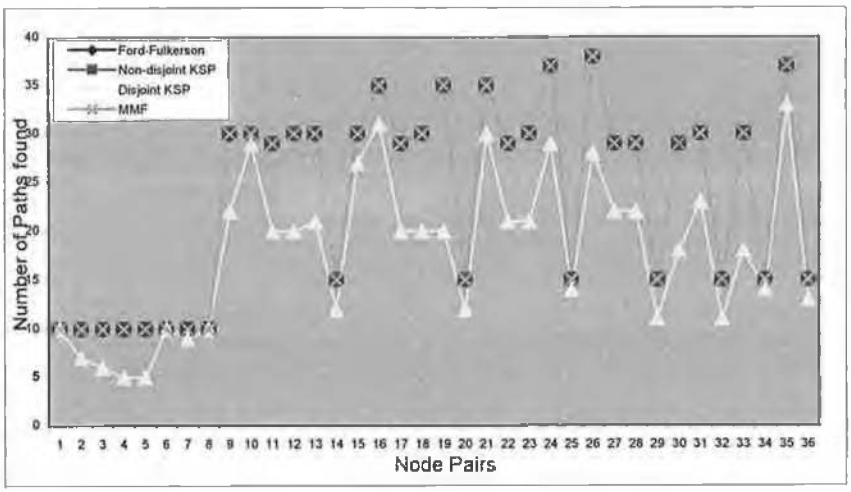


Figure 3.11 Result for Network 4

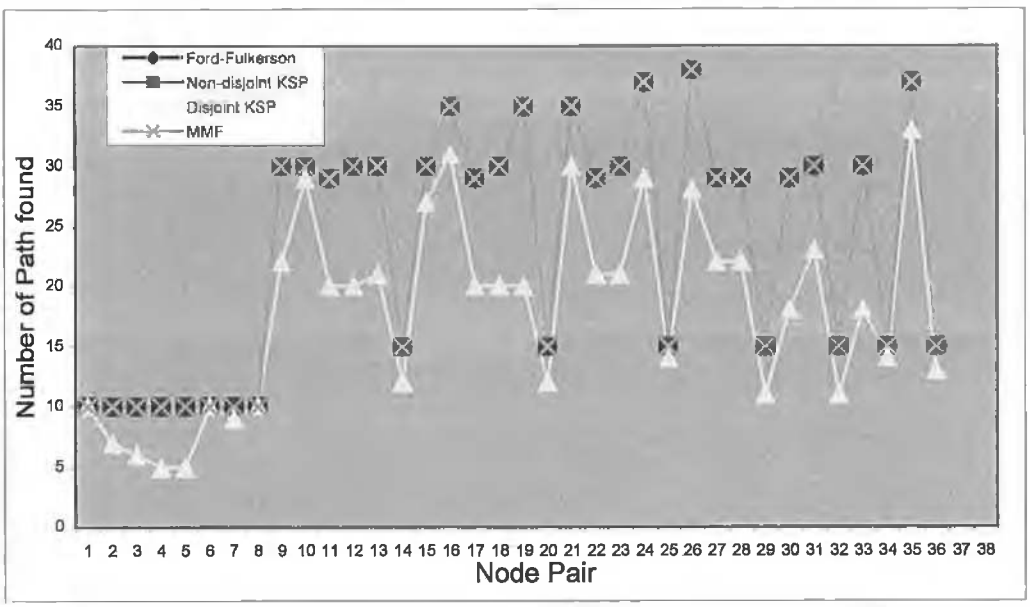


Figure 3.12 Results for Network 5

	Ford-Fulkerson's Algorithm(FF)	Non-disjoint KSP algorithm (N-KSP)	Ratio FF to N- KSP	Disjoint KSP algorithm (KSP-D)	Ratio of FF to KSP-D	Matrix Maximum Algorithm(MMF)	Ratio of FF to MMF
Network 1	173	173	1	162	0.94	173	1
Network 2	646	646	1	565	0.88	649	1
Network 3	3909	3858	0.98	2747	0.70	3909	1
Network 4	896	896	1	723	0.80	896	1
Network 5	846	846	1	644	0.76	846	1

Table 3.2 Results Generated by Four Feasible Path Algorithms for Five Test Networks

The results generated by applying each of the FPAs to each of the test networks are shown in the graphs in Figure 3.8 to Figure 3.12. For each node pair in each network, the amount of feasible paths found using each algorithm is determined. These results are graphed in aforementioned graphs. The overall results are summarized in **Table 3.2**. The information of the above figures associated with each test network includes:

1. For all possible nodes pairs between which the FPAs algorithms are executed to find feasible paths. The node pairs are ordered in the following principles: the source ID is always greater than that of the destination for one nodes pair and Node Pairs are sorted in the increasing order of the sum of their source ID and destination ID. For example, a set of node pairs is ordered as (1,2), (1, 3) … (3, 4), (3, 5) and so on.

2. The number of the feasible paths between a given pair of nodes with regard to the four FPAs algorithms described above.

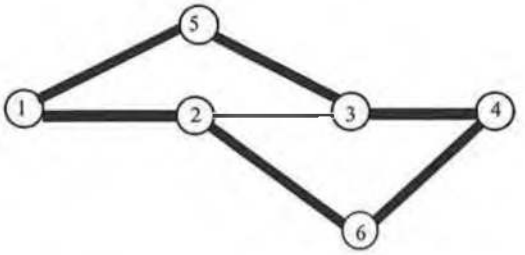
In Table 3.3 the execution times for the non-disjoint KSP algorithm and disjoint KSP algorithm are shown.

This is seen more easily in the table - The results in Table 3.2 show that the Ford-Fulkerson's algorithm finds the highest amount of feasible paths in each of the five test networks. Compared with Ford-Fulkerson's algorithm the non-disjoint KSP can find 100% in the test networks 1, 2, 4 and 5 and 97% in the test network 3 of the maximum amount of the feasible paths. The 3% loss of the feasible paths by the non-disjoint KSP

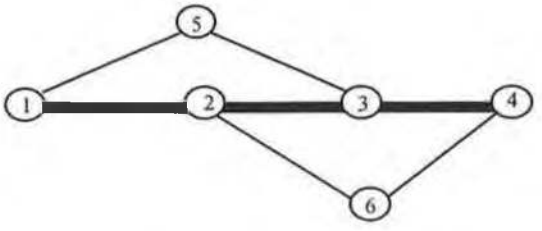
algorithm results from the fact that one feasible path found maybe restricts from finding the other feasible paths that share some common spans with the former one.

Figure 3.13 illustrates why the KSP algorithm can not find the maximal amount of feasible paths in some cases. We assume that each span in Figure 3.13 carries the same amount of capacity. At most, two span-disjoint paths can be found between nodes 1 and 4. One path is 1-5-3-4 and the other is 1-2-6-4 shown in Figure 3.13 (a). But if the path 1-2-3-4 is chosen, then only one span from nodes 1 and 4 can be found in total shown in Figure 3.13 (b). The KSP algorithm operating logic hop distances may make the latter choice because it has no basis for preference between the two equal length paths choice. And if the path via 1-5 and 2-6 to node 4 are longer than the 1-2-3-4 path, the KSP will always make the sub-optimal choice in the particular topology because it must include the shortest route. The issue is therefore how often this trap will arise in realistic networks. In the disjoint KSP algorithm the trap will arise if the particular network topology occurs where in the non-disjoint KSP algorithm the trap will arise if the particular network topology occurs and each spans in the network topology carries the same amount of capacity.

Not only does the non-disjoint KSP algorithm find almost as many feasible paths as the Ford-Fulkerson's algorithm, it also has some extra advantages over Ford-Fulkerson's algorithm, i.e. ease of implementation and control of hop-limits, that make it a good alternative to the Ford-Fulkerson's algorithm.



(a) The Maximal Paths found



(b) The trap resulting in the loss of one path

Figure 3.13 The trap in the KSP algorithm

	Network 1		Network 2		Network 3		Network 4		Network 5	
	Time (s)	Number of paths	Time (s)	Number of paths	Time (s)	Number of paths	Time (s)	Number of paths	Time (s)	Number of paths
Disjoint KSP	17	167	38	565	187	2743	41	723	39	644
Non-disjoint KSP	19	173	45	646	328	3858	47	896	52	864
Percentage difference	10%	3%	15.6%	12.6%	43%	29%	12.8%	20%	25%	25%

Table 3.3 Comparison of two KSP algorithms with respect to the time complexity

The results from the graphs and Table 3.2 shows that the MMF algorithm finds the same amount of paths as the Ford-Fulkerson's algorithm when the hop-limit of feasible paths is large enough, e.g. 7. The MMF algorithm combines the advantages of the Ford-Fulkerson's algorithm with the advantages of the KSP algorithm: it can find the same number of feasible paths as the Ford-Fulkerson's algorithm and it also provides the ability to control the number of hops. Note that the hop-limit we chose for the MMF algorithm is large enough in order that the MMF algorithm can find as many paths as possible because the smaller hop-limit may result in losing some paths in a network that have the larger hops than the preset hop-limit.

Apart from the above advantages, the MMF algorithm is believed to have the much lower time complexity than the other two algorithms in a vector processor equipped computer due to the use of matrix techniques. The effect of the vector processor on the reduction of the time complexity of matrix multiplication was proven in [10]. Here, The execution time of the MMF algorithm is not given because the vector processor is not available.

Table 3.3 shows that the non-disjoint KSP algorithm can find 3% - 29% more paths than the disjoint KSP algorithm for the test networks studied. However, this is at the expense of processing time; the non-disjoint version of the algorithm takes longer in the cases studied. As discussed above, the disjoint KSP algorithm can meet the need of real time services, so it is an important algorithm for use in restorability problem [12]. The

non-disjoint KSP algorithm, on the other hand, is an important algorithm for use in survivability problems.

3.7 Conclusion

The results and analysis have been shown in this chapter that the non-disjoint KSP algorithm compares well with the Ford-Fulkerson's algorithm in terms of the number of paths. Moreover, it is easier to implement and the number of hops in the paths can be controlled.

The new MMF algorithm combines the advantages of the Ford-Fulkerson's algorithm and the non-disjoint KSP algorithm: the MMF algorithm can find the same amount of feasible paths as the Ford-Fulkerson's algorithm does providing the control over the number of hops in the feasible paths as the KSP algorithm. The execution time taken the MMF algorithm in a vector-equipped computer is under investigation although it is believed to have less time complexity than the Ford-Fulkerson's algorithm in theoretical point of view as discussed in section 3.4.3.

Restorability techniques and survivability techniques have different requirement for the FPAs, i.e. restorability techniques are more sensitive to its execution time to meet the need of real time services whereas survivability techniques is more sensitive to network design cost at design time.

To illustrate the different requirements of restorability technologies and survivability technologies for the FPAs, the comparison of two types of the KSP algorithms, i.e. disjoint span KSP and non-disjoint span KSP, is given in terms of execution time and total feasible paths. It has been shown that the disjoint span KSP algorithm is a better option to meet the need of real time services in restorability technologies than the non-disjoint KSP algorithm does. However, in survivability technologies, the non-disjoint KSP algorithm has an obvious advantage that the disjoint algorithm does not, i.e. the amount of feasible paths found by the former is much more than that by the latter.

Chapter 4 Optimal Spare Capacity Placement in Mesh Survivable Networks

4.1 Introduction

In this chapter, we propose a new algorithm to deal with survivable network design problems based on the FPAs described in Chapter 3. In the survivable network design, the Spare Capacity Placement (SCP) algorithms are used to place spare capacity in a network to prevent them from network failures, e.g. a single span cut, in mesh survivable network design. In recent years, much work has been done on this area [2][4][7][29]. In general, we can divide the SCP algorithms into two categories: Linear or Integer Programming (LP/IP) approaches and heuristic approaches. In the LP/IP approach, the problem is formulated as a linear or integer problem and standard linear or integer programming techniques are used to obtain a solution. In the heuristic approach, the problem may be formulated in a complex manner (very non-linear discrete state space) and some heuristic approaches are used to solve the problem. These approaches often operate quicker than the LP/IP approaches since the heuristics are designed specifically for the problem under study and use knowledge of the structure of the problem to obtain good solutions. All the work to date using these two approaches assumes that the link cost function is linear. However, modern communications links are only available in set standardized capacities. Hence, the link cost function is stepwise in capacity rather than linear as described in Figure 4.1(a). Since the previous work has used linear approximations to the cost function, it is highly unlikely that they would find the optimal solution to the real problem; they may be useful for finding a first approximation to the solution. However, previous work has not addressed the non-linear cost function problem.

The rest of the chapter is organized as follows. In section 4.2 several types of link cost functions is described. In section 4.3 an IP based SCP algorithm is discussed. In section 4.4 a fast heuristic algorithm is illustrated for SCP problems. In section 4.5 a new algorithm that can deal with the stepwise cost function is proposed.

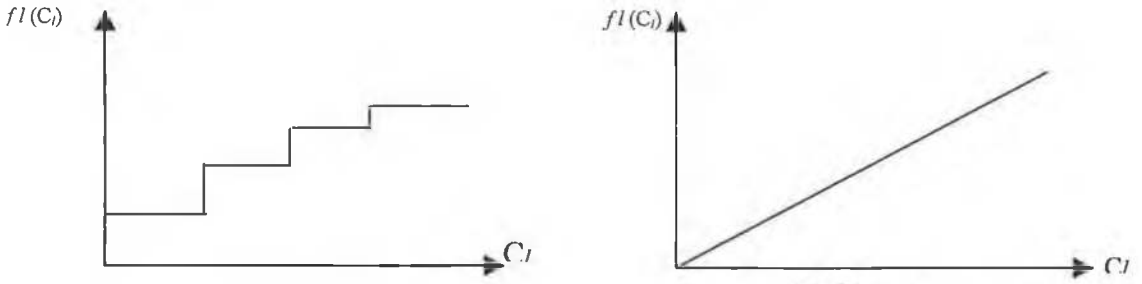
4.2 Link Cost Functions

In [15][16][31], linear link cost and concave link cost function have been adopted in survivable network design and optimal network design.

With the proliferation of high capacity optic fibre, the linear and concave cost functions no longer reflect the reality that optic fibre systems with specific capacities are available. Hence, the existing SCP algorithms developed for a linear link cost function may not applicable properly for the stepwise case. In the following section, several link cost functions will be illustrated.

Five models for link cost functions are reasonable, i.e. linear cost function, linear cost with fixed cost, concave link cost function, concave link cost function with fixed cost, stepwise link cost function all of which are shown in Figure 4.1. As we described above, stepwise link cost functions most closely reflects reality. The other four link cost functions are approximations of stepwise link cost functions, It is worth noting that we assume that a link cost function is always a non-decreasing function of its capacity; a system in which the cost decreases with increasing capacity does not make sense

In [32], Kerner *et al* discussed real costs associated with the installation of both metallic and optical fiber facilities on an interoffice network. In both cases a large cost is associated with channel construction and much of the remainder consists of link costs, which depend on capacity. We can expect to encounter similar cost types when dealing with the transmission network.



(a) Stepwise **(b)** Linear

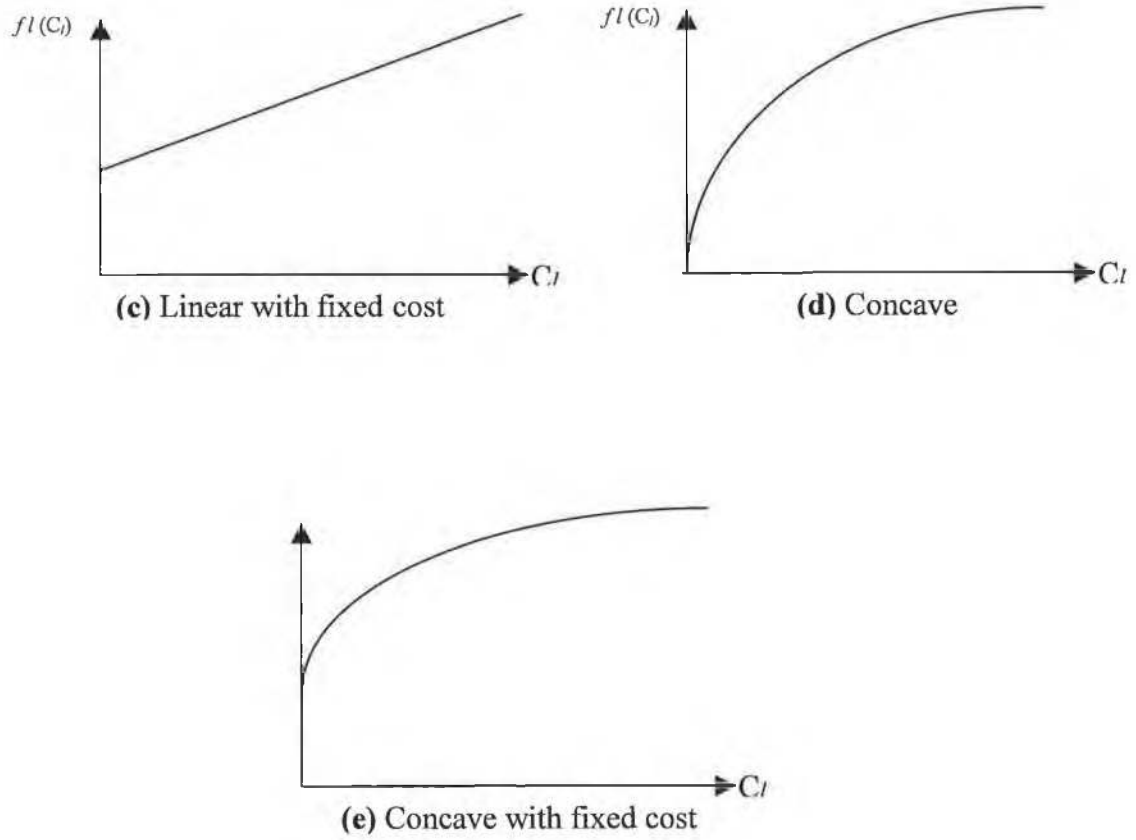


Figure 4.1 Different Types of Link Cost Function

The construction cost is proportional to distance, while acknowledging that this is a gross assumption. Digging up streets will be more expensive in urban than rural areas and also will depend on the terrain. All we need for our model is a total construction cost for a particular link, so we do not need to make that assumption. Links costs will depend on distance, but again, as distance is constant for a particular link, this is not so important. Link cost will depend on capacity, however, as the capacity determines the number, size and type of links required.

Network optimization problem has been studied for stepwise and concave link cost functions [33][34]. However, little work has been done on the Spare Capacity Placement (SCP) problem where the link cost function is stepwise. As optic fibres are only available with specific capacities, and the amount of capacity on a link contributes significantly to its cost, the stepwise link cost function shown in Figure 4.1(a) is the

closest to reality. This is why the stepwise link cost function is chosen for the SCP model.

As discussed above, all SCP algorithms to date were based the assumption that the link cost function is linear in capacity. It is unlikely that the solution found by these SCP algorithms using a linear cost function is coincident with the optimal solution to the SCP found using the stepwise cost function. However, it is likely that the solutions based on the assumption of the linear cost function may be close to the optimal for the stepwise cost function problem when the stepwise link cost functions are close to be linear. A new heuristic SCP algorithm is proposed here to solve the SCP problem when stepwise cost functions are used.

In case the link cost function is assumed to be linear, the LP/IP based SCP algorithm can be used to obtain one optimal solution [14][15][16][21]. But, the very large execution time of these algorithms limits their application in large-scale networks. Many heuristic algorithms have been proposed for large network SCP problems. While these algorithms operate much quicker than the LP/IP based ones, and, hence, they can be applied to large networks, the results they obtain for small networks are poor, leading us to believe that the results they obtain for large networks can be improved on [18][19].

First, we describe the IP based SCP algorithm as proposed in [21] and one heuristic SCP algorithm as proposed in [18]. Next, a new heuristic algorithm to solve the SCP problem for a stepwise is proposed.

4.3 IP-Based SCP Solution Techniques

Mesh survivable networks use Digital Cross-connected Systems (DCS) to minimize the amount of spare capacity required to re-route traffic in the case of failed. In mesh survivable networks the spare capacity on one span can contribute to the survivability of other spans, network redundancy, on the other hand, is not dedicated to restoration of one span. These networks are called "mesh" not to imply that the network is a full mesh, but to reflect the ability of the rerouting mechanism to exploit a mesh-like topology highly diverse and efficient rerouting of disrupted traffics.

LP/IP solution techniques make it possible to obtain optimal solutions to SCP problems in which the link cost function is linear in capacity. Due to the granularity of capacity on network spans, the IP formulation of SCP problems is a more realistic formulation

than the LP formulation. However, the LP formulation is still applicable in transport networks where the capacities of the spans are so huge that the granularity of the capacity can be neglected.

4.3.1 IP based SCP algorithms

4.3.1.1 Previous works on the LP/IP based SCP algorithm

In the recent decade a lot of the LP/IP based approaches to solve the SCP algorithm have been proposed. In [14] one approach has been to develop constraints for the SCP problem based on a network's cutsets after the failure of a single span. A cutset is a set of spans which when severed divide a connected network into two distinct parts. The min-cut max-flow theorem provides constraints on survivability through the set of minimum cuts in the network [35]. Which cutsets limit the maximum flow possible between those nodes seeking feasible paths after a failure is dependent on the spare capacity in the network. It is usually impractical to include all cutsets in the constraint set since the number of cutsets is an exponential function of the size of the network. Choosing cutsets to populate the constraint set iteratively after intermediate spare capacity designs can reduce the size of the constraint set dramatically. Prior work in [3] and [20] populates constraint set with cutsets iteratively, choosing cutsets according to a heuristic, until the solution provided a spare capacity placement which was 100% span restorable and minimized total spare capacity.

A more recent approach has been to specify flow constraints based on a suitable set of predefined routes over which path-sets must be implemented [12][22]. An IP formulation using this approach which optimize the placement of spare capacity in 100% span restorable network was reported in [12]. In such a formulation a fixed working capacity layout is given, and the constraint set is based on eligible restoration routes between each pair of nodes terminating a span. When the IP completes, the total flow feasible along those restoration routes is adequate to restore the lost capacity of any span cut in [19]. [37].

4.3.1.2 An IP based SCP algorithm

In this section, an IP based SCP algorithm proposed in [21] will be detailed. Unlike the IP's presented in [12] and [22], it uses a highly diverse route set that exploits mesh-like topologies to optimize working and spare capacity or only spare capacity, and considers both span and path restorable networks. Comparing these basic approaches [12][

14][22], it is obvious that when the constraint set is formed using eligible routes instead of cutsets, it only has to be defined once, and a solution will be ensured in a single linear or integer program run with no iteration. Moreover, while either approach specifies the optimal spare capacity per span, the route-based approach also yields details of the actual paths used to restore each span failure. This information is helpful when evaluating the performance of a distributed restoration mechanism operating in the survivable network design [36], and useful in a centralized restoration paths.

The following notations are used:

- C_i Cost of a channels (working or spare channels) assigned to span *i*.
- S Number of network spans in the network.
- H Limit to the number of hops on a path that can be used to reroute traffic in the case of failure.
- L_i^r The survivable level required for demand pair *r* upon the failure of span *i*. $0 \leq L_i^r \leq 1$ (for 100% network survivability, L_i^r =1 for all demand pair *r* and all spans *i* in a network, $L_i^r = 0$ represents that the demand *r* will not be recovered upon the failure of span *i*)
- D The number of node pairs that have nonzero demand between them.
- D_i Number of demand pairs affected by the failure of span *i*.
- d^r Number of demand units between node pair *r*.
- X_i^r Number of working channels lost from demand pair *r* upon the failure of span *i*.
- P_i^r Number of feasible restoration routes for demand pair *r* upon the failure of span *i* that do not violate the pre-set hop-limit.
- $f_i^{r,p}$ The restoration flow through the p^{th} restoration route for demand pair *r* upon the failure of span *i*.
- $\delta_{i,j}^{r,q}$ Take the value of 1 if the p^{th} restoration route for demand pair *r* after the failure of span *i* uses span *j*, and 0 otherwise.
- s_j Number of spare channels placed on span *j* for network failures.
- w_j Number of working channels carried on span *j*.
- Q^r Total number of working routes available to satisfy the demand between node pair *r*.

- $g^{r,q}$ The working channels required on the q^{th} working route to satisfy the demand between node pair r.
- $\zeta_j^{r,q}$ Take the value of 1 if the q^{th} working route for demand pair r using span j.

The objective function is:

$$\text{Min} \left\{ \sum_{j=1}^{S} C_j . s_j \right\} \qquad (1)$$

The constraints to be satisfied are:

1) Restoration flow meets target restoration levels for each demand pair r:

$$\sum_{P=1}^{P_i^r} f_i^{r,p} \geq \left\lceil X_i^r \times L_i^r \right\rceil \quad \forall r = 1,2,...,D. \ \forall i = 1,2,..., S. \qquad (2)$$

2) Span j's spare capacity is sufficient to meet the simultaneous demands of all node pairs affected by any one span failure:

$$s_j - \sum_{r=1}^{D_i} \sum_{p=1}^{P_i^r} \delta_{i,j}^{r,p} \bullet f_i^{r,p} \geq 0 \qquad (3)$$

3) The total demand lost from demand pair r after the failure of span i is the sum of the flows over working routes of the demand pair r traversing span i:

$$\sum_{q=1}^{Q^r} \zeta_i^{r,q} \bullet g^{r,q} = X_i^r \qquad \forall r = 1,2,...,D. \ \forall i = 1,2,..., S. \quad (4)$$

4) $f_i^{r,p}$, $g^{r,q} \geq 0$ and integer.

5) s_j , $w_j \geq 0$ and integer.

As formulated, the IP formulation can be adapted to optimize spare capacity placement for either a span or path restorable network. If a span restoration design is desired, the

set of all node pairs affected by a failure is restricted to just the single pair of nodes terminating the severed span, i.e. $D_i = 1$, and $X_i^r = w_i$.

In a path restorable network it is advantageous to release the surviving portions of a cut working path and make those paths available to the restoration process. This is called stub release. Stub release is an option in a path restorable network because span restoration only replaces the cut portion of a connection. The channels occupied by the affected demands can be optionally released at the time of the failure and added to the pool of spare capacity available for restoration. To represent stub release in the IP formulation, constraint 2 is replaced with constraint 6 as follows :

6) Span j's spare dimensioning is sufficient to meet the simultaneous demands of all node pairs affected by any one span failure (first double sum) after releasing the surviving portions of cut paths (second double sum).

$$s_j - \sum_{r=1}^{D_i} \sum_{p=1}^{p_i^r} \delta_{i,j}^{r,p} \cdot f_i^{r,p} + \sum_{r=1}^{D_i} \sum_{q=1}^{Q} \zeta_j^{r,q} \cdot \zeta_i^{r,q} \cdot g^{r,q} \geq 0 \qquad (5)$$

$$\forall (i, j) = 1,2,..., S.\ i \neq j.$$

Figure 4.2 IP-based Formulation of SCP Problems

4.3.2 Implementation of IP Based SCP Algorithm

This section focuses on how to solve the IP formulation. The Branch and Bound algorithm [38] is employed to solve the IP problem. First, we investigate all variables in the IP formulation given above.

- **Classification of Variables in IP based SCP Formulation**

We categorize all variables in the above IP formulation into two groups, i.e. input variable and output variable. Input variable is defined as the variable that can be obtained beforehand as input of the IP formulation, output variable is characterized as the output variable of the IP formulation.

How the variable w_i and $g^{r,q}$ are used depends on the problem being solved. In practice, there are two types of networks over which the above IP formulation can be to place spare capacity to prevent them from failure. One is that the network has the fixed

topology and layout of working paths to meet the need of its demand, the other that the network has the fixed topology with layout of working paths unknown. In the first case w_j and $g^{r,q}$ are the constants as the input values, so the equation (4) can be removed from the IP formulation in Figure 4.2. In the second case w_j and $g^{r,q}$ are the variables, which must be determined as output values. In the project the only second case is considered for simplicity.

The input variables of the IP formulation include: C, S, H, L_i^r, D_i, X_i^r, P_i^r, Q^r, $\delta_{i,j}^{r,q}$, w_j, $g^{r,q}$. Its output variables include: $f_i^{r,q}$, $g^{r,q}$, s_j and w_j. The input variable are given by users or network designers, for example, X_i^r is determined by the layout of the working channels in a network, P_i^r, Q^r, $\delta_{i,j}^{r,q}$ can be obtained by the FPAs, e.g. the KSP and the MMF algorithms described in Chapter 3.

- **The Branch and Bound algorithm**

Lots of approaches have been proposed to solve IP problem. The Branch and Bound algorithm and Gomory's All-Integer Dual Algorithm are two of the most popular. In this project, the Branch and Bound algorithm is chosen to solve the IP problem. In the first step of the Branch-and-Bound method, LP-relaxation of our IP formulation (where LP-relaxation is derived by releasing the integer constraints of IP formulation) is solved. It optimal objective value is an upper bound for the optimal objective value of its original IP formulation. If the solution of the LP-relaxation is integer we are done: it is also an optimal solution of the original IP formulation. If not, the feasible region of the LP relaxation is partitioned into two sub-regions giving rise to two new IP-formulations, and two sub-formulations are solved respectively. The solutions of two sub-formulations are compared. In the case the objective value of one solution is less than that of the other and its solution is integer, the solution is the optimal solution of IP formulation. Otherwise, in the case both solutions of two sub-LP formulations are not integer, we choose the sub-LP formulation whose objective value is less than that of the other, and partition the sub-LP formulation into two sub-LP formulation further by branching any non-integer solution, say x_i. The procedure is repeated until the solution of IP formulation is approached. Note that the principle of selection of non-integer solution is to choose one that has the smallest index because it is the closest to the solution than the others with the larger indices and the Simplex algorithm is used to

solve the LP relaxation formulation. We illustrate the Branch-and-Bound algorithm by drawing a flowchart of the above procedure in Figure 4.3.

The general form of the Branch-and-Bound algorithm can now be formulated as follows:

$$\max\{\mathbf{c}^{\mathbf{T}}\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \mathbf{x} \text{ integer } \} \text{ or}$$

$$\text{mix } \{\mathbf{c}^{\mathbf{T}}\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \mathbf{x} \text{ integer } \}$$

We use the set NF to hold the nodes that are not excluded from further consideration; the variable z_L denotes the current lower bound on the optimal solution of the original model. The function $f : F \rightarrow R$ with F the feasible region is the objective function. By F^R is denoted the feasible region of a relaxation of the original model. Optimal solutions of sub-models are denoted by (z, x) with z the objective value corresponding to the optimal solution x.

Input:

Values for the parameter of the optimization model max $\{f(x) \mid x \in F\}$, with F the feasible region.

Out: Either

(i) The message: the model has no optimal solution; or

(ii) The solution of the model.

Step 0. Initialization. Define $F_0 = F^R$, NF = {0}, and $z_L = -\infty$.

Step 1. Select a label $k \in$ NF. Let max{f(**x**) $x \in F_k$ }, with $F \subseteq F^R$, be the sub-model associated with label k; call this sub-model S_k. Go to Step 2.

Step 2. Determine, if exists, an optimal solution (z_k, x_k) of S_k (note that z_k is an upper bound of max $\{f(x) \mid x \in F_k \cap F\}$). If S_k has no optimal solution, define $z_k = -\infty$. Go to Step 3.

Step 3. S_k is excluded from further consideration, if either one of the following situations occur:

(a) $z_k = -\infty$ (i.e. s_k has no optimal solution);

(b) $z_k < z_L$ (i.e. z_k is worse than the current best solution);

(c) $z_k \geq z_L$. If $x_k \in F$ (i.e. z_k is at least as good as the current best solution),

then define $z_L = z_k$ and go to step 5 with NF = NF\{k}; otherwise, go to Step 4.

Step 4. Partition F_k into two or more new subsets, say $F_{k_1}, \ldots, F_{k_s}$. Define, NF = (NF\{k}) $\cup$ {$k_1,\ldots,k_s$}. Go to step 1.

Step 5. Optimality test and stopping rule. If NF $\neq \Phi$, then go to Step 1. Stop the procedure when NF $= \Phi$; the current best solution is optimal. If there is no current best solution, i.e. $z_k = -\infty$, then the original model has no optimal solution.

Figure 4.3 The branch and bound algorithm

In Step 3(c) we consider the situation that $z_k \geq z_L$. For $z_k = z_L$, and $\mathbf{x}_k \in F$ we could exclude sub-model S_k from further consideration, since further branching may only lead to alternative optimal solutions. Note that the above procedure is applied to maximizing model. In case of a minimizing model, the Branch-and-Bound Algorithm can easily be adapted: e.g. z_L has to be replaced by the "current upper bound" z_u, $-\infty$ by ∞, and the inequality signs reversed.

- **The Solution to LP- Relaxation of IP Formulation**

From discussion above, the method to solve the LP-relaxation of IP formulation is of utmost importance. The PSIMPLEX procedure that is based on the revised simplex method is employed to solve the LP-relaxation problem. We will illustrate the PSIMPLEX procedure by a flowchart in Figure 4.4. The procedure being given is applied for minimizing model.

We assume the number of input variables is n and the number of constraints is m in our example, so the standard formulation of LP problem can be given as follows:

Min $\{c_1x_1 + \ldots\ldots + c_nx_n\}$

Constraints to be satisfied:

$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1$

$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2$

$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$

$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m$

and $x_1, x_2, \ldots, x_n \geq 0$ and integer.

To directly apply PSIMPLEX method some modifications of the above LP formulation is required and several artificial variables are introduced, so the above formulation is converted into a new and equivalent formulation as follows:

$$\text{Max} \quad x_{n+m+1} \qquad (6)$$

$$\begin{aligned}
\text{Subject to} \quad & a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n + x_{n+1} &= b_1 \\
& a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n + x_{n+2} &= b_2 \\
& \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots & \\
& a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n + x_{n+m} &= b_m \\
& a_{m+1,1}x_1 + a_{m+1,2}x_2 + \dots + a_{m+1,n}x_n + x_{n+m+1} &= 0 \\
& a_{m+2,1}x_1 + a_{m+2,2}x_2 + \dots + a_{m+2,n}x_n + x_{n+m+2} &= b_{m+2} \\
& \text{and } x_1, x_2, \dots, x_{n+m} \geq 0. &
\end{aligned}$$

Where

$$a_{m+1,j} = c_j, \; a_{m+2,j} = - \sum_{i=1}^{m} a_{ij}, \quad j = 1, 2, \dots, n$$

$$b_{m+2} = \sum_{i=1}^{m} b_i,$$

We have added the nonnegative artificial variables x_{n+i}, $i = 1, \dots, m$, into the structure variables $x_1, \dots, x_n$ to create a simple basis matrix B =I. Since

$$x_{n+m+2} = -(x_{n+1} + \dots + x_{n+m})$$

the variable x_{n+m+2} is the negative sum of the artificial variables. Clearly, we must have $x_{n+m+2} \leq 0$.

The revised problem has m+2 equations in n + m+ 2 variables. The basic feasible solution, if it exists, has m + 2 variables from the set $\{ x_1, \dots, x_n, x_{n+m}, x_{n+m+2}\}$, with $-x_{n+m+1}$ representing the optimal objective function value and x_{n+m+2} =0. In the following flowchart, the procedure is divided into two phrases. In phase I we maximize x_{n+m+1} subject to the constraints of (6). If max $x_{n+m+2} = 0$, we begin phase II with the objective function

$$x_{n+m+1} = -(c_i x_i + \dots + c_n x_n)$$

While keeping $x_{n+m+2} = 0$. We define the matrix $\overline{A}$ as

$$\bar{A} = \left[\begin{array}{c} A \\ \hline a_{m+1,1} \ldots a_{m+1,n} \\ a_{m+2,1} \ldots a_{m+2,n} \end{array}\right]$$

Which is A augmented by two last rows. Row m +2 is used to compute the relative cost vector p in phase I. row m +1 has the same role in phase II. We also introduce an $(m+2) \times (m+2)$ matrix which initially is:

$$U = \left[\begin{array}{c|c} I_{m \times m} & 0 \\ \hline 0 & I_{2 \times 2} \end{array}\right]$$

This first m row and columns of U will contain the inverse of B. The last two rows of U will be used to determine the vector entering the basis (row m +2 in phase I and row m + 1 in phase II).

Using this notation and initial values $x_{n+i} = b_i$, $i = 1, \ldots, m$, the computational steps in phase I and phase II are as follows:

Phase I:

Step 1. If $x_{m+n+2} < 0$, calculate

$$\delta_j = row_{m+2}(U).col_j(\bar{A})$$

$$= \sum_{p=1}^{m+2} u_{m+2,p}\, a_{p,j}, \quad j = 1,2,\ldots, n.$$

and continue. If $x_{m+n+2} = 0$, go to go phase II, step 1.

Comment: In phase I the objective function to be maximized is x_{m+n+2} and all cost coefficients except $c_{m+n+2} = 1$ are zero. The values of δ_j are the components of the relative cost vector p.

Step 2. If all $\delta_j \geq 0$, then x_{n+m+2} is at its maximum and no feasible solution to the original LP problem exists. If at least one $\delta_j < 0$, then the variable to be introduced into the basic set is x_k such that

$$\delta_k = \min_{\substack{1 \le j \le n \\ \delta_j < 0}} \delta_j$$

Step 3. Compute

$$y_i = row_i(U).col_k(\bar{A}) = \sum_{p=1}^{m+2} u_{ip} a_{pk}, \quad i = 1,2,..., m+2$$

Step 4. Calculate

$$\min_{\substack{1 \le j \le m \\ y_j < 0}} \left[\frac{x_i}{y_i} \right] = \frac{x_l}{y_l} = \theta$$

if $y_i \le 0$ for all i = 1, 2,..., m, there is no feasible solution. Otherwise, the variable x_l is eliminated from the basic set

Step 5. Calculate the new values of the variables in the basic solution

$$\overline{x_l} = \theta$$

$$\overline{x_i} = x_i - \theta y_i \qquad (i \neq l)$$

for $i = 1, 2, ..., m+2$, and

$$\overline{u_{lj}} = \frac{u_{lj}}{y_l}$$

$$\overline{u_{ij}} = u_{ij} - y_i \frac{u_{lj}}{y_l} \qquad (i \neq l)$$

for $i = 1, 2,..., m+2, j = 1,2,..., m$. Return to step 1.

Comment: the columns m+1 and m +2 of U do not change. The phase I iteration are continued until $x_{n+m+2} = 0$ or it is determined that no feasible solution exists. In the former case we go to phase II.

Phase II:

Step 1. Here we maintain $x_{m+n+2} = 0$. Compute

$$\gamma_j = row_{m+1}(U).col_j(\bar{A})$$

$$= \sum_{p=1}^{m+2} u_{m+1,p} a_{pj}, \qquad j = 1,2,\ldots, n$$

Step 2. Compute

$$\gamma_k = \min_{\substack{1 \le j \le n \\ \delta_j < 0}} \gamma_j$$

The variable x_k is selected to enter the basic set. If all $\gamma_j \geq 0$, then x_{n+m+1} is at its maximum value and the original problem is solved.

Step 3. Calculate

$$\gamma_i = \sum_{p=1}^{m+2} u_{ip} a_{pk}, \qquad i = 1, 2,\ldots, m+2$$

Step 4. Find

$$\min_{\substack{1 \le i \le m \\ y_i > 0}} \left[\frac{x_i}{y_i} \right] = \frac{x_l}{y_l} = \theta$$

If all $y_i \leq 0$, the objective function x_{n+m+1} can be made arbitrarily large. The computation is terminated. Otherwise, proceed to step 5.

Step 5. Calculate

$$\overline{x_l} = \theta$$

$$\overline{x_i} = x_i - \theta y_i \qquad (i \neq l)$$

for $i = 1, 2, \ldots, m+1$, and

$$\overline{u_{lj}} = \frac{u_{lj}}{y_l}$$

$$\overline{u_{ij}} = u_{i,j} - y_i \frac{u_{lj}}{y_l} \qquad (i \neq l)$$

for $i = 1,2,\ldots, m+1$, $j = 1, 2,\ldots, m$.

Return to step 1.

Figure 4.4 PSIMPLEX algorithm for LP problem

The Branch-and-Bound method with PSIMPLEX is very time consuming $O(2^N - 1)$, where N is the number of variables of the IP based SCP formulation given in Figure 4.2, although it can be applied to obtain an good solution to SCP problems. Note that the number of variables in the IP based SCP formulation is S*S*D, where S and D are the number of network spans, and network connectivity respectively. In practice, it works only for small and middle-scale networks, which does not make any practical use. For large-scale network (e.g. transportation network), more effective algorithms are required. In the following section, we will introduce the algorithm proposed by [18]. The algorithm is based on the heuristic principle, which lead dramatically improvement in the time complexity compared with IP/LP based SCP algorithms.

4.4 A Heuristic Algorithm for Spare Capacity Placement (SCP)

In [18] an useful and effective heuristic approach termed max-latching to the SCP problem has been proposed to find a reasonable solution to SCP problem for large scale networks. A straightforward heuristic of average case complexity $O(S)$, where S is the number of network spans, is faster than the IP based SCP algorithm $O(2^N - 1)$. The algorithm has two disadvantages; it can only be used for span network restoration and it is a local search SCP algorithm.

The main idea of the heuristic algorithm is as follows. When a single span failure occurs a FPAs is employed to find feasible paths on which spare capacity is placed to reroute the disrupted working paths minimizing network cost.

Assume a network graph G has S spans and N nodes and a vector **w** of working capacity (w_i) on each span *j*. C_i is the cost per channel on span *i*. The issue is how to specify *s*, the vector storing the amount of spare channels on each span *i* (s_i) so that the cost of the network, i.e. $\sum C_i \cdot s_i$, is minimised.

For every span *i* (taken one at a time) there is a set of feasible paths through the rest of the network at the event of the span *i* failure. The spare capacities on such paths are greater than or equal to w_i so that the w_i can fully be recovered at the event of a span *i* failure. The paths consist of circuit-like continuous channels each using an individual traffic unit (e.g. an STS-1 or STS-3 transport unit) on each span on feasible paths. For span restoration paths connect the end nodes adjacent to the failed span. The number of

channels traversing any span cannot exceed s_j. Working and spare capacities, w_i and s_i can only be integer numbers. For simplicity, we let all $C_i = 1$, In which case the capacity redundancy $(\sum s_j)/(\sum w_i)$, is the design efficiency measure.

Using this algorithm, the problem is solved as follows. Let P_i be a binary matrix of S rows, representing spans, by k_i columns each representing a distinct path, not exceeding hop-limit H, between the end nodes of span *i*, excluding span *i* itself. k_i is the number of eligible distinct (not disjoint) paths for restoration of failed span *i*. In this project the KSP algorithm is employed to determine feasible paths. Columns of P_i are sorted left to right in order of increasing weight (e.g., length in hops). Let $a(w_i, k_i)$ be a vector of the most-nearly-equal assignment of the required restoration flow for span *i* over the k_i routes, with placement of the excess allocation (due to whole number effects) on the lower numbered paths. For example a(10, 3) = (4, 3, 3), a(11, 3) = (4, 4, 3), etc. So the following formulation is obtained.

$$S(G, w) = \text{rowmax}[(P_1 \,.\, a^T(w_1, k_1)), (P_2 \,.\, a^T(w_2, k_2)), \ldots (P_s \,.\, a^T(w_s, k_s))]_{sxs} \quad (7),$$

is a sufficient (i.e. fully restorable) and reasonably efficient SCP solution. Rowmax takes the row-wise maximum of matrix elements. The idea is as follows: each product $P_i \,.\, a^T()$ yields a vector of the spare channels quantities required on other spans to restore span *i*, as if the span *i* was the only failure to consider. An S × S matrix is therefore formed where each column expresses the spare channel requirement on each span over all restoration pathsets in which it participates. The principle is that for any span *j* there will always be some other span *i* which will require more spare capacity on span *j* than any other span for realisation of the required pathsets. When this is true, we say that span *i* is the forcer of span *j*. Several span may equally force another span, so the forcer relationships are in general many to one.

Equation.7 expresses a simple principle through which adequate and reasonably efficient SCP solutions can be obtained compared to using IP. By its nature, *s* will always yield a fully restorable network but in general with excess spare capacity. Some lines of reasoning suggest, however, that this principle should be reasonably efficient: First, where all forcer relationships are 1:1, the heuristic algorithm would equal the IP if

the flow assignments to routes are also the same. Secondly, in a fully connected graph with all *w* equal, IP and, results of the heuristic algorithm would again be identical.

Herberg *et al* [16] therefore tested a procedure which each span in turn is considered a failure span, observes the spare channels required by the levelling flow assignment to routes and latches the maximum spare channels on each span as other spans are allowed to force the network spare channels. However, while Equation 7 is an expression of the basic max-latching concept. Where each span forces the others in isolation, a practical improvement is to allow spans to force the network in sequence and let the flow assignment stage for the current span first exploit the spares already forced by prior spans. While this improves the algorithm designs, it introduces dependency on the order of span selection. Several ordering principles have been investigated in depth in [18]. A simple ordering is by decreasing w_i / k_i, the idea being to let spans with the largest working flows relative to the number of restoration routes go first as they tend to be strong forcers. Subsequently spans will find their restorability partly or wholly satisfied by routing first to take advantage of the already forced spares. Only the flows requirement that is unroutable through the current state of the s_i maximums, is subjected to the flow assignment function a() and is allowed to further force the network spare channels. RM algorithm is used to find all feasible routes between a given pair of nodes (i.e. the endpoints of a failed span). A pseudo-code of the algorithm is presented in Figure 4.5.

- A Binary matrix of a given network topology where a_{ij} is the entry (i,j) of the matrix A.
- P Binary matrix where P_i is a binary matrix of S rows, representing spans, by k_i columns each representing a distinct route, not exceeding H hops.
- W Matrix storing the number of working channels where w_{ij} is the number of working channels on the span (i,j).
- F Matrix storing the number of feasible routes for the specified span failure where k_{ij} is the number of feasible paths between the disrupted span (i,j).
- S Matrix storing the number of spare channels for all possible network failures.
- SS Number of spans.
- N Number of nodes

- H Hops-limit for feasible paths

Initialization:

Initialise S by setting all entries of S to 0.

Procedure:

While (all spans in the network)

{

Comparing w_{ij}/f_{ij} and rank spans in the order of decreasing w_{ij}/f_{ij}.

}

For (all spans in the network and assume they fail in the order of decreasing w_{ij}/f_{ij}.)

{

Using the (FPAs) algorithm described in Chapter 3 to find all feasible paths (k_i) in order to obtain a(w, k). And then calculate $P_i.a(w_i\ k_i)$.

}

While (all spans in a network)

{

$S_{span}(i,j) = \text{rowmax}[(P_1 . a^T(w_1, k_1)), (P_2 . a^T(w_2, k_2)), ... (P_s . a^T(w_s, k_s))]_{sxs}$

}

Figure 4.5 Pseudo-code for max-latching Algorithm

4.4.1 Time Complexity of the max-latching Algorithm

Complexity analysis in [18] shows that the algorithm procedure, (coded as described, not with the matrix operations of Eqn.1) is theoretically O(S log S) in its dependence on network size. This reflects the succession of (i) finding each spans' hop limited eligible path-set, O(S), (ii) sorting spans in decreasing w_i/k_i (O(S *log* S) with a heap-sort) and (iii) allowing each span to act as a forcer of other spans, also O(S). These steps are consecutive no nested so the theoretical complexity is O(S *log* S). However, the coefficients in the O(S) terms heavily dominate execution time as the sort is very fast. In contrast, finding all eligible routes up to hop length H is $O(H\ d^H)$ in a network of effective average node degree d. But H and d are not dependent on the network size;

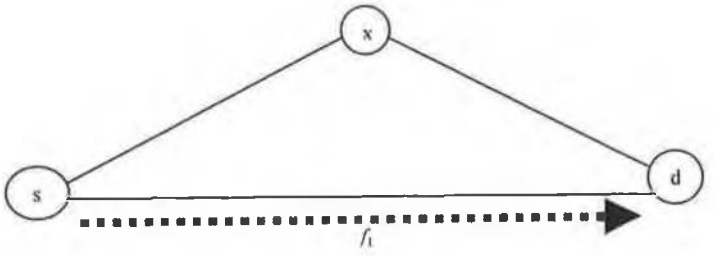
they only express the topological diversity of the network graph and the length limit for restoration.

4.5 A New Heuristic Algorithm for the SCP problem with Stepwise Link Cost Function

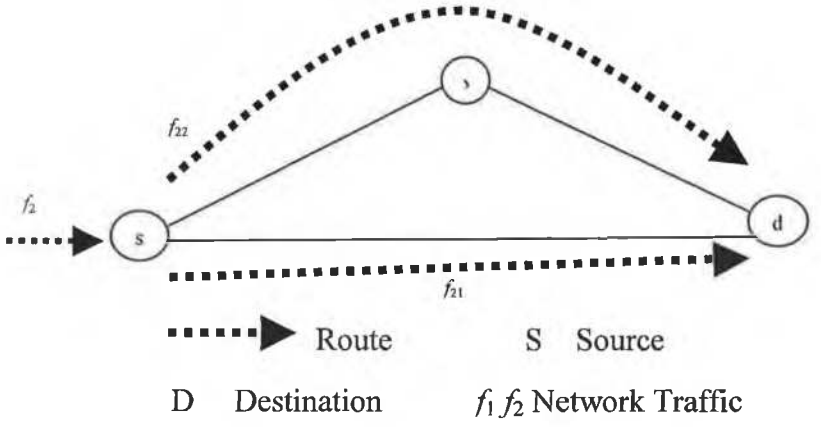
4.5.1 The Problem of the Existing SCP algorithm

Both IP based SCP algorithm and the heuristic algorithm described above was developed with the assumption that the link cost function is linear in capacity. In Equation 1, C_j is the cost of one capacity unit, i.e. STS-3, on the span j, that is, the cost of the span is proportional to the number of channels it carries on as described Figure 4.1(b). While the cost of setting the span, i.e. digging for optic fibre setting, and other relevant equipment, i.e. DCS, is considered the link cost function is like that described in Figure 4.1(c). So the equation (1) can be held if and only if C_j is constant where C_j may not be the same for different spans in the network. Therefore, the equation (1) can be used to obtain the minimal cost of the survivable network while the condition of linear link cost function is held.

In addition, the max-latching approach was developed based on the assumption that the link cost function is also linear in capacity. If this assumption is not valid, the minimum amount of spare capacities in the network may not be coincident with the minimum cost of the network. Say, there are two network strategies as described in Figure 4.6 (a) and (b), the capacity f_1 in the network 1 is greater than the capacity f_2 in the network2. In network 2, the capacity f_2 is divided in two, i.e. f_{21} and f_{22} where $f_{21} + f_{22} = f_2$. The f_{21} and f_{22} are placed in two distinct paths respectively as illustrated inFigure 4.6(b) whereas the f_1 in the network 1 is placed in one path in Figure **4.6** (a).



(a) Network 1 Strategy



(b) Network 2 Strategy

Figure 4.6 Two Network Strategies

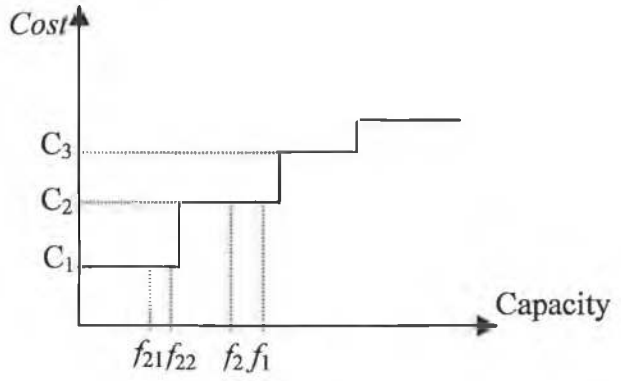


Figure 4.7 Stepwise Link Cost Function

For simplicity, we assume that the link cost functions of all spans in two networks are the same as described in Figure 4.7, where $C_1 > C_2$. Hence, it is easy to know the cost of network 2 with capacity f_1 is $C_2 + C_1$ whereas the cost of network 2 with capacity f_{21} and f_{22} is $C_1 + C_1$. $C_1 + C_1$ is greater than $C_1 + C_2$, so the cost of network 1 with capacity f_1 is lower than that of network 2 with capacity f_2 although the capacity f_1 in network 1 is greater than the capacity f_2 in network 2. Hence, the major procedure in max-latching algorithm that minimizes the amount of spare capacities required to minimize the cost of survivable network does not work properly in case the link cost function with its capacity is stepwise.

In order to address the problem, a new heuristic algorithm named Stepwise Capacity Heuristic algorithm (SCH) is proposed here. Before we start to illustrate the SCH algorithm the Addition Minimum Increment (AMI) algorithm will be introduced

because the SCH algorithm was developed based on the FPAs described in Chapter 3 and the AMI algorithm.

4.5.2 Addition Minimum Increment algorithm (AMI)

Finding "suitable" paths to place the spare capacity for restoring the failed traffic minimising the network cost is a crucial issue for survivable network design, the term "suitable" is characterised as a path on which placing spare capacity required results in the minimum network cost. There have been many works on addressing the issue [13][39]. The most popular algorithm proposed in 1959 is Dijkstra's algorithm that is used to find the shortest path in distance between a given pair of nodes. In Chapter 3, Dijkstra's algorithm was applied in the KSP algorithm to find the K shortest paths. Since the length of the paths contributes significantly to their cost, the shortest path is always coincident with the minimum path cost in case the link cost function of a network is approximately linear and concave in capacity [31]. But, in case the link cost function is stepwise in capacity, the shortest path is no longer coincident with the minimum cost path. We propose a new algorithm, AMI (Addition Minimum Increment), to find a path on which to place the specified amount of spare capacities with the minimum cost increment.

The basic idea of the AMI algorithm is based on the Dijkstra's algorithm. We assume the AMI path will be searched for between node *s* and *d* with the specified amount of capacity c_{sd}. We start by giving a permanent label 0 to the source node *s*, because capacity between *s* and itself is always zero. All other nodes get labelled ∞, temporarily, because they have not been reached yet. Then we label each immediate successor *i* of source *s*, with temporary label equal to the link cost increment after placing capacity c_{sd} on link (*s, i*). It is obvious that node *min* with smallest temporary label among these immediate successors is the node where there is the minimal increment of link cost if c_{sd} is placed on the link (s, *min*). Since all labels of s immediate successors *i* is nonnegative, there can be no smaller label than one of *min* from *s* to *i*. Therefore we make the label of *min* permanent. Next, we find all immediate successors of node *min*, and shorten their temporary labels if the label from *s* to any of them is smaller than by going through *min* (than it was without going through *min*). Now, from among all temporarily nodes we pick the one with the smallest label, say node *y*, and make its label permanent. The node *y* is the second smaller label node from *s*. Thus, at each iteration, we reduce the values of temporary labels whenever possible

(by selecting the most recent permanently labelled node), then select the node with smallest temporary label and make it permanent. We continue in this fashion until the target node d gets permanently labelled. In order to distinguish the permanently labelled nodes from the temporarily labelled ones. We will keep a Boolean array ***State*** of order n. When the *i*th node becomes permanently labelled, the *i*th element of this array changed from false to true. Another array, D, of order n will be used to store labels of nodes. A variable *recent* will be used to keep track of most recent node to be permanently labelled. In Figure 4.8, a pseudo-code of the AMI algorithm to find the minimal cost increment path to place the demand left unassigned is presented:

Variables used in the algorithm are given below:

- A Binary adjacency matrix storing the network topology where if the span exists between node pair (*i, j*), set the entry $a_{i,j}$ of the matrix A to 1, otherwise, 0.
- C Capacity matrix storing the number of working capacity, where the entry $c_{i,j}$ of matrix C is the number of working capacity on the span (*i, j*).
- *s* Source node.
- *d* Destination node.
- ***State*** Array of the type Boolean, when node *i* is to be permanently labelled, the entry ***State***[*i*] of the ***State*** is true, otherwise, false.
- *Recent* The most recent node permanently labelled
- P Predecessor array, where the elements P[*i*]of array P store the predecessor node ID of node *i*, for example, P[*Recent*] is predecessor of node *Recent.*
- INF Upper bound of the amount of capacities carried on an optic fibre.
- D[*i*] Minimal sum of the cost increment of spans resulting from placing the specified amount of capacity on the spans from *i* to source *s*, measured so far.
- F(C, *i*) Optic fibre span stepwise cost function with capacity C carried on the span *i*. For simplicity, we assume the span cost function of all spans in the network is the same, so the function F(C, *i*) can be simplified to the function F(C). It is worth noting that the assumption of the same span cost function is not necessary for the AMI algorithm.
- E Array to store all nodes to have been permanently labeled measured so far.

Initialization:

While (for all nodes in a given network)

{

State[*i*] = false

P[*i*] = -1

D[*i*] = ∞

}

$a_{s,d} = 0$

Recent = *s*

E = {*s*}

Procedure:

State[*s*] = true

D[*s*] = 0

While (if there still are nodes that do not belong to E or *Recent* node is disconnected from the other part of the network or ***State***[*d*] != true)

{

While (for all nodes *i* ,the immediate successors of the node *Recent*, which have been not labelled permanently and update D[*i*])

{

$TempC = c_{s,d} + c_{Recent,i}$

$Cost = F(TempC) - F(c_{Recent,i})$

if (Cost < D[*i*])

{

D[*i*] =Cost

P[*i*] = *Recent*

}

}

For (All immediate successors *i* of the node *Recent*)

{

Find the node with the smallest value among D[*i*], say *y*

State[*y*] = true

Recent = *y*

E = {*Recent*}

}

}

Figure 4.8 Addition Minimal Increment Algorithm

Note that the array P[] keep track of the immediate predecessor of a node in the AMI path from node s to d. At the end of the execution, the AMI path cost for the amount of the disrupted capacities, $c_{s,d}$, is given by D(*d*). The actual path can be obtained by tracing backward the predecessors of the nodes in the P array from node *d* to *s*. That is the sequence of nodes

$$s, P(P[...]), ..., P(P[d]), P[d], d,$$

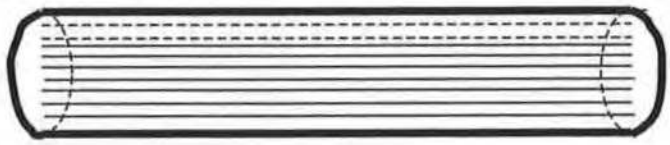
is an AMI path from *s* to *d*. In case there is no path from *s* to *d* in a given network, D[*d*] will remain ∞. This condition will occur if and only if at some point all the temporarily labeled nodes of the point have ∞ label. Upon detecting this condition, we must exit from the outer loop in the "while" loop step and stop.

4.5.3 Stepwise Cost Heuristic (SCH) algorithm on SCP Problems with Stepwise Link Cost Function

The SCH is developed based on the FPAs and the AMI algorithm. In networks there are always some spare capacities in the spans of the network resulting from stepwise cost function (explained below). The issue is how to utilize these spare capacities? The SCH algorithm uses a maximum flow algorithm, the Ford-Fulkerson's algorithm, to find the maximal amount of feasible spare paths consisting of these spare capacities to re-route the disrupted traffic at the event of network failure. The AMI algorithm is then used to determine the AMI path for the disrupted traffic left that is not rerouted to minimize the network design cost.

- **Spare Capacity resulting from the Stepwise Cost Function**

In a given network G(V, E) where V is denoted a set of nodes and E a set of spans of the network G, we define two matrices C and R storing the amount of working channels and the maximal capacity on each optic-fiber cable respectively. Since the link cost function is stepwise in capacity, the entry $R_{i,j}$ of matrix R, storing the number of channels on span (*i, j*), is always less than or equal to the entry $C_{i,j}$ of matrix C, storing the maximal capacity on span (i, j), as explained in Figure 4.9. In Figure 4.9, there is a STS-9 span where 7 STS-1s are the working channels and 2 extra STS-1 channels.



—— Working Channels ------ Spare Channels

Figure 4.9 Structure of Spans

In practice, 7 STS-1 capacities are active to transform the network services, but we have to install 9 STS-1 (i.e. STS-9) span for this purpose because STS-9 spans are only available in markets. Therefore, two STS-2 channels in the STS-9 are installed but idle. It is obvious that using these extra channels would not result in any more cost of networks.

In case no SCP algorithm is applied to place spare capacity in the network, 2 extra channels can still be used to reach some level of network survivability because the failed traffic at the event of network failure can be re-routed through these extra channels. The SCH algorithm can effectively take the advantage of these extra channels to restore the part of the disrupted traffic. Since the granularity of capacity of network links, as described in Figure 4.9, results in these extra channels in a network, using these extra channels in this manner will not incur any extra network design cost. The SCH algorithm can be implemented as follows. The algorithm only considers single span failure;

- **Description of the Stepwise Cost Heuristic (SCH) algorithm**

The algorithm only considers single span failure – the algorithm generates a network design which is resilient to a single span failure; the network design will not be resilient to multiple span failures. First, to avoid reaching local optimal solution to the SCP problem, we take a span in turn as failed in a random order (take one at a time). Second, determine all demand pairs affected by the span failure and release all affected working paths through the failed span which can be put into the pool of extra channels and make those available to the restoration process. Third, the Ford-Fulkerson's algorithm is applied to obtain the maximum number of extra paths, consisting of extra channels, between each pair of affected demand in turn. In case the number of extra paths is greater than or equal to the affected paths, only the actual number of the extra paths is

labeled temporarily as the occupied paths that will no longer be available for the other demand pairs affected by the same span failure. In case the number of the extra paths obtained by the Ford-Fulkerson's algorithm is less than the affected paths, all extra paths found is labeled temporarily as the occupied paths. Fourth, the AMI algorithm will be employed to find an AMI path to place the same the number of spare paths as the affected paths left unsaved in order to recover them with the minimal cost increment. And then the newly placed spare capacities are labeled temporarily as the occupied paths and update the extra channels available by the stepwise link cost function. When all affected demand pairs by the span failure is examined all temporarily occupied paths will be converted to the extra paths available for other span failure in the network; Repeat the above procedure in the fashion until all spans in the network are examined. To make the above description clearer, a pseudo-code of the SCH algorithm is given below.

Given:

- A Network binary adjacent matrix.
- C Network capacity matrix for link capacity
- R Network capacity matrix for actual working capacity
- LC Number of working paths left un-restored after using the Ford-Fulkerson's algorithm
- TO A matrix, where its entry TO_{ij} stories the number of extra channels available so far on the span (i, j).

Procedure:

While(All spans in a network)

{

(1) Randomly choose one span failed which starts from node i and terminates at node j, and then remove it by setting a_{ij} to zero.

(2) Determine all demand pairs affected by the single span failure and the corresponding amount of working paths failed.

(3) Release the network capacities occupied by the working paths failed at the event of the span failure and makes those available to following restoration process.

While (All demand pairs affected)

{

(a) Using the Ford-Fulkerson's algorithm to find the maximum number of the extra paths between demand pair (*i, j*).

(b) If the number of extra paths is greater than or equal to the working paths failed between the demand pair (*i, j*) and then update the matrix TO by subtracting the actually used extra channels, making up the extra paths found, from the spans which these extra paths pass through. And then go to the end of the "while" loop, otherwise go to step (c).

(c) Use the AMI algorithm to find the AMI path on which to place the failed working paths left unsaved (i.e. LC), and update the matrix C and TO by using the stepwise link cost function.

}

(4) Update the matrix TO by subtracting matrix R by matrix C.

}

Figure 4.10 Pseudo-code of the Stepwise Cost Heuristic (SCH) algorithm

The version of the SCH algorithm described above is of path restoration, but it can also work for span restoration with a small modification. For span restoration there is only one demand pair affected when a span fails in a network, i.e. the demand on the failed span, so the step (2) of the first "while" loop in Figure 4.11 can is simplified as follows:

(2") Determine the amount of demand in the failed span.

So, it can be seen that the version of the SCH algorithm for span restoration is a special case for path restoration.

Like IP based SCP algorithm the SCH algorithm also takes stub release into account in order to minimize the amount of spare capacity required.

Chapter 5 Results and Comparison

5.1 Introduction

In chapter 4, three SCP algorithms, i.e. the IP based SCP algorithm, the max-latching algorithm and the SCH algorithm, have been illustrated. In this section the results with regard to execution time and network cost from the three SCP algorithms will be presented

Here, the SCH algorithm and the IP based SCP algorithm can be applied to both the path restoration design and span restoration design. But they are only implemented for span restoration in this chapter because they will be compared with the max-latch algorithm, an only span restoration algorithm.

We will apply the above three SCP algorithms over 10 test networks described in section 5.2 with 4 different scenarios of link cost function introduced in section 5.3. The results, presented in section 5.4, help us to figure out the performances of these three SCP algorithms in terms of execution time and network cost required by a single network span failure scenario, and effect of the different link cost scenarios on the network cost obtained by three SCP algorithms.

5.2 Test Networks

Ten networks are used as the test networks in this chapter. Three of them, i.e. the test networks 1, 7 and 2, with their associated demand matrices, detailed in **Error! Reference source not found.**, **Error! Reference source not found.** and **Error! Reference source not found.** of the Appendix A, have been used in Chapter 3. Figure 5.1 shows the architecture of the test network 6 that many literatures [12][14][15][20] adopted to test their SCP algorithms, where the numbers next to the network links represent their working capacities. The rest of test networks are generated in random, the number of nodes of these test networks ranges from 8 to 40. Table 5.1 shows the detailed information of these test networks,

	No. of Nodes	Nodes of Spans	Network Working Capacity
Network 1	8	16	136
Network 2	9	17	153
Network 3	9	19	813
Network 4	10	21	920

Network 5	**11**	**23**	**1252**
Network 6	**16**	**28**	**397**
Network 7	**20**	**44**	**2158**
Network 8	**19**	**52**	**2078**
Network 9	**31**	**97**	**4324**
Network 10	**39**	**128**	**6210**

Table 5.1 The Information of ten Test Networks

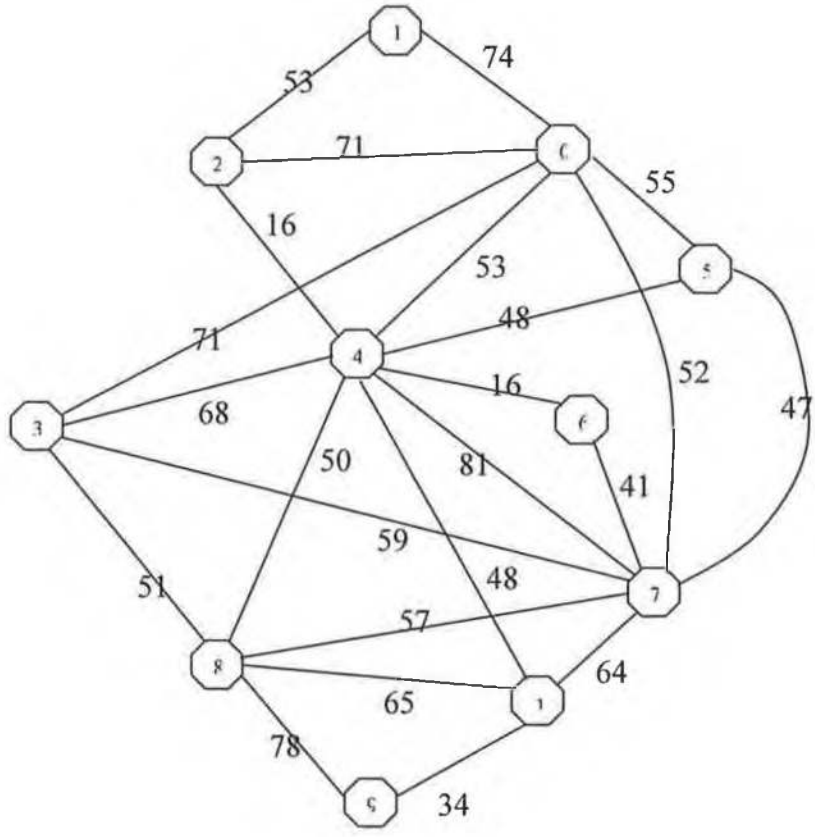
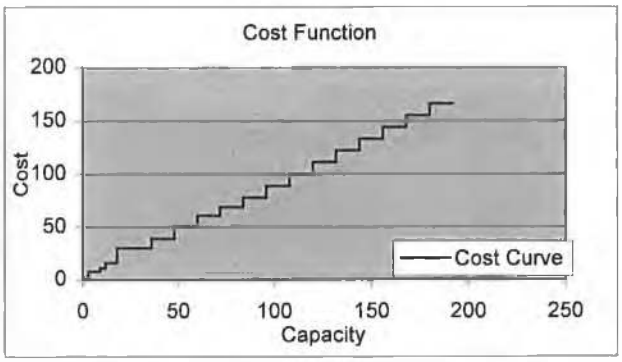


Figure 5.1 Test Network 6 Architecture

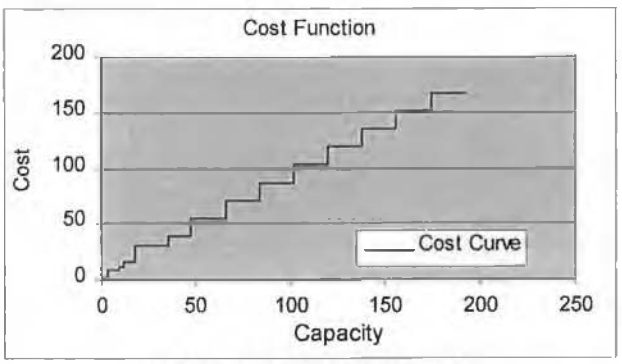
5.3 Stepwise Link Cost Function Scenarios

The chapter 4 introduced five types of link cost function, i.e. linear cost function, linear cost with fixed cost, concave link cost function, concave link cost function with fixed cost, stepwise link cost function. As described above, optic fibers carrying the specific amount of capacity are only available in modern markets, so stepwise link cost function most reflects the reality. Hence, we have designed four link cost scenarios where the link cost function is stepwise in capacity described in Figure 5.2 and Table 5.2. The four stepwise links cost functions are based on the SONET network signal hierarchy, for example, STS-1 and STS-3, as described in Chapter 2. When the capacity required is bigger than the maximum capacity that can be carried by a single link available in the

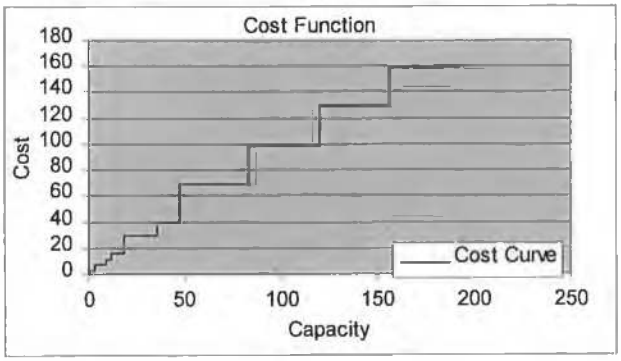
markets. To solve this problem, several optic fibres are placed in parallel. In our case, the maximum capacity in a single optic fibre is STS-48. In this project, we bind several STS-12, STT-18, STS-24 and STS-48 optic fibres with a STS-48 link together to obtain more than STS-48 capacity span respectively as described in Figure 5.2 (a) through (d) and Table 5.2.



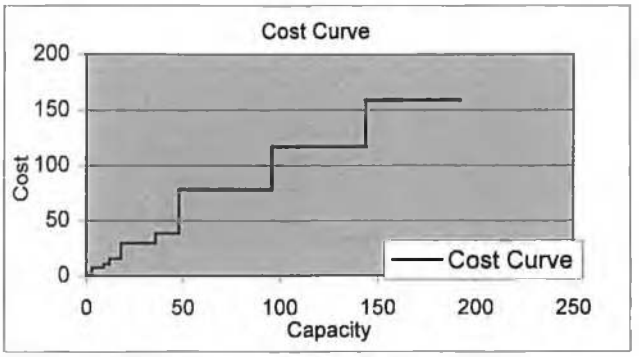
(a) Link Cost Function 1



(b) Link Cost Function 2



(c) Link Cost Function 3



(d) Link Cost Function 4

Figure 5.2 Four Link Cost Function Scenarios

Cost Function 1		Cost Function 2		Cost Function 3		Cost Function4	
Capacity (STS-N)	Cost	Capacity (STS-N)	Cost	Capacity (STS-N)	Cost	Capacity (STS-N)	Cost
1	1	1	1	1	1	1	1
3	2	3	2	3	2	3	2
9	8	9	8	9	8	9	8
12	11	12	11	12	11	12	11
18	16	18	16	18	16	18	16
36	30	36	30	36	30	36	30
48	39	48	39	48	39	48	39
60	50	66	55	84	75	96	78
72	61	84	85	120	105	144	117
84	69	102	101	156	135	192	159
96	78	120	117	192	165		
108	89	138	133				
120	100	156	149				
132	111	174	165				
144	122	192	181				
156	133						
168	144						
180	155						
192	166						

Table 5.2 Four Scenarios of Link Cost Function

The link cost function 1 in Figure 5.2(a) has the smallest capacity "jump" among four link cost function scenarios whereas the link cost function 4 in Figure 5.2(d) has the largest capacity "jump". In other words, the link cost function 1 is the closest to linear compared with the others, so the IP based SCP algorithm is expected to get the closest solution to the optimal one to SCP problems when the link cost function 1 is employed.

In section 5.4.1, we will conclude the performance of three SCP algorithms with four link cost functions scenarios.

5.4 Spare Capacity Placement Test Results

It is important to be clear on what we are looking for from our results. The following two criteria are considered most important in evaluating the SCP algorithms.

(i) Cost; our objective function is a representation of monetary cost and this must be minimized.

(ii) Speed; as computation time is limited, the length of time is very important for the application of the SCP algorithms over large-scale networks.

In the following section, three SCP algorithms will be applied over ten test networks, described above, to obtain their execution time and total cost while the survivability is set to 100 %. The effect of different link cost functions on the total network cost for each SCP algorithm is also investigated. To ensure the comparison is valid, all SCP algorithms are performed on the same machine, i.e. a computer with Pentium processor operating 180Mhz.

5.4.1 Comparison of time complexities among three SCP algorithms

In this section, we are interested in seeing the performance of three SCP algorithms in terms of execution time. Table 5.3 gives the results. Note that the sum of the numbers of network nodes and spans represents the size of the test networks.

Test Networks	Network Size (Nodes + Spans)	Execution Time (sec)		
		SCH Algorithm	Max-latching Algorithm	IP based SCP algorithm
Network 1	24	0.25	0.4	358
Network 2	26	0.2	0.525	684
Network 3	28	0.2	0.55	1028
Network 4	31	0.2	0.625	2301
Network 5	34	0.2	0.525	4543
Network 6	44	0.25	0.775	uncompleted
Network 7	64	0.3	1.325	uncompleted
Network 8	71	0.2	2.05	uncompleted
Network 9	128	0.6	5.725	uncompleted
Network 10	205	0.8	7.575	uncompleted

Table 5.3 Execution time of three SCP algorithms over ten test networks

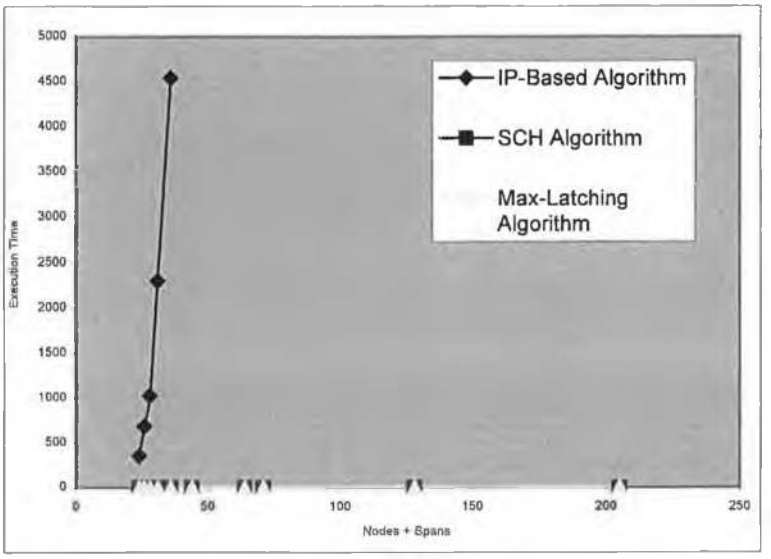
Observations:

(i) The SCH algorithm is by far the fastest in all cases, and its execution time increases very slowly while the network size increases.

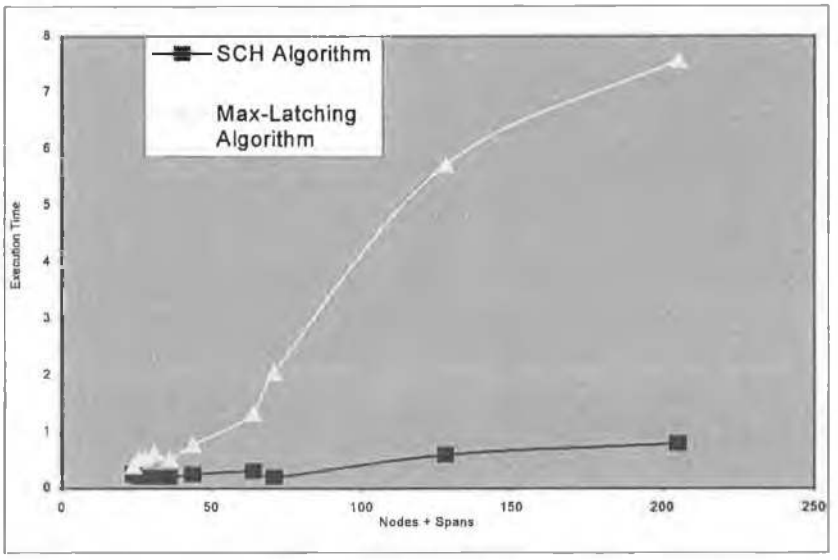
(ii) The IP based SCP algorithm has the longest execution time among the three SCP algorithms, and the time taken is increasing exponentially when the network size increases, even worse, the algorithm can not be completed when the network size is over 44.

Figure 5.3 displays a comparison of three SCP algorithms with respect to their execution time. In Figure 5.3 (a) displays a comparison of how much time is taken by the SCP algorithm with those by the IP based SCP algorithm and the max-latching algorithm over ten test networks. In Figure 5.3 (b), a comparison of execution times taken by the SCH algorithm and the max-latching algorithm is given.

The execution time, taken by the IP based algorithm, is the longest compared with those taken by the max-latching algorithm and the SCH algorithm due to its highest time complexity. The time complexity of this algorithm is of the order $O\ (2^{s*s*d} -1)$ [8], where s*s*d, s and d are the number of the variables in the SCP formulation, given in Figure 4.2, the number of spans and the network connectivity respectively, and the execution time taken by the IP based SCP algorithm increases dramatically when the network size increases. So the IP based algorithm is not suitable to deal with the large-scale network, whereas the SCH algorithm and the max-latching algorithm can work effectively for this case.



(a) Execution Time of three SCP algorithms



(b) Execution Time of SCH Algorithm and Max-Latching algorithm

Figure 5.3 Execution Time of three SCP Algorithms over ten Test Networks

5.4.2 Performance of three SCP Algorithms in terms of total Network Cost Savings

In this section, we wish to examine the performance of three SCP algorithms in terms of network cost. Here, we consider the results from the SCH algorithm as the reference, its relative network cost savings (percentage) with the other two SCP algorithms can be derived from the following equation.

$$\text{Network cost savings} = ((C_{ip}(orC_h) - C_r)/C_r),$$

Where C_r, C_{ip} and C_h represent the network costs from the SCH algorithm, the IP based SCP algorithm and the max-latching algorithm respectively.

As mentioned in Chapter 4, the IP based SCP algorithm and the max-latching algorithm can only work on the SCP problem with the linear cost function scenario. To make it valid comparing the network cost C_r from the SCH algorithm with the network costs, i.e. C_{ip} and C_h from the other two SCP algorithms, the C_{ip} and C_h must be recalculated based on the stepwise link cost function scenario. The following reasons make the recalculation reasonable:

1) All three SCP algorithms are applied to determine the number of spare capacity and their layout required for the same level of network survivability, say, 100% network survivability.

2) The results from the IP based algorithm and the max-latching algorithm are independent from the linear cost C_j per span channel, given in Figure 4.2 and section 4.4.1.

Table 5.4 displays the following results:

1) Costs for 100% network survivability from the three SCP algorithms over ten test networks. The cost for each SCP algorithm is the average cost over four stepwise link cost function scenarios.

2) The relative cost saving (percentage) of the SCH algorithm compared with those of the other two SCP algorithms.

	Cost from My Algorithm	**Cost from max-latching Algorithm**	**Relative Cost Savings (percentage)** $(C_h)-C_r)/C_r$	**Cost from IP based algorithm**	**Relative Cost Savings (percentage)** $(C_{ip}-C_r)/C_r$
Network 1	60	93	55%	64	6.7%
Network 2	65	131	102%	102	56.9%
Network 3	461	588	28%	447	-3.03%
Network 4	427	635	48.7%	431	0.94%
Network 5	599	707	18%	578	-3.5%
Network 6	148	421	178.4%	N/A	N/A
Network 7	796	1364	71.4%	N/A	N/A
Network 8	704	1080	53.5%	N/A	N/A
Network 9	1180	2224	100%	N/A	N/A
Network 10	1675	3100	85%	N/A	N/A

Table 5.4 The Performance of three SCP algorithms in terms of Network Cost Savings

Observation:

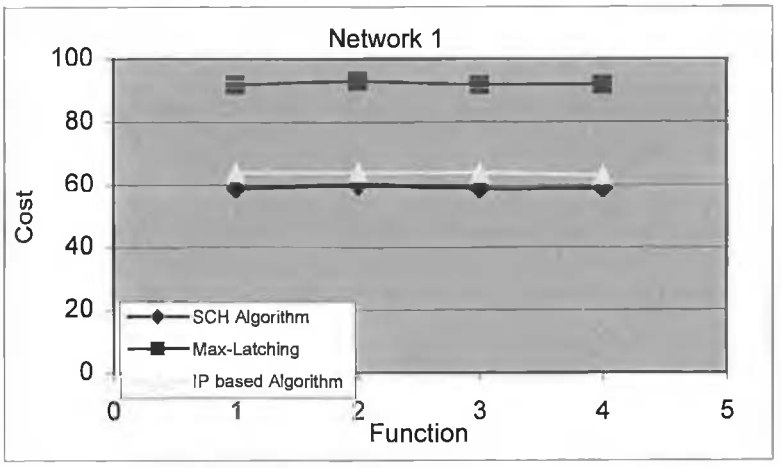
(1) The SCH algorithm is the most effective approach to solve the SCP problems in terms of total network cost, where the link cost function is stepwise in capacity, in most cases. Compared with the max-latching algorithm, its relative cost savings ranges from 18% to 178% (80% on average). Compared with the IP based SCP algorithm, its relative cost savings of the SCH algorithm ranges from –3.5% to 56.9% (11.6% on average).

(2) No results from the IP based SCP algorithm are available when the size of the test networks is over 44 due to the high time complexity of the algorithm (2^N -1), where N is the number of the variables of the formulation given in Figure 4.2.

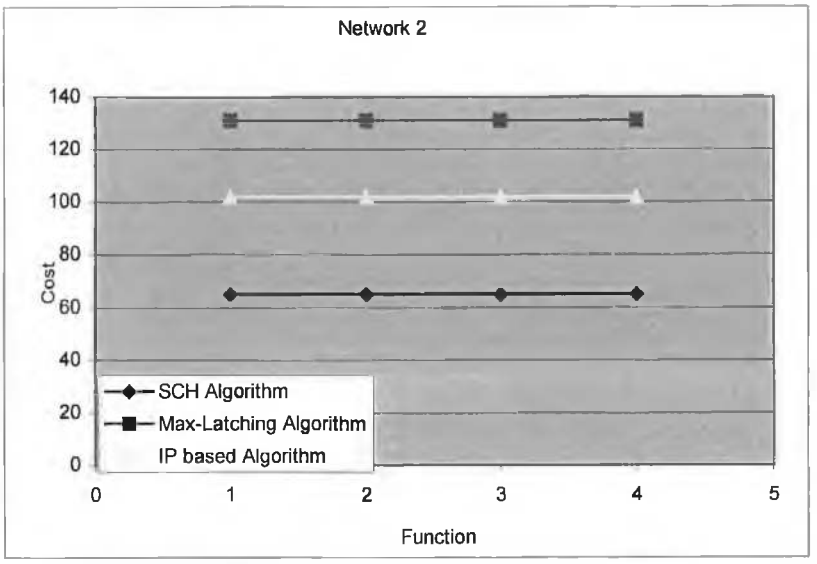
From the above results, we could see that the SCH algorithm produces the best solution in most cases, as we have expected. The SCH algorithm is a local search algorithm, and maybe reach hence a local optimal result, but it is designed to utilize of the spare channels resulting from the step cost function which the other two SCP algorithms do not, which covers most of its disadvantage, a local search. Therefore, the SCH algorithm, in general, gets the better solution than the IP based SCP algorithm, a global search algorithm. The reason why the IP algorithm gets the better result than the SCH algorithm in some cases will be explained in the following section.

5.4.3 Effect of the Stepwise Cost Function Scenarios on the Network Cost

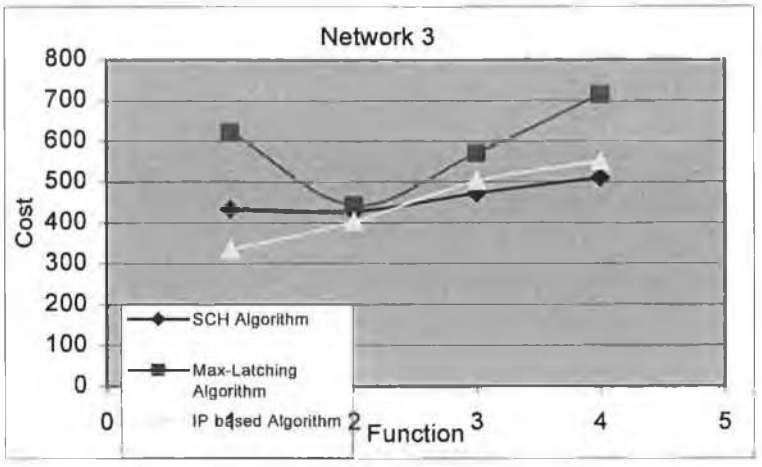
In this section, we look at the effect of the different stepwise cost function scenarios on the performance of three SCP algorithms in terms of the network cost and see how much benefit the SCH algorithm can obtain from the four stepwise cost function scenarios. Figure 5.4 (a) through (j) shows the network costs from three SCP algorithms with four stepwise cost function scenarios over ten test networks.



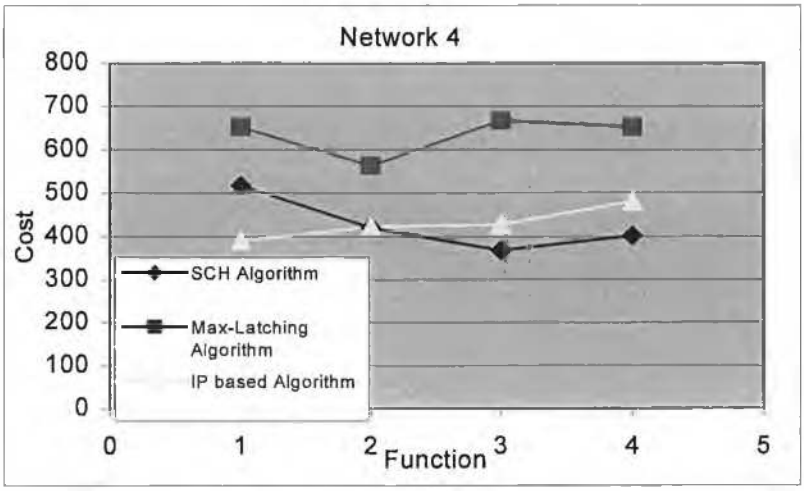
(a) Cost for Network 1



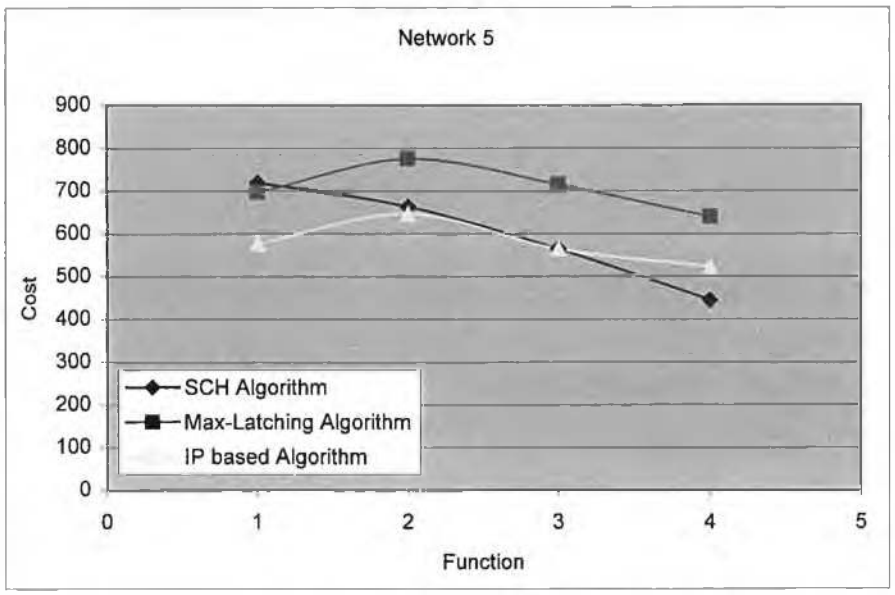
(b) Cost for Network 2



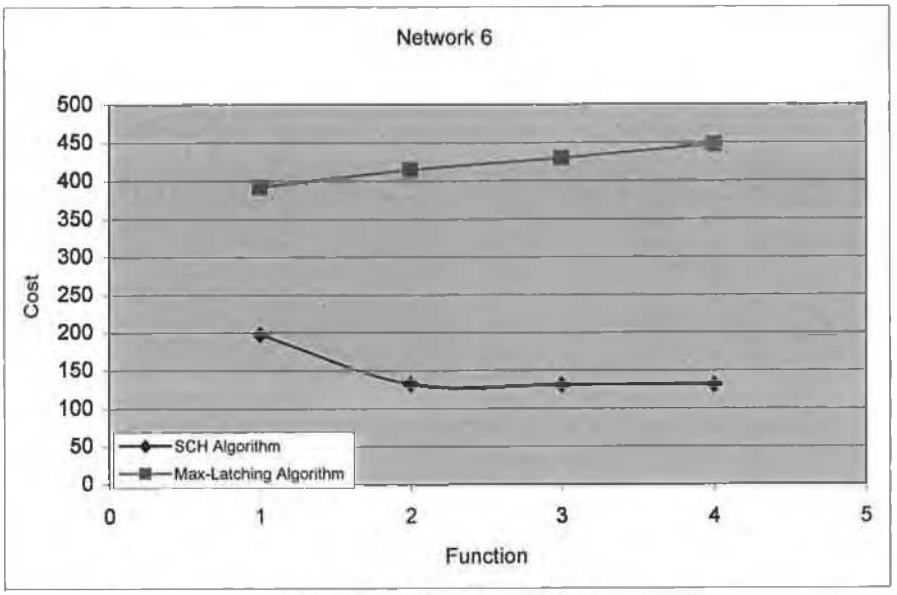
(c) Cost for Network 3



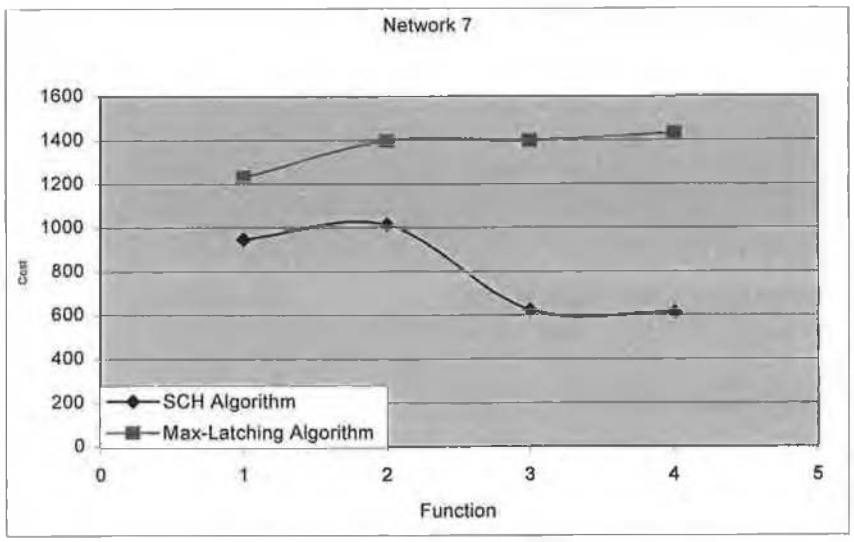
(d) Cost for Network 4



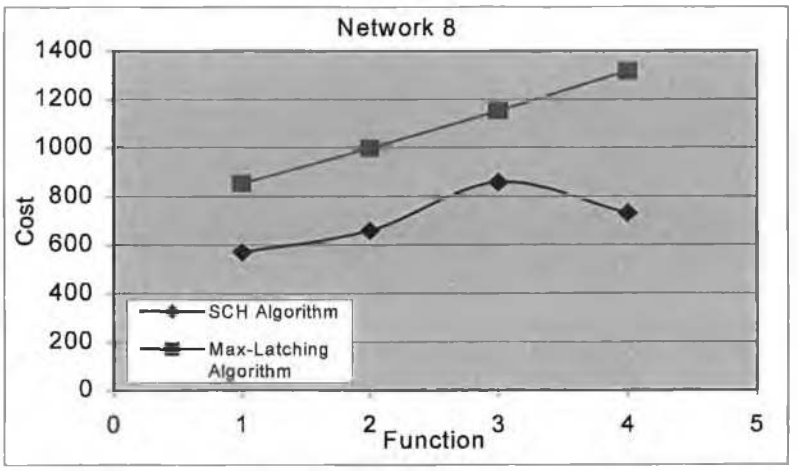
(e) Cost for Network 5



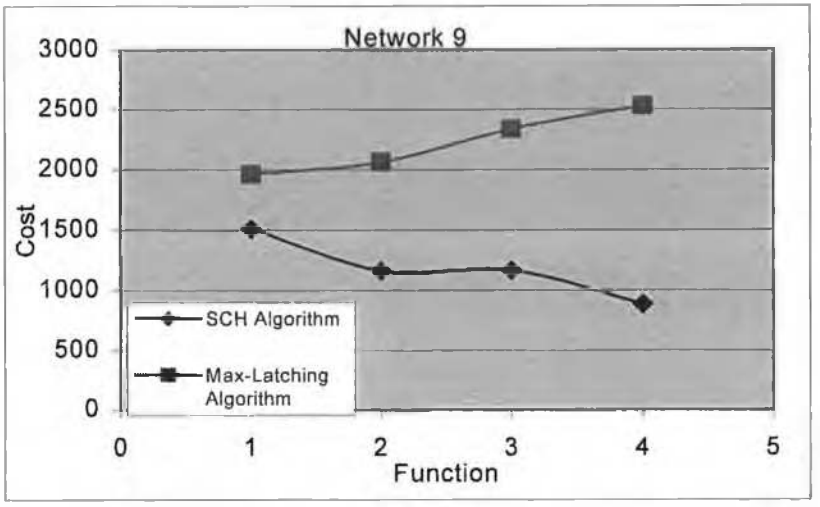
(e) Cost for Network 6



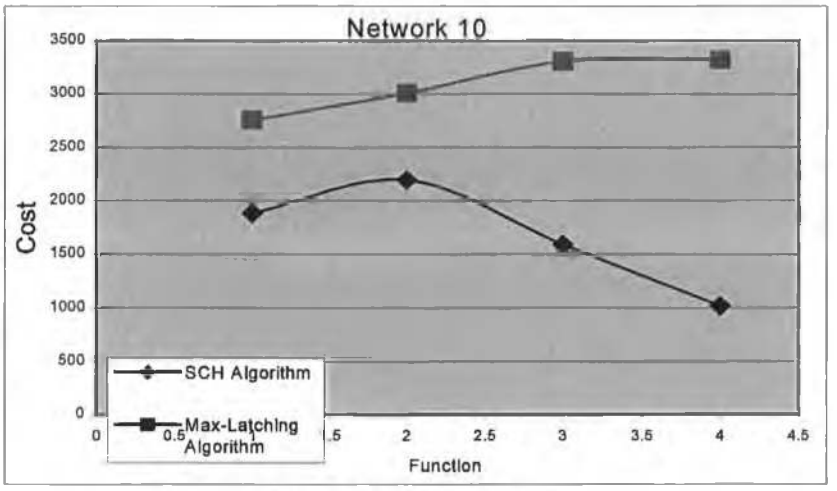
(f) Cost for Network 7



(g) Cost for Network 8



(h) Cost for Network 9



(j) Cost for Network 10

Figure 5.4 Network Cost with Four Link Cost Scenarios

It is worth noting that the four stepwise link cost scenarios is sorted in the order of increasing in terms of capacity "gap", for example, the function 1 has STS-12 capacity gap whereas the function 4 has the STS-48 capacity. Hence, the function 1 is the closest to linear among the other cost function scenarios.

Observation:

(1) The SCH algorithm always obtains the minimal network cost compared with the other two SCP algorithms when the stepwise cost function 3 and 4 are applied.

(2) Compared with the max-latching algorithm, the SCH algorithm always obtains the better results for four stepwise link cost function scenarios.

(3) Compared with the IP based SCP algorithm the IP based SCP algorithm obtains the better results than the SCH algorithm does in the test network 3, 4 and 5 when the cost function 1 and 2 are applied. However, the cost savings (percentage) of the SCH algorithm over the IP based SCP algorithm for all cases still reach 11.6% on average given in section 5.4.2.

(4) With the capacity "gap" bigger the result from the SCH algorithm become better relatively compared with those from the other two SCP algorithms, in other words, the SCH algorithm can get most from the stepwise link cost function scenarios.

From the results, we see the SCH algorithm get the most benefit, as we have expected, while the capacity "gap" of the stepwise link cost function increases from the function 1

to the function 4. As a whole, the SCH algorithm obtains the better results than that from the IP based SCP algorithm does because the SCH algorithm is designed to utilize spare channels on each network span resulting from the stepwise link cost function, which the IP based SCP algorithm is not. However, the disadvantage of the SCH algorithm is that it is a local optimal (search) algorithm whereas the IP based SCP algorithm is a global one, as a result, the results from the IP based SCP algorithm are better than those are from the SCH algorithm in some cases. When the stepwise link cost function is close to linear, the spare channels of network spans resulting from it get little, so the SCH algorithm loses its advantage of using the spare channels so that the results from it become worse than that from the IP based algorithm.

Chapter 6 Conclusion

This chapter is concerned with summarizing the observations that arise from this research work. Firstly, section 6.1 is concerned with a discussion of experimental results. Section 6.2 discuses possible directions for applications and further research. Finally, section 6.3 sums up the principal conclusion.

6.1 Overall Discussion of Results

In Chapter 3, four Feasible Path Algorithms (FPAs) algorithms have been illustrated, i.e. the Ford-Fulkerson's algorithm, the disjoint KSP algorithm, the non-disjoint KSP algorithm and the Matrix Maximum Flow (MMF) algorithm. The Ford-Fulkerson's algorithm is the best in terms of total feasible paths, however, its major disadvantage is that it can not control the hops of the feasible paths. As to the non-disjoint KSP algorithm, it can find up to 98% of the feasible paths that the Ford-Fulkerson's algorithm does, and easily control the length of the feasible paths, so the KSP algorithm is a good alternative to the Ford-Fulkerson's algorithm in many cases. The MMF algorithm is a newly proposed FPAs. It has the following advantage: 1) Obtain the same amount of the feasible paths as the Ford-Fulkerson's algorithm does when hop-limit of feasible paths is large enough. 2) Be of control of hop-limits as the KSP algorithm is, and moreover it is believed to take less execution time than the Ford-Fulkerson's algorithm does in a vector-equipped computer as discussed in Chapter 3.

Restorability techniques and survivability techniques have different requirement for the FPAs, i.e. restorability techniques are more sensitive to its execution time to meet the need of real time services whereas survivability techniques is more sensitive to network design cost at design time.

Two types of the KSP algorithms have been implemented to find feasible paths in this project: the disjoint KSP algorithm and the non-disjoint KSP algorithm. The disjoint KSP algorithm is faster than the non-disjoint KSP algorithm, but, find the less account of feasible paths, so the disjoint KSP algorithm is a better option for restorability technique than the non-disjoint KSP algorithm, but the non-disjoint KSP algorithm is more suitable for survivability techniques than its counterpart.

Our primary concerns in evaluating the three SCP algorithms are network cost and execution time. In Chapter 4, three Spare Capacity Placement (SCP) algorithms have

been introduced, i.e. the IP based SCP algorithm, the max-latching algorithm and the SCH algorithm.

The best solution in terms of execution time is consistently the SCH algorithm. The execution time of the IP based SCP algorithm is longest among the three SCP algorithms and increases dramatically when the size of networks increases due to its higher time complexity $O(2^{s*s*d})$, where s and d are the number of network spans and connectivity, so the IP based SCP algorithm is not suitable to design large-scale survivable networks. The max-latching algorithm and the SCH algorithm are the local search algorithms, therefore, they have much less time complexity than the IP based SCP algorithm. Both the SCH algorithm and max-latching algorithm can be applied to large-scale networks, for example, transport networks.

Our second evaluation criteria is network cost. Here, the SCH algorithm obtains the best results in most cases. Compared with the max-latching algorithm, the results from the SCH algorithm is always better, as we have expected. In principle, The SCH algorithm should always have obtained the better results than the IP based SCP algorithm does because the SCH algorithm can take use of the spare channels resulting from the stepwise link cost function, which the IP based SCP algorithm can not. However, the IP based algorithm is a global search algorithm whereas the SCH algorithm is a local search algorithm; if the stepwise cost function is close to linear (e.g. the stepwise link cost function 1 in Figure 5.2) the IP based SCP algorithm may get a better solution than the SCH algorithm does. When the stepwise link cost function is close to linear, the spare channels resulting from it get little, so the SCH algorithm will lose its advantage of using spare channels so that the results from it become worse than that from the IP based algorithm. In this project, no exact linear cost function was chosen because it does not exist.

In summary, the SCH algorithm is the best choice to design the survivable network, especially, large-scale transport network due to its excellent performance in terms of execution time and network cost. The IP based SCP algorithm is a better choice for the small-scale network design in case the stepwise link cost function is close to linear. We do not recommend use of the max-latching algorithm in survivable network design due to its bad performance in network cost.

6.2 Application and Further Work

In this section, we discuss applications arising from this project and potential for further development.

6.2.1 Application to Survivable Network Design

The SCH algorithm is designed to solve Spare Capacity Placement (SCP) problems in mesh-type survivable networks in case stepwise link cost functions are applied. The SCH algorithm can quickly determine the networks that have a capability to prevent them from a single span failure, so it can be applied for large–scale networks, e.g. transport networks.

6.2.2 User Interface Development

It is generally useful in computer-aides design to have some user interaction with the design process as, for many problems, humans have a superior ability to visualize what a good solution should look like.

6.2.3 Further Refinement of the SCH Algorithm

In this project, we implemented the SCH algorithm for span restoration and a single span failure, we can also refine the SCH algorithm to work for restoration path and other network failure i.e. multiple span failure and node failure. In addition, we can develop a dynamic scheme (i.e. restorability technique) corresponding to the SCH algorithm to deal with the unexpected network failure or traffic growth. Furthermore, we can apply the SCH algorithm in the VP level of ATM networks with some modifications as in [40].

We could say that this project is successful by looking at the result analysis, since the SCH algorithm is the fastest among the other two SCP algorithms, and obtain the best result in terms of network cost in most cases.

References

[1] Tsong-Ho Wu, *Fiber Network Service Survivability*, ARTECH HOUSE, INC. 1992.

[2] Hideki Sakauchi, Yasuyo Nishimura and Satoshi Hasegawa, "A Self-Healing Network With An Economical Spare-Channel Assignment", GLOBECOM 90 1990. pp 438-443, 1990.

[3] Hiroaki Komine, Takafumi Chujo, Takao Ogura, Keiji Miyazaki, and Tetsuo Soejima, "A Distribution Restoration Algorithm For Multiple-Link And Node Failures of Transport Networks", Proc. IEEE GLOBECOM 90, pp459-463, 1990.

[4] C. Han Yang and Satoshi Hasegawa, "Fitness: Failure Immunization Technology For Network Service Survivability", IEEE Globecom 88, 1988.

[5] W. D. Grover, B. D. Venables, M. H. MacGregor, "Performance studies of a self healing network protocol in Telecom Canada long haul networks", IEEE Global Conf. Commun., Dec. 1990. pp. 452-458, 1990.

[6] Martin De Prycker, Asynchronous Transfer Mode: Solution For Broadband ISDN, Ellis Horwood Limited 1993.

[7] Demetrios Stamatelakis, (M.Sc.), NSERC Scholar, "Theory and Algorithms for Preconfiguration of Spare Capacity in Mesh Restorable Networks," Department of Electrical & Computer Engineering, University of Alberta, Spring 1997.

[8] Maciej M. Syslo, Narsigh Deo and Janusz S. Kowalik, Discrete Optimization Algorithm with Pascal programs, Prentice-Hall, Inc., 1983.

[9] Mike Sexton, Andy Reid, Braodband Networking: ATM, SDH, and SONET, ARTECH HOUSE, 1997.

[10] E.Oki, N.Yamanaka, "A recursive matrix-calculation method for disjoint path search with hop link number constraints", EICE Transactions Communications, Vol. E78-B, no.5, pp769-774, May 1995.

[11] Peter V. O'Neil, Advanced Engineering Mathematics, Wadsworth, Inc. 1987.

[12] M.H Macgregor and W.D.Grover, "Optimized K-shortest-paths Algorithm for Facility Restoration", Software-Practise And Experience 1994 Vol. 24 PT 9. Pp823-834, 1994.

[13] M. Gondran, M. Minoux, Graphs and Algorithms, John Wiley & Sons Ltd. 1984.

[14] Meir Herzberg, "A decomposition Approach to Assign Spare Channels in Self-Healing Networks", GLOBECOM 93 1993. pp 1601-1605, 1993.

[15] Meir Herzberg and Stephen J. Bye, "An Optimal Spare-Capacity Assignment Model for Survivable Networks with Hop Limits", GLOBECOM 94 1994. pp1601-1606, 1994

[16] Meir Herberg, Stephen J. Bye, and Anthony Utano, "The Hop-Limit Approach for Spare-Capacity Assignment in Survivable Networks", IEEE/ACM Transactions on Networking. Vol. 3. NO. 6, December 1995.

[17] M. T. Busche, C. M. Lockhart, and C. Olszewki, "Dynamic K-Shortest Path (DKSP) Facility Restoration Algorithm", IEEE Globecom'94, 1994.

[18] W.D. Grover, V.Rawat and M.H. MacGregor, "Fast Heuristic Principle for spare capacity placement in mesh-restorable SONET/SDH", Electronics Letters 1997 Vol. 33 PT 3 pp195-196, 1997.

[19] W.D.Grover, T.D. Bilodeau and B.D.Venables, "Near Optimal Spare Capacity Planning In a Mesh Restorable Network", GLOBECOM 91 1991, pp 2007-2012, 1991.

[20] B.D.Venables, W.D.Grover, and M.H.MacGregor, "Two Strategies for Spare Capacity Placement in Mesh Restorable Networks", IEEE Int. Conf On Communications ICC93 Conf Record Vol. 1 Geneva May 1993.

[21] R.R.Iraschko, M.H.MacGregor and W.D.Grover, "Optimal Capacity Placement for Path Restoration in Mesh Survivable Networks", 1996 IEEE INT. CONF. On Communication, Dallas Texas, 1996.

[22] Kazutaka Murakami and Hyong S. Kim, "Joint Optimization of Capacity and Flow Assignment for Self-Healing ATM Networks", Proc. IEEE ICC 95 JUNE 1995.

[23] D.Antony Dun, Wayne D. Grover, and Mike H. MacGregor, "Comparison of k-Shortest Paths and Maximum Flow Routing for Network Facility Restoration", IEEE Journal on Selected Areas in Communications, Jan. 1994, vol. 12, no. 1, pp.88-99.

[24] Rainer R. Iraschko, NSERC Scholar, (Ph.D.), "Path Restorable Networks," Department of Electrical & Computer Engineering, University of Alberta, fall 1996.

[25] W.D.Grover and M.H.Macgregor, "Potential for Spare Capacity Preconnection to Reduce Crossconnection WorkLoads in Mesh-Restorable Network", Electronics Letters, vol. 30, No. 3, 3rd Feb. 1994.

[26] Jiro Yamada and Akiya Inoue, "Intelligent Path Assignment Control For Network Survivability And Fairness", IEEE ICC'91.

[27] Takafumi Chujo, Hiroaki Komine, Keiji Miyazaki, Takao Ogura and Tetsuo, "Distributed Self-Healing Network and Its Optimum Spare Capacity Assignment Algorithm", Electronics and Communications In Japan, Part 1, Vol. 74, No. 7 1991.

[28] Takafumi Chujo, Hiroaki Komine, Keiji Miyazaki, Takao Ogura and Tetsuo

Soejima, "The design And Simulation of An Intelligent Transport Network With Distributed Control ", Network Operation and Management Symp. SAN Diego, Feb 1990.

[29] Hideki Saauchi, Yasuyo Okanoue and Satoshi Hasagawa, "Spare Channel Design Schemes For Self-Healing Network", IEICE Transaction Communication 1992 Jul. Vol. E75-B PT 7.

[30] B.C.Hturton and M Bentall, "Benchmark Networks For Both Physical And Virtual Networks", IEE Electronics and Communications Fifteenth UK Tele-traffic Symposium, On Performance Engineering In Information Systems. 1998.

[31] James McGibney, (M.Sc.), Modern Global Optimization Heuristics in the Long Term Planning of Telecommunication Networks, School of Electronic Engineering, Dublin City University, 1995.

[32] M. Kerner, H. L. Lemberg and D. M. Simmons, "An Analysis of Alternative Archectures for the Interoffice Network", IEEE *J. Select. Areas Commun.*, vol. SAC-4, pp 1404-1413, December 1986.

[33] B. Gavish *et al*, "Fiberoptic Circuit Network Design Under Reliability Constraints", IEEE *J. Select. Areas Commun.,* vol. SAC-7, pp 1181-1187, October 1989.

[34] A. Kershenbaum, P. Kermani and G. A. Grover, "Mentor: An Algorithm for Mesh Network Topological Optimization and Routing", IEEE Transactions on Communications, Vol. 39, No. 4, April 1991.

[35] T. C. Hu, Integer Programming and Network Flows, Reading, MA: Addison-Wesley, 1969.

[36] Grover, W. D., "The selfhealing network: a fast distributed restoration technique for network using digital cross-connect machines," *Proc. IEEE GLOBECOM '87*, Dec. 1987, pp. 28.2.2-28.2.6.

[37] Chao, C. W., Dollard, P. M., Weythman, J. E, Nguyen, L. T., Eslambolchi, H., " FASTAR-a robust system for fast DS3 restoration," *Proc. IEEE GLOBECOM'91*, Dec. 1991, pp.39.1.1-39.1.5.

[38] Gerard Sierksma, Linear And Integer Programming: Theory and Practice, Marcel Dekker, Inc., 1996

[39] S.D Nikolopoulos, A.Pitsillides, "Towards Network Survivability by Finding the K-best paths through a Trellis Graph", International Conference on Telecommunications (ICT'96), Istanbul, Turkiye, pp817- 821, April 1996.

[40] R. R. Iraschko, M. H. MacGregor and W. D. Grover, "Optimal Capacity Placement for Path Restoration in STM of ATM-Mesh Survivable Networks," IEEE/ACM Transactions on Networking, vol. 6, no. 3, June 1998.

Appendix A Test Networks for the Feasible Path Algorithm

In this section, five test networks used in chapter 3 and chapter 4 are depicted below.

In these diagrams, each line represents a network link, and the two numbers next to each link represents that link capacity and the length of the span; the length of the span is in brackets. Note that the length of network links is given by our measurement in these diagrams.

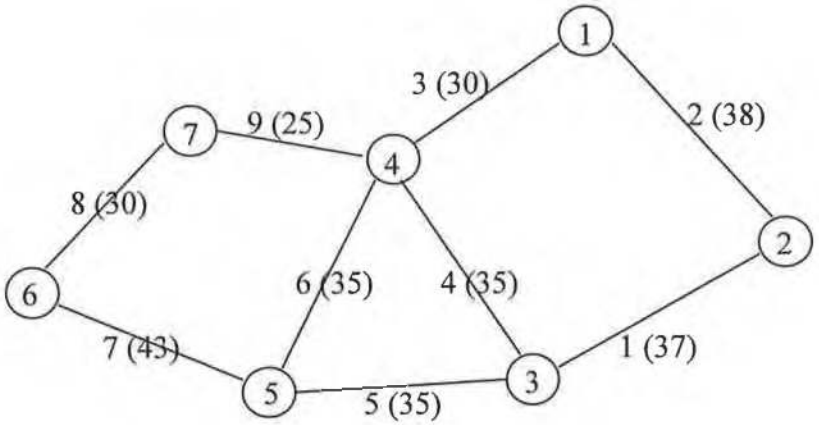


Figure A. 1 Topology of Test Network 1

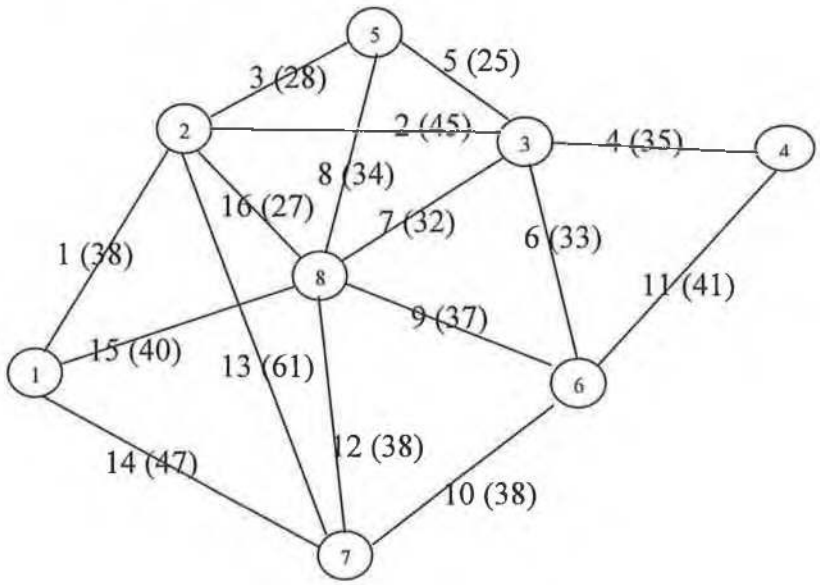


Figure A. 2 Topology of Test Network 2

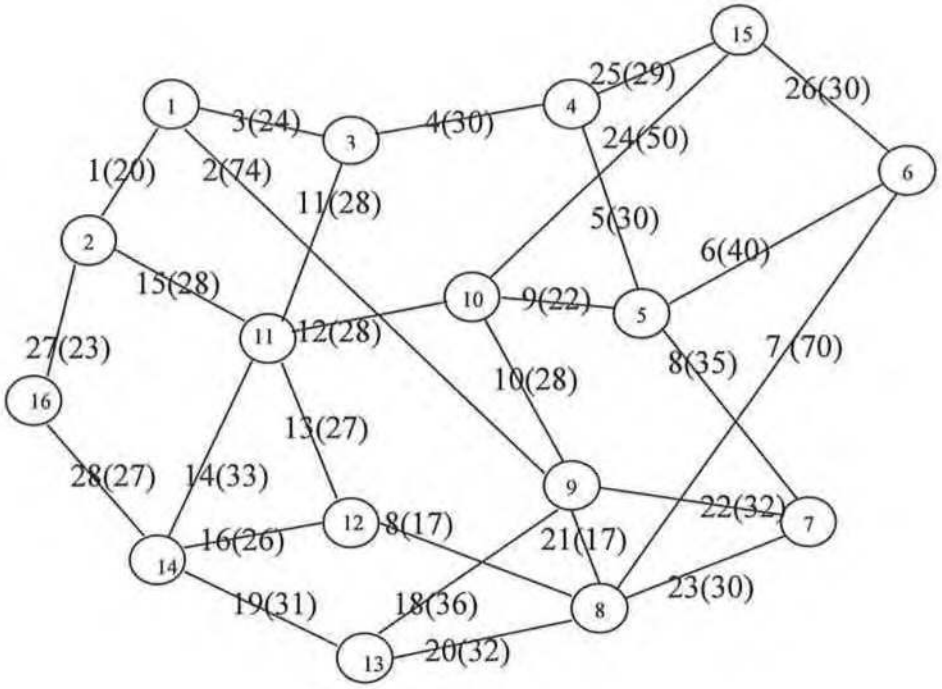


Figure A. 1 Topology of Test Network 3

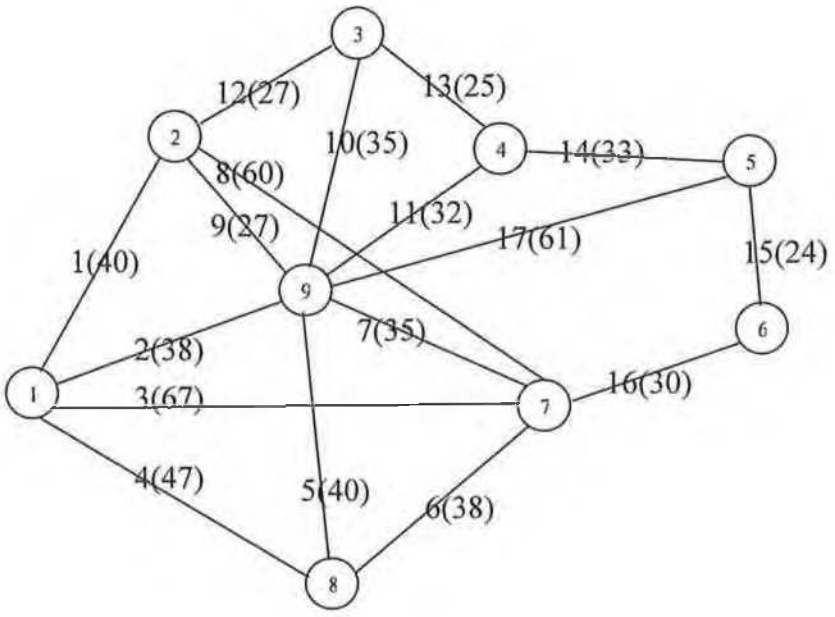


Figure A. 2 Topology of Test Network 4

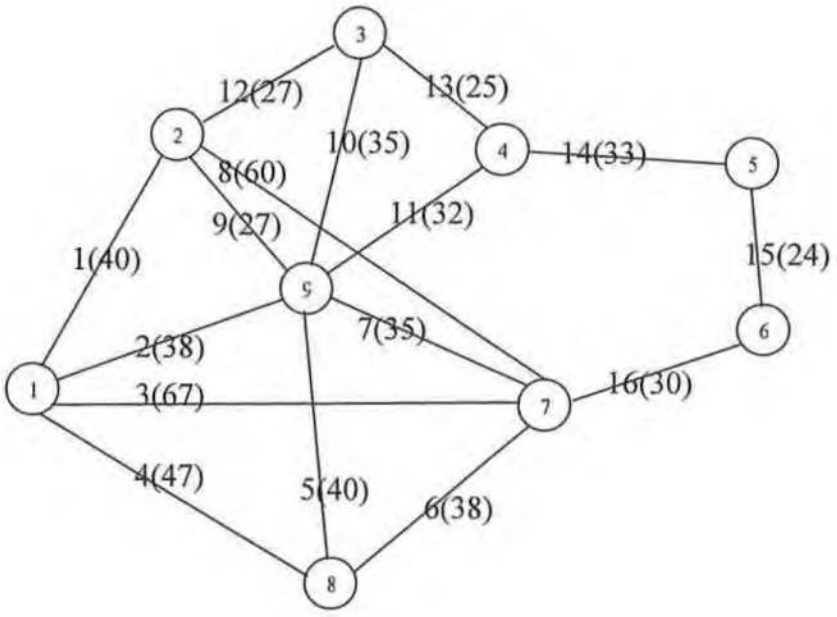


Figure A. 1 Topology of Test Network 5