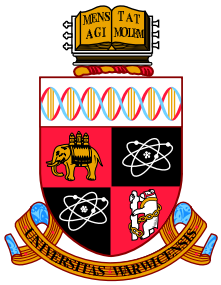
THE UNIVERSITY OF
WARWICK



Multipath Selection for Resilient Network Routing

by

Nayyar Almas Kazmi

a thesis supervised by

Prof. Jürgen Branke & Prof. Arie M.C.A. Koster

and submitted for the degree of

Doctor of Philosophy

Warwick Business School

May 2013

Abstract

In this dissertation we study the routing problem for multi-commodity survivable network flows, with splittable demands, and propose end-to-end path-based solutions where maximum link utilization is minimized, in order to improve resilience in existing telecommunication networks.

We develop mixed integer programming models, and demonstrate that, when the selection of disjoint paths is part of the optimization problem (rather than when k-shortest paths are pre-selected, as in earlier works), maximum link utilization is reduced and the overall network also balances out. We find that three paths are usually enough to reap the benefits of a multipath approach. A reduction in maximum link utilization also provides a margin by which demand values can grow without causing congestion.

We also prove that the disjoint multipath selection problem is NP-complete, even for the case of one node-pair. This warrants a recourse to efficient solution methods within ILP (such as decomposition), and to matheuristics. Our literature survey of applications of heuristic techniques, and those combining heuristics with exact methods, shows a research gap, which we attempt to bridge through a novel heuristic algorithm. The heuristic works well and, in several cases, yields better solutions than ILP (in a given time limit), or provides solutions for problems where ILP could not even find one valid solution in the given time limit.

We also study this problem within a decomposition methods framework: i.e., column generation. The pricing sub-problem is a mixed non-linear programme, for which we propose an ILP formulation. We find some lower bounds for missing dual values and use them as surrogates. We then show that the lower bounds are valid and present examples where the proposed pricing is applied to path generation for self-protecting multipath routing.

Declarations

This thesis is my own work and has not been submitted for a degree at another university.

The following papers have been published on the basis of material presented in Chapters 5, 6 and 7:

- N. A. Kazmi and A. M. C. A. Koster. An integer linear programming model for optimal self protection multi-path selection. In *ICUMT'11*, pages 1-7, 2011.
- N. A. Kazmi, A. M. C. A. Koster, and J. Branke. Formulations and algorithms for the multi-path selection problem in network routing. In *ICUMT'12*, pages 1-7, 2012.

Contents

Abstract i

Declarations ii

List of Tables vii

List of Figures ix

Abbreviations xii

Chapter 1 Introduction 1

Chapter 2 Network Routing - An Overview 9

2.1 Circuit Switching vs Packet Switching 10

2.2 Network Architecture 12

2.2.1 Layering 16

2.3 Routing Protocols 19

2.3.1 Open Shortest Path First (OSPF) 20

2.3.2 Multi-Protocol Label Switching (MPLS) 21

2.4 Delay and Link Utilization 23
2.5 Summary 24

Chapter 3 Understanding Resilience **25**
3.1 Standardization 26
3.2 Quantifying Resilience 28
3.3 Protection Mechanisms 32
3.3.1 Lower layer protection schemes 33
3.3.2 Logical layer protection schemes 35
3.3.3 Re-routing 42
3.3.4 Comparative analysis 43
3.4 Summary 48

Chapter 4 Heuristic Routing **49**
4.1 Heuristic Techniques 50
4.1.1 Greedy Algorithms 52
4.1.2 Local Search 52
4.1.3 Greedy Randomized Adaptive Search procedure 53
4.1.4 Tabu Search (TS) 54
4.1.5 Simulated Annealing (SA) 55
4.1.6 Variable neighbourhood Search (VNS) 56
4.1.7 Evolutionary Algorithms (EA) 56
4.1.8 Ant Colony Optimization (ACO) 57
4.2 Heuristics for Network Routing 59
4.2.1 Evolutionary Algorithms for Network Routing 60
4.2.2 ACO based heuristic solutions 63
4.2.3 Local search meta-heuristics 68

4.3 Hybrid Methods 74
4.4 Summary 79

Chapter 5 Solving Routing Problems with ILP **81**
5.1 Single Path Allocation Problem 82
5.1.1 Node-based Approach 83
5.1.2 Path-based Approach 85
5.2 Multipaths in Routing 90
5.2.1 Multipath structures for network resilience 90
5.2.2 Self-Protecting Multipaths (SPM) 91
5.3 Path Selection for SPM 93
5.3.1 Model description 94
5.3.2 Complexity of the path selection problem 96
5.3.3 Adding scenario-specific constraints 102
5.3.4 Model improvements 104
5.4 An alternative Path Selection Model 105
5.5 Summary 109

Chapter 6 Evaluation of the ILP Models **110**
6.1 Path Generation 111
6.1.1 Description of the algorithm 114
6.2 Computational Experiments 118
6.2.1 Testing the impact of path selection 118
6.2.2 A comparison between E1 and E2 models 136
6.2.3 A comparison of multipath and 1+1 methodology . . . 140
6.3 Summary 141

Chapter 7 Solution Methods for the Routing Problem **143**

7.1 Hybrid Approach 144

7.1.1 Description of the Heuristic Algorithm 145

7.1.2 Experimental Results 146

7.2 Column Generation 151

7.3 CG Framework for Routing 154

7.3.1 Dual formulation 156

7.3.2 The pricing problem 160

7.3.3 Solving the pricing problem 162

7.4 Summary 167

Chapter 8 Conclusions **168**

8.1 Future Research Directions 173

List of Tables

3.1 Effect of outages in terms of network availability. (Source [64]) 30

3.2 Restoration times and their impact on services. (Source [64]) 31

3.3 Framework for resilience 41

4.1 List of heuristic and meta-heuristic methods 52

4.2 Summary of meta-heuristic methods using the concept of Ants 65

6.1 Paths in a complete Graph 112

6.2 Path generation process 117

6.3 Network topologies, demands, and capacity per link 120

6.4 Optimal/best congestion values and optimality gaps for different combinations of number of disjoint paths and maximum number of selected paths 122

6.5 Number of paths used by different demands for germany17 instance 125

6.6 Number of paths used by different demands for europe28 instance 129

6.7 Number of paths used by different demands for germany50 instance 134
6.8 Best congestion values, dual bound, gap, and CPU times for 10 paths/demands and variable maximum number of selected paths 135
6.9 Comparison of number variables and constraints in test instances 138
6.10 Optimal/best congestion values and optimality gaps for different combinations of number of paths and maximum number of selected paths 139
6.11 Solution statistics for *di-yuan*instance 141

7.1 Optimal/best congestion values and optimality gaps for different combinations of number of paths and maximum number of selected paths 149

List of Figures

2.1 Functional View (Source: Pióro and Medhi [105]) 14

2.2 Network representations 15

2.3 A comparison of 7-layer OSI with 4-Layer Internet Model . . . 17

2.4 Hour-Glass Model 20

3.1 Path-based protection schemes 36

3.2 Shared Mesh Protection (SMP) 37

3.3 P-cycles 38

3.4 Self Protecting Multipath (SPM) scheme 39

3.5 Disjoint path selection 46

3.6 Classification of protection strategies 47

5.1 Directed representation of an undirected link 83

5.2 Representation of variables 99

5.3 Graph for SAT instance: $(\bar{X_1}) \wedge (X_1 \vee \bar{X_2}) \wedge (\bar{X_1} \vee X_2) \wedge (\bar{X_2})$ 100

5.4 Graph for an unsatisfiable SAT instance: $(\bar{X_1}) \wedge (X_1 \vee X_2) \wedge (\bar{X_2})$ 102

6.1 A sample graph 115

6.2 Nobel-eu network 119
6.3 Europe28 network 120
6.4 Maximum link utilization over all scenarios for germany17 with $|P_d|$=3, $\beta = 3$ 124
6.5 Maximum link utilization over all scenarios for germany17 with $|P_d|$=4, $\beta = 3$ 125
6.6 Maximum link utilization over all scenarios for germany17 with $|P_d|$=5, $\beta = 3$ 126
6.7 Maximum link utilization in different failure scenarios for germany17 instance 127
6.8 Maximum link utilization over all scenarios for europe28 with $|P_d|$=3, $\beta = 3$ 128
6.9 Maximum link utilization over all scenarios for europe28 with $|P_d|$=4, $\beta = 3$ 128
6.10 Maximum link utilization over all scenarios for europe28 with $|P_d|$=5, $\beta = 3$ 129
6.11 Maximum link utilization in different failure scenarios for europe28 instance 130
6.12 Maximum link utilization over all scenarios for germany50 with $|P_d|$=3, $\beta = 3$ 131
6.13 Maximum link utilization over all scenarios for germany50 with $|P_d|$=4, $\beta = 3$ 132
6.14 Maximum link utilization over all scenarios for germany50 with $|P_d|$=5, $\beta = 3$ 133
6.15 Maximum link utilization in different failure scenarios for germany50 instance 134

6.16 Number of paths used per demand 137

7.1 Framework of the hybrid approach 148

7.2 Computational time Vs number of demands 151

7.3 Branch and Price Framework for every B&B node 153

7.4 A network with 7 nodes and 10 edges 165

Abbreviations

ABC	Ant Based Control
ACO	Ant Colony Optimization
ADM	Add Drop Multiplexer
APS	Automatic Protection Switching
AR	Ants Routing
AS	Asynchronous System
ASCII	American Standard Code for Information Interchange
ASGA	Ant System Genetic Algorithm
ATM	Asynchronous Transfer Mode
B&B	Branch and Bound
BGP	Border Gateway Protocol
BLSR	Bidirectional Line-Switched Ring
BT	British Telecom
CAF	Cooperative Asymmetric Forward
CDR	Column Dependent Row Generation
CEN	European Committee for Standardization
CENELEC	European Committee for Electro-technical Standardization
CG	Column Generation

Char/s	Characters per second
CSMA	Carrier Sense Multiple Access
DGA	Distributed Genetic Algorithm
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Space
DSL	Digital Subscriber Line
DSP	Demand-wise Shared Protection
EA	Evolutionary Algorithm
ECMP	Equal Cost Multi-Path
EGP	Exterior Gateway Protocol
ENISA	European Network and Information Security Agency
ETSI	European Telecommunications Standards Institute
FRR	Fast Re-Route
FTP	File Transfer Protocol
GARA	Genetic Adaptive Routing Algorithm
GPP	Global Path Protection
GRASP	Greedy Randomized Adaptive Search Procedure
GRM	Genetic Routing Algorithm with migration
HTML	Hypertext Markup Language
HTTP	Hyper Text Transfer Protocol
IEC	International Electro-technical Commission
IEEE	Institute of Electronics and Electrical Engineers
IEEE-SA	IEEE Standards Association
IETF	Internet Engineering Task Force
IETF	Internet Engineering Task Force
IGP	Inter-Gateway Protocol
ILP	Integer Linear programming
IP	Internet Protocol

IS-IS	Intermediate System - Intermediate System
ISO	International Standards Organization
ISOC	Internet Society
ISP	Internet Service Provider
ITU	International telecommunications Union
LDP	label Distribution Protocol
LER	Label Edge Router
LP	Linear Programming
LSA	Link State Advertisement
LSP	Label Switched Path
MACO	Multiple Ant Colony Optimization
Mb/s	Mega bits per second
MILP	Mixed Integer Linear Programming
MPLS	Multi-Protocol Label Switching
MTBF	Mean Time Between Failures
MTTR	Mean Time To Repair
NGN	Next Generation Network
NP	Non Polynomial
OMP	Optimized Multipath
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
p-cycle	Protection Cycle
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RDP	Remote Desktop Protocol
RFC	Request for Comments
RIP	Routing Information Protocol
RMP	Reduced Master Problem

RPP	Regional Path Protection
RSVP	Resource Reservation Protocol
RTP	Real Time Transfer Protocol
RWA	Routing and Wavelength Allocation
SA	Simulated Annealing
SAT	Satisfiability Problem
SDH	Synchronous digital Hierarchy
SI	Swarm Intelligence
SLA	Service Level Agreement
SMP	Shared Mesh protection
SMTP	Simple Mail Transfer Protocol
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management protocol
SONET	Synchronous Optical Networking
SPF	Shortest Path First
SPM	Self-Protecting Multi-Path
SPP	Span Path Protection
SRLG	Shared Risk Link Group
SynthECA	Synthetic Ecology of Chemical Agents
TCP	Transmission Control Protocol
TDMA	Time Division Multiplexing
TE	Traffic Engineering
TFTP	Trivial File Transfer Protocol
TS	Tabu Search
TTL	Time To Live
UDP	User Datagram Protocol
UPSR	Unidirectional Path-Switching Ring
VNS	Variable Neighbourhood Search

VoIP	Voice over Internet Protocol
WDM	Wavelength Division Multiplexing
Wi-Fi	Wireless Fidelity
WWW	World Wide Web
XML	Extended Markup Language

1

Introduction

As a result of deregulation and competition, the telecommunication industry is booming and new applications, exploiting the potential of networks, are on offer everyday. Applications such as cloud computing, Virtual Private Networks, VoIP, social networking and many more have become commonplace. All these applications use the logical abstractions of the underlying physical network and are heavily, if not solely, dependent on a functioning, reliable network service. A disruption in the underlying network will cripple the whole set of services and the application using it. Therefore, a much greater emphasis is

being placed on devising strategies to combat network failures. Governments as well as companies are making back-up plans to cope with any untoward event. A high level of availability of network is expected and required by governments, regulators, and companies while awarding contracts to network service providers. Penalties are imposed for service going below the agreed-upon level. Reliability in the network service provides a competitive edge over others, whilst even a slight disruption in the network's operations may mean a loss of revenue in millions. A study conducted by Gartner Group revealed that, in 2004, losses of around $500 million could be attributed to network failures [64].

The survivability of the network is at risk from both intentional and unintentional faults. While sabotage is outside the scope of this thesis, link failure resulting from unintentional human errors, malfunctioning of a device or a fiber cut is the central concern. Fiber optic cable failure is one of the highest reported causes of a network outage [64] and, despite best efforts, such failures are inevitable. Consequently, we need strategies to minimize these errors and to detect them quickly and recover from them in a reasonable time. That is why self-protecting mechanisms - to provide resilience - are our prime focus.

In order to come up with a survivability strategy, a minimum requirement is to have a bi-connected physical network. A bi-connected network is one in which there exist at least two disjoint paths between any two nodes.

An important question is: how to measure the quality of a network. Some years ago,, criteria such as minimum bit-rate, maximum packet loss or maximum delay in the packets, were used to quantify the network service. Nowadays, however, more sophisticated methods where the performance of a

network as a whole is quantified, are used. Different measures for the network performance thus used include network availability, network reliability, restorability and unavailability. Availability is one of the most commonly used measures. It comprises the cumulative availability of each of the network elements. Most service level agreements specify the agreed-upon network availability, of, say 80% of the time, and penalties are imposed if the service drops below this mark.

Protecting node failures is relatively straightforward as it is simply a question of providing an additional device as a back-up. However, the most challenging task is that of protection against link failure. Link failure is more difficult to detect. Protecting a link failure involves detecting a failure, establishing a backup path to override the failed link, and switching to the backup path both by the sending node and the receiving node. In order to provide a good quality of service, a failure restoration must take place faster than the time it takes to reach the effect of failure to the end user. Using expensive transport technologies such as SONET, one can guarantee such failure recovery, but it is prohibitively expensive and inflexible for future expansions. On the other hand, if the restoration is left only to the routing protocols, costs do come down and we obtain flexibility in design but a quick recovery from failure is not guaranteed.

Earlier, dedicated backups were used. These ensured very fast protection, but almost doubled the network costs. Network service providers would not want to tie up too much money in the installation of spare cables, in anticipation of a failure, which might never occur. Back-up capacity generates no revenue. In order to maximize revenue, service providers therefore try to minimize the spare capacity reserved for failure protection, without impairing

the service quality. Therefore, options whereby the back-up capacity can be shared have become more popular. A step further is to remove the classification between the primary and the backup capacities, and to utilize all the available capacity and mitigate failures by sharing the capacity. According to [64], path restoration strategies with sharing and stub release are "one of the most efficient class of survivable networks".

In sharing the capacity among different paths, one of the strategies being looked at is demand splitting. It involves distributing the traffic over multiple paths, so that at any one given time, only a fraction of demand traffic is affected in case of a failure. This strategy distributes the impact of a failure.

The idea of using multipaths is not new. For example, applications such as bit torrent, which are used these days extensively for downloading huge files (e.g., movies, songs, books) use more than one path for downloading. Congestion control on each path is handled independently. For Transmission Control Protocol (TCP), the Internet Engineering Task Force (IETF) has formed a working Group named TCP Multipath [1] to develop functionality that will enable existing TCP sessions to use multiple paths simultaneously. Their aim is to prepare mechanisms to deploy multipath capability in the existing networking infrastructure, without making any significant changes. At network layer too, multipath capability is used. For example, the Equal Cost Multipath (ECMP) mechanism within the Open Shortest Path First (OSPF) routing protocol splits demand on all of the available shortest paths.

In this research we aim to look at the strategies that use multipaths at network layer routing. The idea of multipath requires selection of appropriate paths for use. If the paths selected are not the best, in terms of capacity utilization or cost, then we cannot reap the benefit of using multipath strategies.

The aim of our study is to look at the *path selection* problem for these multipaths. The study focuses on protection mechanisms that use multipaths for traffic routing. The Self-Protecting Multipath (SPM) mechanisms proposed by Michael Menth of University of Würzburg (Germany) forms the starting point of this work.

When routing strategies are devised as part of network dimensioning, then the costs are to be optimized. But, generally, networks are designed for a failure-free situation, and then redundancy to ensure availability is added at the second stage. For an existing network infrastructure, the aim is to optimize the available resources. Considering that the networks already exist, with known infrastructure, the objective of this research is to study the issue of *survivability* and to examine different routing and re-routing mechanisms with a view to identifying their *resilience*.

This research focuses on identifying those end-to-end paths that require *less* spare capacity and remain robust. We want to further identify such routing plans that can operate in all single link failures – thus ensuring the overall availability of the network to remain high.

For a multipath routing, the *selection* as well as the *number* of such paths needs to optimal. The simulation results of the SPM have shown that the use of multiple disjoint paths decreases the extra capacity required as a backup. The need for determining the optimal number of paths (for a given traffic demand) and then setting the rules for selecting the most reliable, yet inexpensive ones, is a challenge for researchers, managers and network administrators alike. Under this study we have taken up and addressed this challenging task of network flow routing, *with the objective of determining the optimal disjoint paths between each node-pair with minimum network capacity*

utilization.

The main research questions we consider in this thesis are:

- How to formulate the problem for disjoint path selection that optimizes the worst case capacity utilization?
- What is the structure and complexity of the formulated problem?
- What solution methods should be adopted based on the explored characteristics of the problem?

The contributions of this research is towards:

- survey of exact and heuristic approaches to solve this problem,
- solving this complex problem through development of exact models and heuristic methods and
- exposition of the computational complexity of various optimization problems under study.

The thesis is organized as follows:

Chapter 2 provides a brief introduction to the network architecture and its layering concepts. These concepts form a basis to understand the network routing mechanisms that are being used in practice. We discuss two routing protocols in detail, namely OSPF and MPLS, as these represent two different philosophies to route. The first one uses hop-to-hop based routing decisions, while the second one uses pre-determined paths. In this study we are concentrating on end-to-end paths, so it is important to compare and contrast the path-based method with the hop-to-hop method, which is the most widely

used protocol in today's Internet. Chapter 2 also discusses the concept of delay and discusses how it relates to minimizing the maximum link utilization.

Chapter 3 focuses on the topic of resilience. It looks at the quantification and management of resilience and provides an overview of the organizations responsible for managing the "resilience" aspect of a network. Here we also discuss the protection strategies that are either in use or are proposed in the literature, and provide an overview of their defining characteristics. We identify a research gap and explain our strategy to address it.

Chapter 4 looks at network routing from a completely different perspective. It summarizes the heuristic methods that are used for routing. Nowadays, when network sizes are increasing, very large and complex problems arise. Solving them in a exact manner is not always possible. The alternative is to use heuristics in some way. The study of heuristic methods thus provides us with an insight that can be utilized in thinking out of the box. In this Chapter we have reviewed not only the pure heuristic methods, but also the mixing of heuristics and exact methods. We provide examples of such mixing and discuss their pros and cons. The chapter serves as a necessary preclude to our development of a hybrid algorithm in Chapter 7.

In Chapter 5 , we present an overview of routing solutions based on multipaths and then discuss the Self Protecting Multipath (SPM) routing model that forms the basis of our work. The Mixed Integer Linear Programming (MILP) models for the routing problem are presented and discussed in this Chapter and we also compare different models and discuss the complexity of our network routing problem.

Models presented in Chapter 5, were implemented using different programming environments. Using these implementations we present, in Chap-

ter 6, different experiments that we conducted in order to evaluate the performance of each model. We compare their performance with some benchmark instances available online.

Chapter 7 builds on the discussion in Chapter 4 and presents a heuristic method to solve the routing problem under discussion. Here we furnish a hybrid algorithm, which combines ILP and neighbourhood search heuristic to find the optimal routing. We also present a comparison between this heuristic algorithm and the exact methods discussed in Chapter 5. Chapter 7 also presents a column generation framework to solve the routing problem. The formulation of the pricing problem and its characteristic are discussed here. We also show the validity of the pricing problem with the help of an example.

Chapter 8 comprises a conclusive summary of the whole thesis and lists the salient research contributions. It also discusses the lines of future inquiry flowing out of the present research as well as other possible approaches that can be adopted to extend this work.

2

Network Routing - An Overview

In this Chapter we describe the general characteristics of a network. We discuss the architecture of a network, its layout, organization, routing mechanisms and failure quantification. We also provide a brief overview of routing in OSPF and MPLS protocols, as these relate directly to our work.

A computer network is a group of connected devices. These devices are capable of interaction with each other. A computer network is normally required for data and voice communication and sharing of resources (such as files, printers, information etc).

The communication within a network is governed by the network's administration policies. Many networks can be connected together to form a larger network. The connected networks normally form a hierarchical structure, where backbone networks are connected to one another and other networks hook onto this larger network. The most important example of such inter-networking is the Internet, which now has a global presence. It has gradually gained so much importance that today's social and economic development hinges upon smooth and reliable functioning of the Internet (and thus networks in general). The dependence on the Internet is going to further increase with time. Increasingly more devices are going to be attached to it and more services will be delivered through it. Compared to 2010, it is estimated that in 2020, there will be about 10 times more users and about 50-100 times more traffic per user [83]. This will result in a total growth of traffic by a factor of 500-1000. To cater for this traffic, the throughput has to increase by at least a factor of 10. The other issues which are becoming more and more important in this context are climate-change. At present, 3% of world-wide energy is consumed by ICT infrastructure [83], and with the increase in the bulk of connected devices, energy consumption will become a serious consideration. All these factors make efficient management of networks (and Internet) more crucial. The need for a sensible, reliable and flexible network management can not be over-emphasized.

2.1 Circuit Switching vs Packet Switching

The first ever electronic communication can be dated back to about 100 years, when the first call was made by Alexander Graham Bell in 1876. This led to

the development of modern-day telephony, which is based on the principle of circuit switching. All call requests go to a switching board, where each call is switched to a designated connection between caller and the recipient of the call. The connection stays available for the entire duration of the call. At the end of a call, the connection is removed and the released circuit/capacity becomes available to other callers. Since each caller has its own private, guaranteed, isolated (constant) data rate from end-to-end, circuit switching provides high quality un-interrupted service. In the past, each connection was an end-to-end physical wire. Later with the development of high capacity wires, it became possible to allocate many connections on the same physical links by allocating bandwidths to each call. Thus, instead of physical circuits, virtual circuits are assigned to each call. Public Switched Telephone Network (PSTN) is an example of a service based on circuit switching.

The drawback of circuit switching is that it requires management of circuits (states) at the beginning and end of each connection (call) to establish and close each circuit. Also, the allocated bandwidth for any connection must be able to carry the maximum traffic generated. However, since the network traffic is bursty in nature, it would mean that the traffic will reach its maximum only sporadically. Hence, the allocated bandwidth will remain unused for most of the time. Different applications using the network have different data rate requirements. The required data rate varies significantly from application to application. For example, for video streaming 6 Mb/s is required, while for typing only 1 char/sec would be enough. Allocation of fixed data rates makes circuit switching inefficient.

By contrast, packet switching does not require dedicated circuits. Data is divided into small chunks called packets, and is sent over the network. A

packet is a self-contained envelope and, when it reaches any switch/router, it is forwarded to the next hop. No connections are required to be established before starting the communication. Similarly, no post communication management is involved. Each router maintains a Forwarding Table, which stores the next hop based on the destination address of any packet. Its main advantage is that it uses link capacity efficiently. In case of failures of links or routers, alternative paths can be re-calculated and packets can be re-routed on those paths. Because of it flexibility of use, packet switching is used for data communication over the networks. The Internet communication is also based on the principle of packet-switching. One of the founding architects of the Internet, David Clark, gives an insight on this choice of packet-switching in his article [28]. There are two main reasons: firstly, the purpose of the Internet was to connect existing networks, most of which were packet-switched. Secondly, the type of services that were required (such as remote login), were also easily implementable through packet-switching. Hence, packet-switching became an integral part of the Internet communication system. Resultantly, most of today's networks are packet switched.

We are also looking into the problem of routing for the packet switched networks. We explain below how these networks are organized and managed.

2.2 Network Architecture

The networks in which routing decisions and management is controlled by a single entity are called Autonomous Systems (AS). For example, an Internet Service Provider (ISP) has control over its own AS. Each ISP can make its routing decisions independent of other ASs. The routers within an AS main-

tain the forwarding tables. These table have entries containing forwarding addresses corresponding to each destination. These tables are maintained by sharing information with each other. These tables are refreshed from time to time, to keep the network status information up to-date. The routers at the boundary of AS have the capability to accept and send packets from/to other ASs.

The traffic generated in an AS is transported on a physical network, called transport network. For example, in the UK, British Telecom (BT) has a large physical infrastructure, and leases the bandwidth to many ISPs. It is also possible for some network owners to install their own physical facilities. The transport network provider may send the traffic from multiple ISPs on the same network.

The demand for traffic network (or the logical network) is based on the users. It is a random process, as it not known when a user will initiate a request and how much traffic it will generate on any link. There needs to be a mechanism to estimate the traffic arrival rates. Transport network providers estimate traffic requirements based on extrapolated traffic demand and set up their services on semi-permanent basis accordingly. Changes are made only periodically, such as at the end or renewal of contracts.

The traffic networks use protocols such as OSPF, Intermediate System-Intermediate System (IS-IS) for routing, while for transport networks technologies such as synchronous digital hierarchy (SDH), synchronous optical networking (SONET) and Wavelength Division Multiplexing (WDM) are used. In a traffic network we have routers and links, while in a transport network there are digital cross-connects and cables for mesh networks or Add Drop Multiplexers (ADMs) for ring topologies. Fig. 2.1 shows how both transport

and traffic networks are inter-related. A link might appear to be a direct one between two nodes A and B in a logical network. But, in actual fact, a set of physical connections might be used to provide such a link. Figs. 2.2(a) and 2.2(b) show the physical and logical views of the same network. In order to provide resilience, it is important to understand this dependence between the two links, since a single failure at a physical level might translate into multiple logical link failures. Hence that the back-up paths must not use the same physical connection that are used by the primary paths.

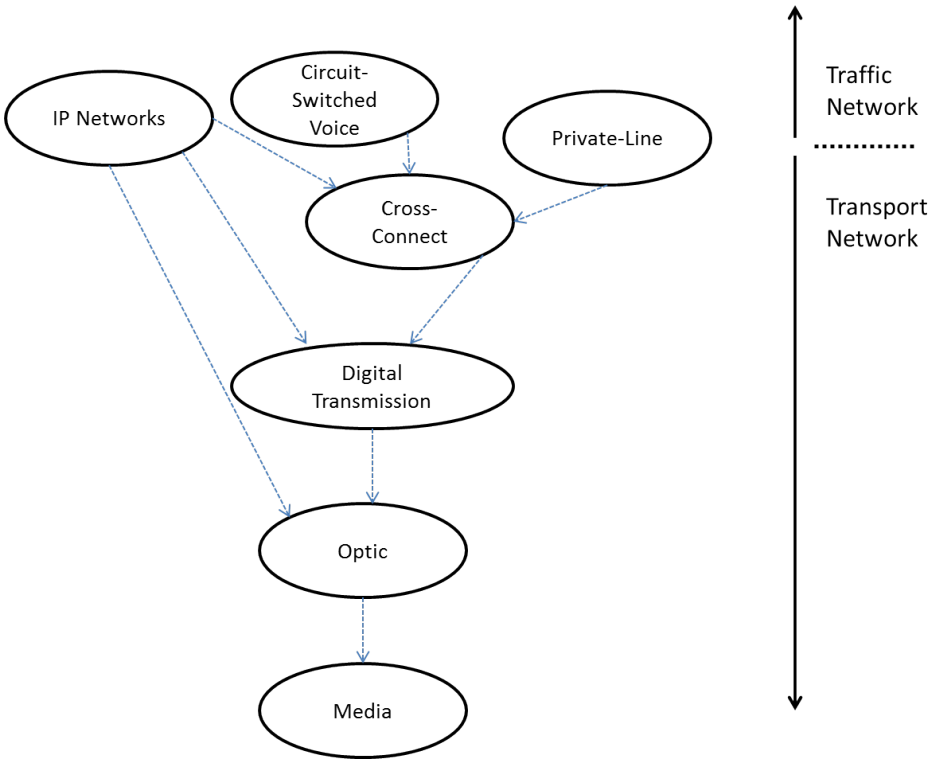
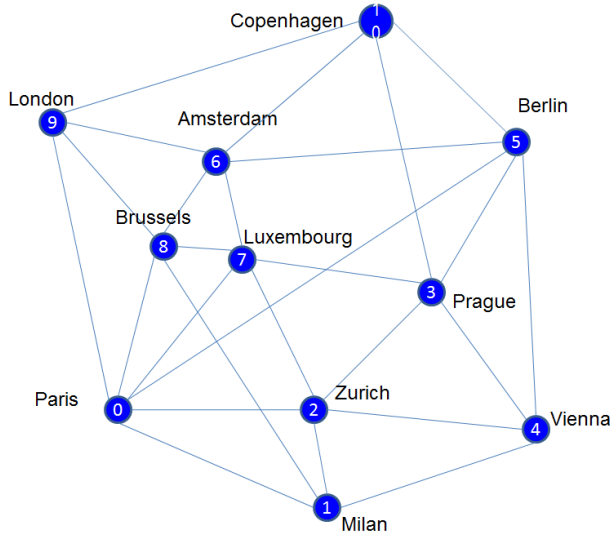
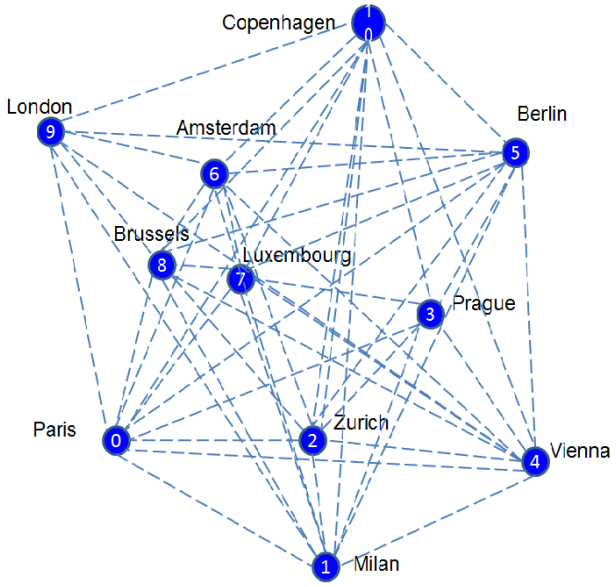


Figure 2.1: Functional View (Source: Pióro and Medhi [105])



(a) Physical Network



(b) Logical Network

Figure 2.2: Network representations

2.2.1 Layering

The concept of layering is fundamental to network communications. Each layer works independently, encapsulating its internal details from other layers. Hence the complex task of network management is decomposed into manageable independent units. There are two famous models for telecommunication networks. One is the OSI[1] model which describes network as a 7-layer model. The second model, which was developed later, was a more concise 4-layer Internet model, in which layers of OSI model were merged and re-defined. Fig. 2.3 shows the relationship between the two layering models. These layers are:

- Application layer
- Transport layer
- Network layer
- Link layer

Each layer is governed by a clearly defined protocol for that layer. These protocols facilitate the functions of each layer and encapsulate the internal processing details from other layers. The transmission takes place by dividing the data into smaller units. Each layer also has its own unit of transmission. The peer layers at the sender and receiver ends communicate with one another.

The actual request for communication is generated at the application layer. There are a number of protocols that run at the application layer[2]

[1] Open Systems Interconnection

[2] Some of the examples of application layer protocols are HTTP, FTP, TFTP, SNMP, SMTP, DHCP, RDP, Telnet (It is a network protocol that uses a virtual terminal connection to provide a bidirectional interactive text-oriented communication facility on the Internet

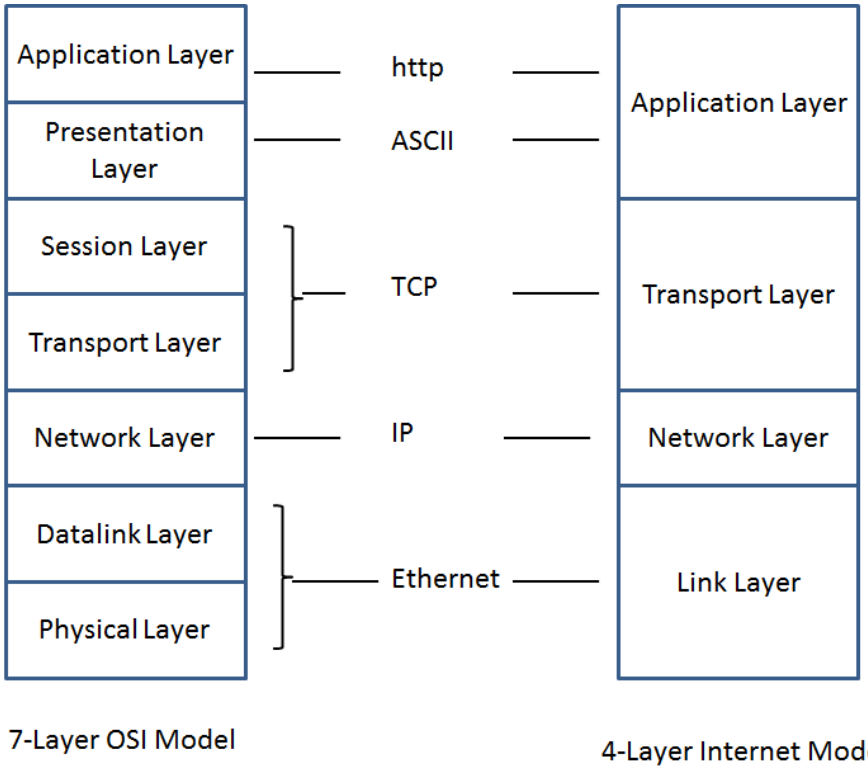


Figure 2.3: A comparison of 7-layer OSI with 4-Layer Internet Model

At transport layer, the data received from the application layer is divided into segments and a header is annexed to each segment. At the receiving end, the transport layer is capable of reading the header and can re-assemble all the segments and then pass it on to the application layer. Depending on the communication needs, the transport layer may use Transmission Control Protocol (TCP) or User Datagram Protocol (UDP).

TCP is used when a reliable and secure data transmission is required. Applications such as emails, worldwide web and file transfer would require such a communication. A connection between source and destination hosts is established first and then TCP segments are transmitted. The receiving

or local area networks), and X-Windows (It is a network protocol that provides a basic graphical user interface for networked computers).

transport layer ensures that the segments are received in an orderly manner. It acknowledges the receipt of all the segments and also assembles them to forward them to the application layer.

UDP, on the other hand, is used when the speed of communication is important and reliability is not required[3]. In contrast with TCP, UDP is a connection-less, unreliable service that does not check for congestion control or ordered delivery of data segments. As it provides a minimum service, it is much faster than the TCP and requires less capacity/bandwidth.

The data segments from transport layer are passed on to the network layer. At network layer, Internet Protocol(IP) is responsible for onward transmission. Here transport layer data segments are annexed with an IP header and these IP datagrams or packets are sent over the network through the Link Layer. IP is a connection-less service, and it sends packets hop-by-hop to the destination. At each hop the receiving router will send the packet to the next hop. Each router maintains a forwarding routing table for each possible destination. These routing tables are maintained by a periodic transfer to information among neighbouring routers. IP is a best-effort service: i.e., it will send the packets to its destination, but if for any reason the packets cannot be forwarded, they are dropped. There can be multiple reasons for dropping the packet, such as congestion on the link, or time lapse for the packet.

The routers are capable of processing only one packet at a time. If more than one packet arrives at they same time, they are stored in a buffer and forwarded on a first-come-first-served basis. However, when the buffer is already full, the arriving packets are dropped. There can be another reason to drop the packet, that is, when a packet over-lives its life. Each packet has a

[3]UDP is used by applications such as VoIP, DNS, SNMP, DHCP and RIP.

field called Time To Live (TTL). Before forwarding a packet, the router checks packet's TTL. If TTL has not expired, the router decrements the TTL value for the packet before forwarding it to the destination. At present, IP version 4 is being used for packet formats and addressing. It uses a 32-bit addressing scheme. It is being replaced by IP version 6, which uses 128 bit addressing. The details of these schemes is beyond the scope of our work.

The routing between the source and the destination hosts is regulated by one of the routing protocols. For packet routing within an AS, an Inter Gateway Protocol (IGP) is used. OSPF and IS-IS are examples of IGP. When a packet leaves the AS, its routing is governed by the Exterior Gateway Protocol (EGP). The Border Gateway protocol (BGP) is the commonly used EGP over the Internet.

The IP is designed to isolate end-to-end protocols from the details of underlying networks. It is commonly presented as an hour-glass model. See Fig. 2.4. IP can be used on top of any underlying transmission technology such as Ethernet, Wi-Fi, DSL etc, which come under the link layer category. Similarly, it is capable of handling transmission requests by any type of application or transport layer protocols.

2.3 Routing Protocols

Here we will discuss the characteristics of routing protocols relevant in the context of this study. We will discuss OSPF, which is the most widely used IGP[4]. We will also discuss MPLS mechanism that works with any kind of

[4]For the sake of completeness, it is worth mentioning here that IS-IS is also a link-state protocol that uses same mechanism as OSPF to maintain the network's view and to calculate shortest paths. While OSPF was built for IP V4, IS-IS was not linked to any specific IP layer

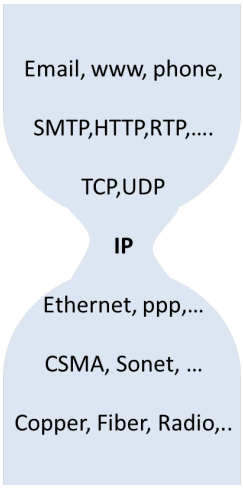


Figure 2.4: Hour-Glass Model

transport technology and creates end-to-end circuits. It works on packet-switched networks, and is meant for reliable transmission. It is different from OSPF, and we will discuss and contrast their characteristics as well.

2.3.1 Open Shortest Path First (OSPF)

OSPF is a link-state protocol, in which each router creates its own view of the network by getting connectivity information from each neighbouring router. Every ten seconds a router sends a Hello packet to each of the routers connected to it. If a response is received, the link is considered up. If an acknowledgment is not received after four Hello messages, the link is considered in down state. A router sends Link State Advertisement (LSA) message to all its directly-connected nodes. Each router maintains a Link State Database, which needs to be synchronized from time to time. Using the costs/weights pre-assigned to each link, each router (node) generates shortest paths (using Dijkstra's

protocol. It has less message communication among routers, which enables it to support more routers in one area than what OSPF can support. The details on IS-IS can be read at [63].

algorithm) from itself to all other destinations in the network and stores them in its routing table. The weights are arbitrary values assigned to links by the administrators. The protocol has a global view of the network at all times. These link state tables and routing tables are reconstructed each time a link's state is changed. This makes OSPF processor and memory intensive. There may be periods when link state tables of different routers are in different states, or link state table and routing table of a router are not synchronized, making the network unstable. When there is a failure, the network is flooded with messages to update the change. Consequently, all the routers have to update their routing tables and shortest paths are re-calculated. If for any destination, there are multiple shortest paths available, then the flow can be equally split among them. This is called Equal Cost Multi-Path (ECMP) routing.

2.3.2 Multi-Protocol Label Switching (MPLS)

Multi-Protocol Label Switching is the technique for packet flow for different classes of users and services in the core IP networks [105]. This is achieved by assigning end-to-end virtual paths or tunnels. It enables the flow of packets between a node-pair for each class of user/service with different QoS requirements. Its explicit routing allows packets to follow a pre-determined path, instead of following a path computed by hop-by-hop routing, as is the case in OSPF. MPLS makes use of Label Switching. The Label-edge router (LER) assigns labels to the incoming traffic. Traffic follows a Label Switched Path (LSP). Labels are switched by Label Switching Routers (LSR). The communication between LERs to set up labels takes place through the Label Distribution protocol (LDP) or Resource reservation Protocol (RSVP). One of

the strengths of MPLS is that it allows establishment of an LSP that passes through different transport mediums, e.g. ATM, Ethernet or Frame relay.

MPLS combined with Traffic Engineering (TE) provides constraint-based routing with (shortest) paths that fulfills the QoS requirements. Different traffic classes with various service requirements can follow different paths. MPLS-TE Fast re-route (FRR) is a feature provided under RSVP-TE. It uses pre-determined paths to provide service in case of a network failure. These can either be one-to-one back up paths, or a facility backup. In case of one-to-one backup, for each primary LSP, each edge or interface is protected. Facility backup, on the other hand, is a many-to-one protection. Here, one edge shared by many LSPs is protected by one bypass path. These backup paths provide pre-planned protection against both link and node failures. In contrast to the OSPF recovery mechanisms, which are slow and congest the network with messages, FRR allows an LSP to be repaired locally at the point of failure. This reduces the recovery times down to around 50 ms, a standard set for SONET rings. In contrast, recovery time in OSPF can be up to several minutes, and, consequently, loss of packets may occur [73]. On a FRR enabled path, when an edge fails, a detour is established starting from the node preceding the point of fault and fault information is communicated back to the ingress router. On receiving the information, the LER redirects the remaining traffic to the backup LSP. However, maintaining the state information for backup paths can be very expensive and can also create excessive traffic.

2.4 Delay and Link Utilization

The demand for traffic networks depends on their usage. All users generate traffic independently. The pattern of traffic generation is a random process. It cannot be described with normal or exponential distributions. It can be described as self-similar traffic whose distribution has a heavy tail.

In order to estimate the traffic, packet (exponential in size) arrival is assumed to follow a Poisson distribution and the well-known M/M/1 queuing model (see [105]). If average packet size is denoted as S_p and the capacity of a link as B bits/sec, then μ_p = average service rate = B/S_p packets per sec (pps). If the average arrival rate is denoted by λ_p, then

$$average\ delay = \frac{1}{(\mu_p - \lambda_p)}$$

$$average\ link\ utilization \quad \rho = \frac{\lambda_p}{\mu_p}$$

The average delay is non-linear in nature[5]. As the average arrival rate λ_p and average service rates come closer, the resulting ρ will approach 1, then delay will go to infinity. Hence, in order to find out an acceptable level of delay, an acceptable level of link utilization can be used as a measuring gauge. By reducing the overall link utilization, we can reduce congestion. *This explains our use of the minimizing maximum link utilization as the objective function.* It maximizes the robustness of the network against unpredicted demand. This objective function has been used by many other studies. For example, Menth et al. [88] and de Sousa et al. [35] have used it for network

[5]To calculate the link utilization, we started with the assumption that the packet arrival rate follows Poisson distribution. However, this assumption is not always true, and the actual delay is generally worse than what we calculate using M/M/1 queuing model [105].

optimization problem, while for the OSPF weight setting problem, Fortz and Thorup [53] have used a cost function based on link utilization, which penalizes more heavily when the link usage, with respect to its total capacity increases.

2.5 Summary

In this chapter we have given an overview of routing in networks. We have discussed how a network is organized and how different protocols operate. Since we are studying the routing problem at the network layer, we explained in detail the working of OSPF and MPLS, which operate on network layer. OSPF provides hop-based routing and has has limitations with respect to protection against failures. It is slow and its recovery process can cause congestion over the network. MPLS, on the other hand, provides path-based routing, which makes it faster and it supports traffic engineering features to provide resilient routing.

In this present work, we will focus on these path-based mechanisms and will study how the potential of these path-based structures can be exploited for a more balanced and resilient network. We will use the minimization of maximum link utilization as the objective of our optimization problem, which we have shown in this chapter is a measure to quantify delay. Minimizing the worst-case link utilization will result in maximizing the robustness of the network against unpredicted demand growth as well as reduce congestion and delay.

In the next Chapter we will explain different path-based resilience mechanisms and will discuss their differentiating characteristics.

3

Understanding Resilience

In this Chapter we discuss the concept of resilience in greater detail and examine the practical issues surrounding resilience. We also look at the quantification and standardization issues related to resilience. We provide an overview of protection and restoration strategies that are used to provide resilience.

We compare different resilience strategies and review the research done in the area of survivable network routing with multipaths and place our study in the existing body of knowledge, by identifying the research gap.

3.1 Standardization

As explained in the previous Chapter, a network can be viewed as a collection of devices, protocols, procedures and algorithms. It is a layered structure. In order to successfully communicate between different layers and among different peer devices, all the involved components must, at each layer, speak/comprehend the same language. The concept of layering helps in encapsulating the internal details of each layer. Thus, only relevant and sufficient information is visible at the other layers. This inter-communication requires standardization of all components (soft and hard) involved in a network.

A network spans, or talks to, other networks, across geographical regions in different administrative controls. Hence, any standardization must involve all countries around the world to agree and act according to the pre-defined standards. To facilitate this process, the International Telecommunications Union (ITU) was formed in 1934 [1]. ITU is now a UN specialized agency for ICTs that allocates satellite, wireless communications and radio frequency spectrums, world-wide, for 193 member countries and 700 private entities in 12 regions. It sets a very wide range of standards, and "is at the heart of the ICT sector ... to create a seamless global communication system that is robust, reliable and constantly evolving."[2]

The standards developed by ITU are for networking, voice and video compression, Internet access, transport protocols, and a large number of other aspects which are required for networking at local as well as international level. These standards are developed through study groups, focus groups and global

[1] It was actually formed in Paris in 1865 as International Telegraph Union, but converted to ITU in 1934.

[2] Source: www.itu.int/e/about/pages/default.aspx [accessed on 6/12/12]

meeting places. Their recommendations cover all types of networks such as dial-up, optical systems, Next Generation Network (NGN) and IP-networks.

There are a number of other standards development organizations, such as ISO, IEEE-SA (Standards Association) and IEC (International Electrotechnical Commission) that work internationally and coordinate with regional organizations. For European countries, the European Telecommunications Standards Institute (ETSI) provides a quick forum for meeting. It has industry specific groups. For example, there are groups for Measurement and Ontology for Network Services, Information Security Indicators and Identity Management for Network Services.

The European Committee for Electro-technical Standardization (CENELEC) is a voluntary standards organization and facilitates inter-European trade. It collaborates with International Electro-technical Commission (IEC). European Committee for Standardization (CEN) collaborates with International Standards Organization (ISO).

The Internet Society (ISOC) is a cause-driven, independent organization for Internet policy, technology standards and development. It facilitates open development of standards, protocols and infrastructure. It tackles issues such as domain name systems, Internet exchange points, Internet Protocol Addressing and routing security. It only deals with policy issues. The engineering issues are dealt with by the Internet Engineering Task Force (IETF). IETF produces technical documents to make Internet work better. The working groups within IETF cover each technical area. A Request for Comments (RFC) is issued and it is placed in public domain. Anyone can contribute to RFCs. Once discussed, these documents become the official documents. For example OSPF version 2 is documented as RFC2328. MPLS main architecture

is RFC 3031. IETF standardize all protocol layers from IP to application layer, which also include standards for services such as emailing and http. HTML and XML standards are not tackled by IETF. These are left to the WWW consortium.

3.2 Quantifying Resilience

The quantification of resilience has *not* been dealt with by any of the aforementioned forums. The European Network and Information Security Agency (ENISA) [44] points out that the word "resilience" has been used loosely and the main challenge in understanding resilience is to accurately define it and devise matrices to measure it. There is a need to develop a standardized framework to measure and respond to the resilience issues.

Resilient and Survivable Networks (Resilinet) [127] is an umbrella project, initiated jointly by the University of Kansas (US) and the University of Lancaster (UK), to develop resilient architecture for the future Internet. One of the outcomes of this project is a framework for the measurement of resilience and survivability. It defines resilience as an ability of the network to provide and maintain an *acceptable level of service* in the face of various *faults and challenges* to normal operation.

The term "network service" refers to all services that are provided through networking, such as VoIP, Location services, IP, frame relay and wi-fi. It may be noted here that services which only exploit the existing network structure, such as distance learning, web browsing etc are not network services; rather these are services based on networks.

An acceptable level of service is defined through various matrices and

is normally written in the Service Level Agreements (SLAs) [3].

Another way to look at acceptability is through the impact of disruption in services. Three levels of service acceptability can be defined:

- acceptable: on or above target
- impaired: between minimum and target level
- unacceptable: below minimum required

In case of failures, a resilient service will be able to operate in severed conditions, while a non-resilient service will quickly deteriorate into an unacceptable level.

One matrix to determine the network resilience is its availability. For backbone networks, availability expectation typically varies between 0.999 to 0.99999. Availability is measured in terms of all the components that are required for the operation of the network. The joint availability of all these components will give end-to-end availability of the network. The availability of each component is calculated by a well-known formula given below:

$A = \frac{MTBF}{MTBF + MTTR}$

MTBF is the the mean time between failures and MTTR is the mean time to repair.

Table 3.1 shows the relationship between network availability and outage time, i.e.: the time for which network is unavailable.

The mechanisms that can restore the network from failures need to be in place. Often a requirement of a single link restoration of 50 ms is quoted.

[3]SLA is a contract document between service provider and the client.

Table 3.1: Effect of outages in terms of network availability. (Source [64])

Availability	**Outage Time**
0.9 (1 nine)	867.6 hours/year (36.53 days)/year
0.95	438.2 hours/year (18.26.53 days)/year
0.99 (2 nines)	87.66 hours/year (3.65 days)/year
0.995	43.83 hours/year (1.83 days)/year
0.999 (3 nines)	8.77 hours/year
0.9995	4.38 hours/year
0.9999 (4 nines)	52.60 minutes/year
0.99995	26.30 minutes/year
0.99999 (5 nines)	5.26 minutes/year
0.999995	2.63 minutes/year
0.999999 (6 nines)	0.53 minutes/year

It is the response time of SONET APS (Automatic Protection Switching), in which failures are detected in approximately 20ms, signaling takes place in 10 ms, another 10 ms are needed for transfer-delay. A spare time of 10 ms is added to cover the risk. However, with remote destinations, a restoration time of 50 ms is very hard to achieve. Thus the question arises, what is the acceptable level of restoration time? Table 3.2 shows the impact of restoration times on the services.

It is the responsibility of a network service provider to establish and maintain a service (at an acceptable level) for up to a maximum duration of time (as per agreed SLA such as 99.9 (3 nines), 99.999 (5 nines)). The service provider is required to anticipate and safeguard against possible threats and faults, such as:

– unintentional operational mistakes

– catastrophes and disasters - both natural and human-induced

Table 3.2: Restoration times and their impact on services. (Source [64])

Restoration Times	**Impact**
less than 50 ms	Recovers without major impact
50ms-200ms	5% voice bands disconnect
200ms - 2s	Voice band call disconnect
2s - 10s	Circuit switched services drops, Private lines disconnect, Some data sessions time out
10s 5 min	X.25packet disconnect, Data session time out
5 min - 30 min	Network congestion, social/ business impacts
more than 30 min	Major social/business impacts

- malicious attacks on traffic
- hardware malfunctions
- Events such as weekends, Christmas, elections, which cause large variations in the traffic volumes
- Other failures affecting service providers, which result in unavailability of infrastructure

The survivability is the capability of the network to operate and accomplish its goal (delivering packets) within required time limits when the system is facing a failure. ITU-T Y.1720 [4] defines survivability techniques as those that enhance reliability performance of a network by providing a capability to recover from service interruption (e.g., due to defects). Fault tolerance is the ability of the network to have redundancies (strategies) to operate normally in cases of small random faults.

There are various frameworks to capture the resilience of a network. They vary on the basis of their objective as well as their definition of scope of resilience. Survivability cannot be defined without knowing the system and threats and faults being considered. Hence, survivability is context-specific. For example, Jabbar [76] proposes a general framework where survivability strategy has two levels. His strategy is called D2R2+DR. D2R2 framework consists of defend, detect, remediate and repair strategies, while a longer term DR framework detects the causes of failures and refines the protection capability of the network.

Zolfaghari and Kaudel [143] provided one of the earliest frameworks to measure survivability. Their work is on the quantification of the survivability performance of a network. They considered the causes of both random as well as the pre-planned failure scenarios (anticipated, planned-for) and proposed a multilayer survivability framework that defines the Control mechanisms, survivability needs and techniques for each layer. Using this framework, the survivability features of a network can be assessed and analysed.

Gruber [65] has also developed a qualitative framework to capture the resilience characteristics of a mechanism. An overview of resilience methods based on multipath structures is also presented by Menth et al. [89]. Their framework compares and evaluates different existing mechanisms for optical and packet-switched networks.

3.3 Protection Mechanisms

Resilience mechanisms can be introduced in a network at different layers. For example, Automatic Protection Schemes (APS) can be used in SONET net-

works. These lower layer protection schemes provide quick protection against pre-planned failures, but are inflexible towards changing situations. Higher layer protection, which is introduced at the network layer routing protocols, is much more flexible. In this section we discuss both types of protection schemes and compare their defining characteristics. It may be noted here that different protection schemes at different layers can be added simultaneously to provide a better resilient network.

3.3.1 Lower layer protection schemes

System/ transmission layer protection mechanisms require structures that are fixed and hard-wired. Once such mechanisms are in place, the operation becomes straightforward, as the failure paths are pre-known. However, this is a static arrangement. Therefore, changing the routing strategy or adapting to a different demand pattern would not be easy, as the protection is dependent on network installations and reconfiguration. Also, these mechanisms protect the whole fibre, and services such as differentiated QoS for different applications/ users would not be possible. These mechanisms are termed as pure protection methods. ITU-T G.808.1 [5] defines four protection mechanisms for end-to-end protection for packet and circuit switched networks. These are: 1+1, 1:n, m:n, $(1:1)^n$[4].

1+1 Protection

In 1+1 Protection, a primary path is protected by a second path through a bridge. Both paths carry the traffic. The starting node (ingress node) sends

[4]From protection point of view, $(1:1)^n$ method is same as that of 1:n, but it is meant for cell/packet based traffic.

traffic on both paths. The end node (egress node) will receive two copies of the flow. It will select one of the two copies. The selection is done on the basis of predetermined criteria such as arrival time, defect indication and quality of flow received. In case of a failure, only one of the two copies will arrive at the egress node. Hence this scheme provides fastest possible switching speed. Capacity requirement to provide protection is at least 100% more than the unprotected traffic, as it require two completely disjoint cables to be laid.

1:1 and 1:n Protection

This type of protection is used to protect channel or fibers between the same end points. In 1:1 protection, a primary path is protected by a backup path. It requires same resources as that of 1+1. However, during the times when this capacity is unused (i.e., the primary path is operational) the backup path is used for other low priority traffic, which can be preempted when needed. If there is one back-up path allocation for one primary path it is 1:1 protection. In order to save the spare capacity, many primary paths can share the same backup channel. Hence the capacity equivalent to the maximum of individual capacities of n paths can be reserved as backup. In case of a failure, the receiving end detects the failure and checks if the backup path is free. If so, it sends the signal to the sending node to switch the traffic on to the backup path. As this scheme requires signalling and switching operations, it is somewhat slower than 1+1 method. Furthermore, at any given time, only one channel can be protected. However, if the failures (and restoration) of all of the n paths do not overlap, all of them can be protected.

m:n Protection

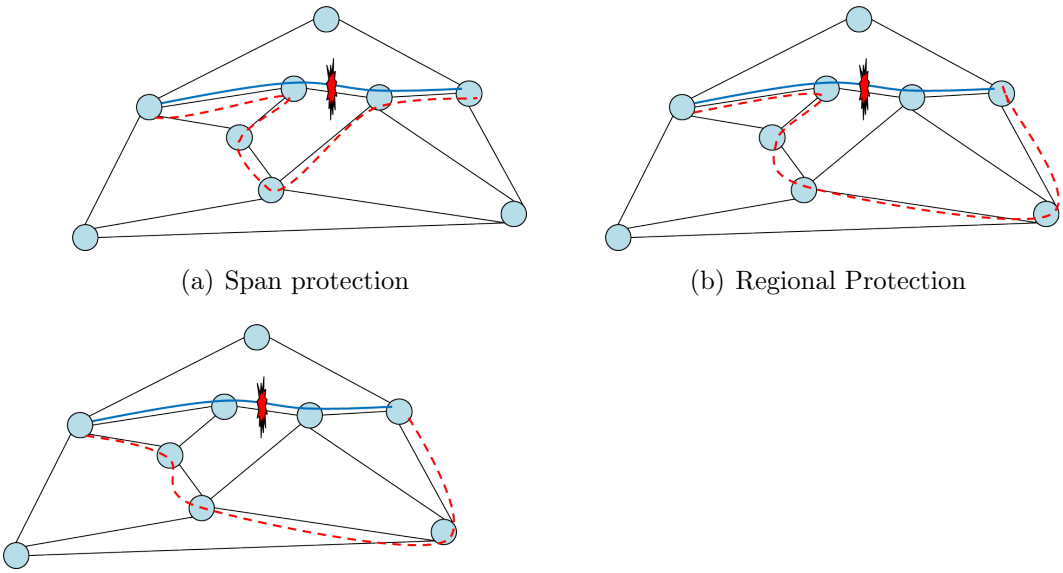
In order to improve the protection capability of the 1:n scheme, a group of n paths can be protected by m back-up paths. Hence, at any given time m paths can be protected simultaneously. Here m is generally less than n. When $n = m$, this scheme provides same level of protection as that of 1:1 protection.

3.3.2 Logical layer protection schemes

The protection can be added in layers higher than the transmission layers. These schemes are sometimes also referred to as restoration schemes, in contrast to the protection schemes implemented at the transmission layer. Such mechanisms are introduced through introduction of paths on demand, using the spare capacity available for this purpose. These restoration strategies can be to restore full path, one link or a group of links on a path.

The backup paths may have dedicated spare capacity, or they can share the backup capacity when the primary paths are disjoint (link /node or risk group). In order to enhance efficiency, when capacity is shared, then only single failures are guaranteed to be covered. ITU-T-Y.1720 describes mechanisms for protection switching for MPLS networks. These are 1+1, 1:1 and shared mesh and packet 1+1. 1+1 and 1:1 schemes are invariably the same as those described in the context of transmission layer protection, but at the logical level. Hence two separate paths are created, one to take the primary traffic and the second one acts as the backup path. The other two methods (shared mesh and packet 1+1) allow the capacity sharing in case of non-simultaneous failures. Shared mesh protection is an end-to-end path protection

Stub-release is another important feature in restoration mechanisms. It



(a) Span protection

(b) Regional Protection

(c) Global Protection

Figure 3.1: Path-based protection schemes

refers to converting the surviving capacity of the failed paths into capacity that is available for restoration purposes. Sharing of backup capacity with stub-release is of significant interest to the researchers, as it represents the "most efficient possible class of survivable networks"[64]. In the remaining part of this section we discuss different restoration schemes that make use of such sharing of capacity.

Span or Local Path Protection (SPP)

When local restoration is in place, the traffic on a protected link, in case of failure, is detoured on a backup path. This backup path has same end-nodes as that of the protected link. See Figure 3.1(a).The node immediately preceding the broken link will bypass the broken link without signalling the failure to the path's end nodes. This results in fast recovery. Disadvantages of this scheme are that it requires many more back up paths for each link. Also sometimes,

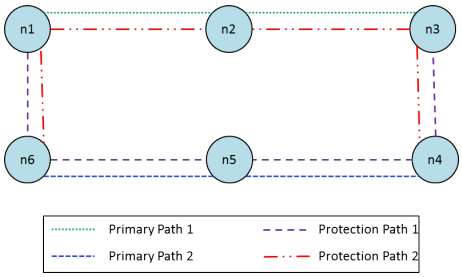


Figure 3.2: Shared Mesh Protection (SMP)

paths have loop backs. This mechanism can cover only link failures. Node failures cannot be protected. It is possible for many edges to share the same capacity.

Regional Path Protection (RPP)

When there is a risk of failure of few links on a path, a regional restoration strategy can be adopted. See Figure 3.1(b).The failure signals have to be sent to the start and end nodes of the protected region. A switching to backup path is initiated. It requires less processing time as compared to global restoration, but more than the local restoration. The spare capacity requirement is more than that of global restoration and less than that of local restoration. It can provide both link and node failure protection.

Global Path Protection (GPP)

In case of global restoration, switching to the backup path is done by both ingress and egress nodes. Signaling is required to inform these nodes of a failure, so that a restoration is activated. This would slow down the restoration. It supports protection against both link and node failures. See Fig. 3.1(c). When the backup paths share the same capacity, this method is the same as that of shared mesh protection.

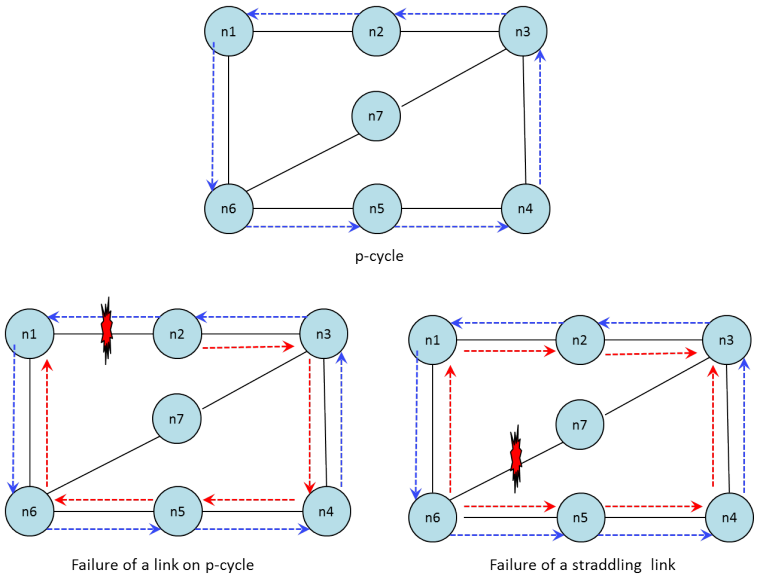


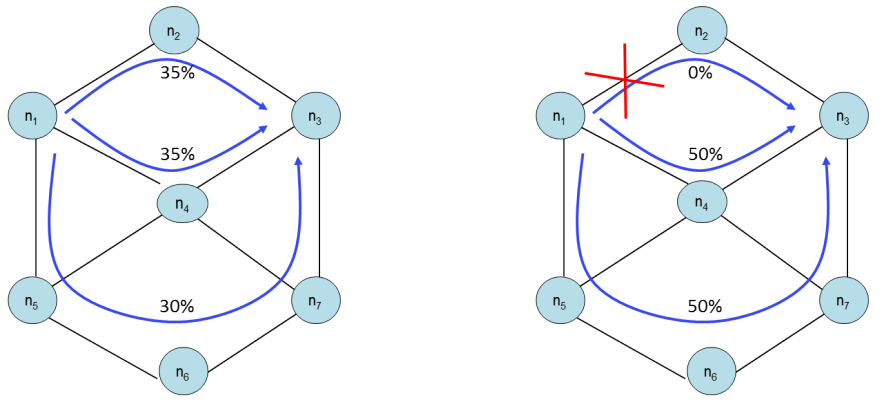
Figure 3.3: P-cycles

Shared Mesh Protection (SMP)

This approach is similar to 1:1 protection as each path is being protected by one backup path. It is also sometimes termed as shared-backup path protection. In this scheme, the backup paths share the same capacity for various paths. This reduces the spare capacity needed to provide protection. Fig. 3.2 shows an example of SMP with two primary and two back up paths. The backup path sharing is done by paths that are either link-disjoint or node-disjoint. If two paths share the same risk group, they can fail simultaneously. Therefore, they cannot share the same backup capacity.

Packet 1+1 Protection (P-1+1)

For packet-based traffic that uses end-to end paths (such as MPLS), P-1+1 protection may be used. For each flow, two paths are set up. The ingress node sends duplicate copies of each packet on both paths. Due to the transmission delay, one copy will arrive earlier than the other. The egress node accepts



(a) Multiple Paths for a demand

(b) Redistribution in case of failure

Figure 3.4: Self Protecting Multipath (SPM) scheme

the first copy and discards the second one. In case of a conventional 1+1 protection scheme, there is one dedicated primary path and the other one is backup. In case of a failure, the egress node needs a notification of a failure and switches to the backup path. In Packet 1+1 scheme, the selection of an incoming packet is independent of previous selection. There is no switching involved. Each packet has its own sequence number embedded in it and is processed independently. Therefore there is no switching delay.

Demand-wise Shared Protection (DSP)

The demand is protected by using diversification strategy (See Koster et al. [82]). The diversification strategy was proposed by Dahl and Stoer [33]. In this strategy the traffic in normal operation (no-failure) is split over multi-paths. The number of required backup paths is determined on the basis of the number of possible node-disjoint paths and number of demand units to be protected. The spare capacity is added on top of each fraction of demand on these paths. In case of a failure of one of the paths, there will still be

enough capacity for demand to flow. By dividing the demand into equal splittable units, the amount of spare capacity required is reduced. This capacity is shared only among paths for one demand. Signalling of failure has to be done to the demand source and destination. The source node will alter the load distribution on multipath. This strategy depends on the availability of node-disjoint paths.

Pre-configured protection cycles (p-cycles) A network when connected in the form of a ring, at the physical layer, is one of the efficient ways of providing protection against failures. It is the typical design feature of SONET/SDH networks. Using SONET's APS or unidirectional path-switched rings (UPSRs), or Bidirectional line-switched ring (BLSR), in case of a failure the traffic is diverted on the protection ring in the opposite direction. There is no need of failure signalling. A similar concept, called p-cycle, is implemented at higher layers even in mesh networks (Grover [64]). It provides protection against on-cycle failures. It also provides alternative routing for straddling links[5]. Backup capacity is not dedicated to any demand, and can be shared. See Fig. 3.3. It uses multipaths only in the case of straddling links.

Self protecting Multipaths (SPM)

An end-to-end protection switching mechanism that uses disjoint multipaths, called Self-protecting Multipath (SPM), was proposed by Menth et al. [88]. See Fig. 3.4. It can be implemented as an explicit routing mechanism over MPLS. In SPM, traffic is transmitted on parallel disjoint paths according to a load balancing function. In case of a failure, traffic is directed to the remaining

[5]straddling link is not on a cycle, but both of its end points are on the cycle.

functional multipaths using a different load balancing function. It has been argued that the SPM has a potential to provide protection against single node and link failures, with lesser capacity requirements than both OSPF rerouting and p-cycles (Menth et al. [90]).

Table 3.3: Framework for resilience

I Strategy	II Multipath	III Rest Time	IV Capacity Shared	V Traffic Distribution	VI TE features	VII Protection Scope
1+1	no	$< 10ms$	no	no	no	both
1:n	no	$< 100ms$	partial	no	no	both
m:n	no	$50 - 150ms$	partial	no	no	both
SMP	backup	$< 100ms$	yes	no	no	both
P-1+1	all	$< 10ms$	no	no	no	both
GPP	no	$< 100ms$	possible	preplanned	LB	both
RPP	no	$<$ GP	possible	no	LB	link
SPP	no	$<$ RP	possible	no	LB	link
DSP	all	$50 - 150ms$	partial	pre-planned	no	node
ECMP	all	$< 100ms$	yes	static	LB	both
OMP	all	several sec	yes	in real time	LB/CC	both
p-cycle	backup	$< 100ms$	partial	pre-planned	no	link
SPM	all	$< 100ms$	yes	pre-planned	LB/CC	link

3.3.3 Re-routing

Re-routing is the recovery mechanism in which recovery paths are calculated dynamically, after the occurrence of a failure, as opposed to the protection switching, where paths are pre-known. OSPF uses re-routing strategy for failure recovery. Restoration paths are not pre-established. In case of a failure, the forwarding paths are re-calculated, which increases the recovery time. We briefly explain two relevant extensions to OSPF that make local decisions in case of failures.

Equal Cost Multipath (ECMP)

In OSPF Hello mechanism is very slow and the network may take up to several seconds to converge. Equal Cost Multi-Path (ECMP) [72] in OSPF is used to bifurcate traffic among two equi-cost shortest paths. This distribution may result in over-utilization of some links, while others remain under-utilized. ECMP is a way to allow nodes/routers to take local decisions. If there exist two equal cost shortest paths, then in case of a failure of one of them, the forwarding node can take a local decision to switch the traffic to the alternative active shortest path. However, finding a weight system that can be used to calculate the equal cost shortest paths is a very difficult problem. The node degree might also restrict the availability of two such paths. Because of these reasons ECMP feature, despite being available in most of the routers, is not very often used.

Optimized Multipath (OMP)

One variation of ECMP is Optimized Multi-Path (OMP) [130]. OMP allocates optimal (instead of equal) fractions of traffic to two shortest paths. The

load distribution is based on the traffic on all the forward links. The problem with OMP is that it adapts very slowly to load changes in the network, and sometimes it may take hours [47, 131].

3.3.4 Comparative analysis

A comparative analysis of all of the protection strategies discussed above is presented in Table 3.3[6]. Column II of Table 3.3 shows whether a multipath capability is possible in the failure protection strategy. Approximate restoration times are given in Column III. The capacity sharing feature, if available, is mentioned in Column IV. The traffic distribution on the multipaths can be either pre-planned or static. Column V shows this feature. For example, in the case of ECMP, a static traffic distribution is applied, as the traffic is always divided equally, while in the case of DSP, it is pre-planned. Column VI shows whether the scheme also offers other traffic engineering features such as load balancing (LB) or congestion control (CC). Finally, Column VII shows the scope of protection, which can be a link or a node or both.

Except for the Packet-1+1, all strategies mentioned here are path-based. Most of the strategies allow backup paths to share the same capacity, which results in efficient capacity utilization. In case of DSP, sharing is allowed only among paths of a single demand. This sharing results in reducing the additional capacity required to protect against failures. However, it also limits the number of paths that can be protected at any single time.

Stub-release, i.e. utilizing the surviving capacity of failed paths, is employed in the case of SMP, GPP and SPM. In case of a failure, many strategies use demand splitting on multiple back-up paths. This splitting ensures efficient

[6] Adapted from Menth et al. [89]

spare capacity utilization. Strategies such as ECMP, OMP and SPM always employ demand splitting. However, other logical layer restoration schemes can also benefit from such splitting of demands. Splitting can give rise to the issue of re-ordering if one of the paths is much shorter than the other.

If demand splitting is used, then the distribution of traffic on the multipaths is another issue to deal with. In case of ECMP, the distribution is static, as it always divides the traffic in equal parts. In most of the other strategies, a pre-planned traffic distribution is known to the routing nodes. In case of OMP, distribution is determined in real-time, which makes it extremely slow. With the sharing of capacity and the use of traffic distribution, the protection strategies aim to provide traffic engineering features such as load balancing. Techniques such as SPM and OMP also provide some form of congestion control.

The path-based schemes are clearly the better options for survivability. The important characteristics that distinguish one routing strategy from the other are:

Sharing: If a backup path is dedicated to a specific primary path, sharing of capacity is not possible. However, if the same capacity can be allocated to different paths in different failure scenarios, sharing of resources from a common pool of spare capacity is possible. Almost all of the path-based approaches use sharing of capacity.

Multipaths: Diversification limits the maximum loss faced by any demand as a result of a link failure. In DSP, for each demand, a little more capacity than the specified demand value is allocated, so that in any failure scenario a specified fraction of demand always survives, without a need for re-routing. The traffic distribution on the multiple paths is fixed. However,

if we allow a redistribution of traffic in each failure scenario, that is, all the paths are allocated different loads, independent of other failure scenarios, then a *global re-routing* takes place.

Stub-release: When one of the links fails, it results in a failure of one or multiple paths for different demands. If in a failure scenario, the surviving capacity of the failed paths becomes available for routing, such mechanism is called stub-release, which can only be available in shared protection schemes.

Failure-dependency: If the redistribution of traffic is allowed only for the paths that are affected by the failures, then, in the case of failure dependence, for each failure scenario a different traffic distribution is identified. Hence, a *restricted reconfiguration* of flows is determined for the paths that are affected by the failure. However, if a failure-independent strategy is adopted, then the affected flows are always restored in the same manner.

DSP uses fixed traffic distribution and shares the capacity only within one demand. So, it is a combination of shared and dedicated protection strategies. The stubs are released within the demand, and can only be re-used by other paths of the same demand. Diversification is used. But traffic redistribution is fixed in different failure scenarios.

SPM, on the other hand, releases the stubs and allows the sharing of freed-up capacity among all demands. It also uses diversification of normal flows. In case of a failure, SPM has both variations, i.e. failure-dependent and independent restoration.

Since the possible number of paths for each demand can be very large, generally pre-determined path sets are used. Both DSP and SPM use pre-determined paths and distribute flows on these available paths. However, determination of an appropriate path set that has mutually disjoint paths is

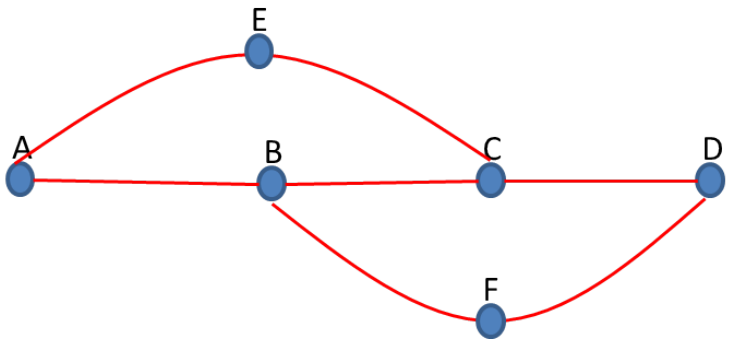


Figure 3.5: Disjoint path selection

a non-trivial problem. If a wrong path is selected, it can preclude the optimal paths. If a network has the structure which is shown in Fig. 3.5, with three equal-length paths, then there are two disjoint paths between the node-pair (A,D) – (A,E,C,D) and (A,B,F,D). However, if the path (A,B,C,D) is selected, then there are no alternative disjoint paths.

Orlowski and Pióro [103] discuss the path-based formulations for survivable network design problems that use different combinations of the above described characteristics. They use sets of candidate paths for each node-pair. The design problems are aimed at minimizing the costs. Where path selection is required, (primary, backup) path-pairs for each demand and the fraction of load allocated to each of these path-pairs is determined. Gruber [65] has also used a similar strategy in his MILP for survivable design problem.

We use SPM as a starting point, since it uses diversification as well as sharing of capacity among all demands. A similar strategy has been used by Gruber [65]. However, the main difference between his work and ours is that he has worked on a network design problem and his objective function is that of cost minimization, while we are looking at minimizing the worst-case link utilization. As far as the path selection strategy is concerned, Gruber defines

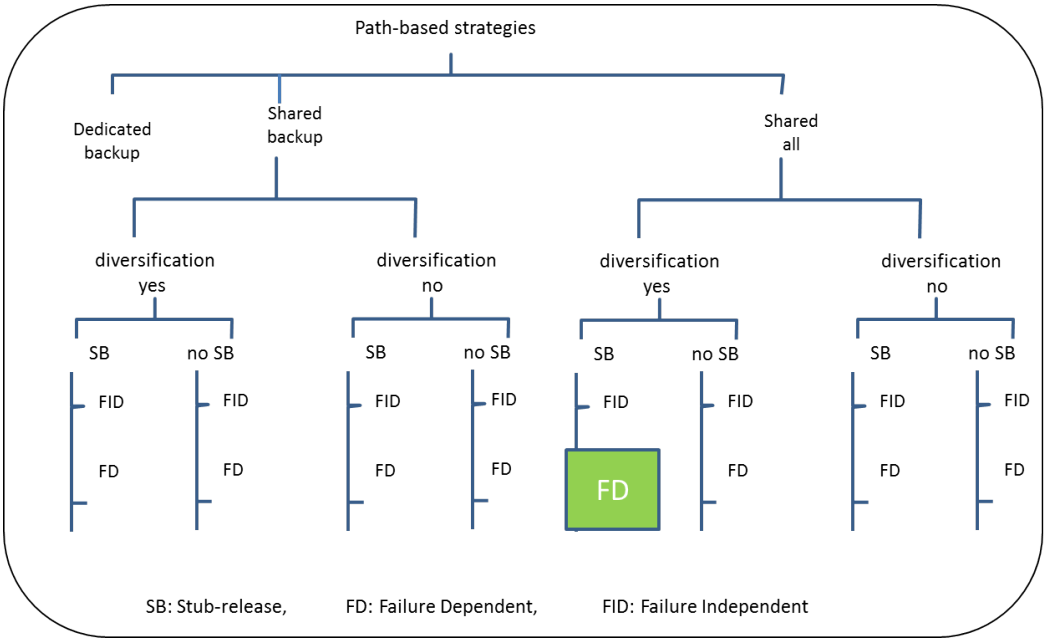


Figure 3.6: Classification of protection strategies

a very large number of path sets for each demand, i.e., for each primary path he defines one path set corresponding to each failure scenario. Hence he predetermines a set of disjoint paths. We, on the other hand (as will be shown in greater detail in Chapter 5), use only one path set for each demand. The disjointness of the paths is ensured with the help of suitable constraints.

To the best of our knowledge, the path selection problem with load balancing objective for multipath survivable routing has not been studied before; and in this thesis we will present novel formulations and heuristic algorithms for this complex problem.

With reference to the four characterizing features described above, our work relates to the path-based restoration strategy, where all paths (primary and back-up) use sharing of capacity, apply diversification (multipaths) and allow re-distribution of traffic only in the failure scenarios that affect the demands. See Fig. 3.6.

3.4 Summary

In this chapter we have looked at the issue of resilience both from management and technical point of view. We have given a brief overview of organizations managing the standardization of networks, in general. We have also looked at how the survivability and resilience of a network are defined and quantified.

We have given an account of various protection, restoration and rerouting strategies used within telecom networks and have discussed their defining features. Based on the evaluation of different restoration strategies, we have identified the most promising area of research in this context. While looking at the existing literature, we have identified a gap – which we intend to fill through novel MILP formulations and heuristic algorithms. The MILP formulations will be discussed in Chapter 5 and the heuristic algorithm will be discussed in Chapter 7. Before that, however, in the next chapter (Chapter 4) we discuss how heuristics have been used to tackle the issue of routing. We will use this survey to guide our choice of heuristic methodology, for the reasons that will also be explained.

4

Heuristic Routing

When heuristic approaches are used, we cannot guarantee the optimality of the solution. But in a situation, where the problem size is so large that it is computationally infeasible to solve it exactly, heuristic solutions offer a good alternative. Even a good lower bound obtained with heuristics can be useful. From a managerial point of view, this approach provides a realistic and worthwhile insight into the problem at hand, which can facilitate the decision-making process. As we will see later in Chapters 5 and 6, the size of our routing problem increases very quickly with an increase in the network size. Therefore, in ad-

dition to the exact methods, we have developed *hybrid* methods (presented in Chapter 7), where we mix the exact and heuristic approaches. The adoption of this *matheuristic* methodology has obliged us to trace its antecedents; and in order to place our hybrid algorithms within this new matheuristic framework, we consider it helpful to discuss the prior heuristic research that might be relevant to our own research agenda. Accordingly, in this chapter, we take cognizance of the current state of research in the area of applications of heuristics for network routing.

The use of heuristics in network routing is not new. Some research has been on-going since the 1990's. The inherent adaptiveness of the heuristic methods make them a good candidate for providing solutions to the ever-changing nature of networks' traffic.

4.1 Heuristic Techniques

Heuristics is a general term for methodologies that find good-enough (not necessarily optimal) solutions. Among these heuristic techniques, a new class of techniques is further categorized as meta-heuristic methods. Meta-heuristics refer to methods where improvement in solutions is arrived at through iterations (of heuristic or exact algorithms) and with a the view to avoid local optima and find very high quality solutions. Meta-heuristics can be applied to a wide set of different optimization problems [38].

A heuristic algorithm can start with partial solutions and build the solution iteratively by adding another component to the partial solution, thus using a constructive approach. For example, starting from one city in the example of TSP, a solution can be constructed by adding cities one by one

to the partial solution. Alternatively, these algorithms can be based on local search. For these algorithms, an initial solution is a pre-requisite. Starting with an initial solution, these methods try to improve the objective value through local improvements. In local search algorithms, the concept of neighbourhood structure is used and the choice of these structures affects the performance of these algorithms. If a set of solutions for a problem is defined as S, then for every $s \in S, N(S)$ is a set of solutions that can be reached by making local changes to s. It would require a single step in the algorithm to move from s to one of its neighbours.

The meta-heuristic methods generally use iterations of either constructive or local search approaches to find solutions. Sometimes neighbourhood moves are used as well to do the improving solutions. At each iteration, some of these methods work on a single solution, while others manipulate a set (called population) of solutions. The use of memory is another distinguishing characteristic. Some meta-heuristics memorize their experience in one iteration and use it to direct the future search, while the others do not store information at each iteration. In Table 4.1, a brief summary of heuristic and meta-heuristic methods discussed in this Section is provided. Table 4.1, Column III shows if the method is constructive (C) or local search(LS) based. Column IV shows if the method manipulates single (S) or population (P) of solutions at each iteration. Column V describes whether the method uses memory to direct its future search and lastly, column VI provides references to the original contributors of the technique.

An important aspect of heuristic methods is to determine when to stop any heuristic algorithm. There is no set rule and in practice different criteria are used. Examples of such criteria are the total number of iterations, number

of iterations without improvement, time elapsed, pre-determined deviation from known upper/lower bounds.

Table 4.1: List of heuristic and meta-heuristic methods

I Method	II Type	III C/LS	IV S/P	V Memory	VI Reference
Greedy	Heuristic	C	S	no	[30]
Local Search	Heuristic	LS	S	no	[6]
GRASP	Meta-heuristic	C	P	yes	[110]
TS	Meta-heuristic	LS	S	yes	[60]
SA	Meta-heuristic	LS	S	no	[79]
VNS	Meta-heuristic	LS	S	no	[91]
EA	Meta-heuristic	LS	P	no	[62] ,[71]
ACO	Meta-heuristic	C	P	yes	[39]

4.1.1 Greedy Algorithms

Greedy algorithms create a solution from scratch by selecting elements incrementally from a given candidate set. All elements are checked for selection in the partial solution. The most attractive element w.r.t. the greediness measure is selected. This algorithm terminates once a complete solution is generated. Since the algorithm does not make an extensive search, the quality of solution is often not good. But it is a good method to quickly create an initial solution, which can be used by other algorithms as a starting point.

4.1.2 Local Search

The local search algorithm starts with one candidate solution. It searches the neighbourhood of the current solution for a better solution w.r.t. a certain predefined criterion. neighbourhood is defined as solutions which are one move

away. A move is defined as a small change in the elements of the solution, such as flipping of a bit in the solution string. During the neighbourhood search, if a better solution is found, it is picked as the next candidate solution, and its neighbours are searched. The search process continues until no further improvement in solution value is possible or the time lapses.

There are two strategies to select a new incumbent solution. The first-improving solution strategy searches the neighbourhood of a current solution only until a solution (if any) that improves the current solution is found. The second strategy is the best-improving solution strategy that searches for the best-so-far solution in the neighbourhood of the current solution. This strategy requires more computational time as compared to the first-improving strategy.

The local search algorithm is prone to being trapped in local optima. A number of meta-heuristic methods (such as Tabu Search and Variable neighbourhood Search) that are based on local search, adopt different ways to avoid this problem.

4.1.3 Greedy Randomized Adaptive Search procedure

Greedy Randomized Adaptive Search (GRASP) is a meta-heuristic that combines greedy algorithm with local search [110]. It has two phases: construction and local search phase. The construction phase uses a greedy algorithm. In this phase, the selection of each component of the partial solution is randomized. At each step, the solution components are ranked according to some greedy function and then best x% ranked components are added to a candidate list. The algorithm then randomly selects the next component from this candidate list. Once a complete solution is formed, local search is applied to

look for further improvements in the solution value. The local search stops when a local minimum is found in the neighbourhood.

This two-phase process is repeated for a pre-determined number of iterations. In each iteration, if an improved solution is found, it is preserved.

4.1.4 Tabu Search (TS)

The main drawback of a local search method is that it can easily get trapped in the regions of local optimum. Glover [60] developed the idea of Tabu Search to overcome this problem. In Tabu Search, a small list of solutions is maintained which are forbidden from selection (hence the term tabu). Tabu list contains solutions based on user-defined rules and/or previously selected solutions. The search starts with a random valid solution. During the neighbourhood search, if a better solution is found, which is currently not in the tabu list, it is selected as next point of search. Tabu list is updated and search continues from the newly selected solution. If no improving solution exists in neighbourhood, the solution with minimum deterioration to the current solution value is chosen as the next solution. This helps in escaping local optimum regions or plateaus with same fitness value. The search continues until a stopping criterion is met. The size of the tabu list is fixed. When the list is full and a new entry needs to be made, the oldest entry is removed from the list. There are various variations of Tabu Search procedure to speed up the search process. See Glover and Laguna [61] for more details.

4.1.5 Simulated Annealing (SA)

Based on an analogy to the physical annealing process, the idea of SA was developed by Kirkpatrick et al. [79]. Annealing is a process in which solid metal is given a heat bath, and as the temperature goes down, the cooling process is slowed down. Resultantly, the metal reaches its low energy state and forms a perfect lattice. Analogously, a higher probability to accept the worse solutions is used at the start of the algorithm, and during the course of the algorithm this value is reduced step by step.

The main algorithm works in two nested loops. At the beginning, a very high temperature value T_0 is selected. The inner loop is a local search algorithm. The selection of a new solution depends on the value of the temperature parameter set for the current iteration. During each iteration of the inner loop, if a better solution is found, it is selected. Otherwise the best available solution is chosen with a probability according to some distribution. Metropolis distribution is often used to determine the probability of acceptance. If $f(s)$ denotes the current solution and $f(s^{'})$ represents the best solution among the neighbours, then the probability of acceptance can be calculated as $e^{f(s)-f(s^{'})/T}$. The probability to choose the neighbour solution is 1, if the neighbour solution is better than the current solution. The inner loop where one temperature setting is used, continues until the stopping criteria are met. After that the temperature value is reduced in the outer loop and the inner loop is initiated again. Depending on the right choice of parameters, SA algorithm may converge quickly. SA only stores the best solution found and no other information from previous runs is used to guide the future search. So, the memory requirements for SA algorithm are almost nil.

4.1.6 Variable neighbourhood Search (VNS)

Unlike local search, where only one neighbourhood structure is used, VNS uses a pre-defined set of finite neighbourhood structures. (Mladenovič and Hansen [91].) It works at two levels. The outer loop, starts with the first neighbourhood, and continues until the stopping criterion is met. The inner loop starts with a local search of neighbours of a perturbed solution s'. s' is randomly selected among neighbours of the current best solution s. The randomization helps avoiding cycles. The local search looks for a first-improving or best-improving solution using the k^{th} (staring from the first one) neighbourhood structure. If an improving solution is found in the current neighbourhood, local search will begin using the new found solution, again starting from the first neighbourhood structure. However, if no improving solution is found, k is incremented and the next neighbourhood is searched. The systematic change in the neighbourhood structures helps to come out of the regions of a local optimum.VNS does not store information from previous iterations, except for the best solution so far, and is a memory-less algorithm.

4.1.7 Evolutionary Algorithms (EA)

An Evolutionary Algorithm is based on the process of natural evolution. There are three variants of EA, genetic algorithms being the most popular one. Here we discuss this variant in detail.

Unlike other meta-heuristic methods, an EA is a population-based method, i.e.: it starts with a set of individuals, called population. An individual represents one possible solution. Individuals are generally encoded as a sequence of discrete, binary-valued or real-valued variables . An initial population is

either created randomly or by using one of the heuristic approaches. The concepts of selection, mutation, crossover and fitness have been borrowed from the evolutionary process in nature. Cross-over is the creation of a new individual (child) by combining two existing solutions (parents). As in nature, an individual sometimes goes through the process of mutation: that is, a small change in its genetic structure. Similar transformation is possible in the EA. For example, if an individual consists of a string of binary integers, flipping of one of its bits might be considered an example of mutation. Selection of individuals for crossover and mutation is done stochastically, based on their fitness value. The fitness function determines the quality of an individual w.r.t. some objective. The higher the fitness value, the better the individual. Higher fitness value means higher probability to survive in the next population.

Starting with an initial population, an EA applies crossover and mutation operators on few randomly chosen individuals from the initial population. The result is a newly created individuals. A selection operator (using the fitness function) is applied to select individuals from the old population and these newly created individuals. The newly formed population is then fed to this algorithm again. This evolution process continues until a stopping criterion is met.

The performance of an EA critically depends on the choice of representation for the population and the design of crossover and mutation operators.

4.1.8 Ant Colony Optimization (ACO)

Dorigo et al. [39] presented the idea of Ant Colony Optimization. ACO belongs to a broader class of swarm intelligence, where foraging behaviour of swarms of

ants, bees and bacteria are studied and are emulated in heuristic algorithms.

Ants are natural agents for finding the shortest path. They use stigmergy – i.e.: communicate by making changes in the nature. This is an indirect way of communication. The change made by one ant is used by other ants while choosing their paths. They do so by laying different quantities of pheromone.

ACO is a constructive heuristic: i.e., a solution is constructed by incrementally adding small components to a partial solution. Any combinatorial optimization problem that can be represented as a constructive heuristic is a potential candidate for ACO application. However, formulating a problem so that it is solvable by ACO method is not always easy. Some problems lend themselves naturally to ACO applications, e.g., TSP and network routing, where ant like behaviour to find the shortest paths can be emulated. To apply ACO metaheuristic, the problem needs to be represented in the form of a fully connected graph of components, called construction graph. The components are linked to each other through connections. Different sequences of these components (for example cities in the case of TSP) define states of the problem. Usually only valid solutions are constructed. The validity depends on satisfying the constraints of the problem. These valid solutions thus constitute a set of candidate solutions. For each candidate solution, there is a cost associated with it. Among these solutions, there will be one or more optimal solutions. Ants are implemented as a stochastic procedure. Using this representation, ants traverse a a completely connected graph of the components.

The ACO algorithm is generally composed of iterations of three tasks: (i) constructing ant solutions, (ii) updating pheromones and (iii) performing centralized daemon actions. The solutions are constructed by generating a

colony of ants. Ants, while traversing a completely connected graph of the components, choose next states to visit by using the pheromone trail and heuristic information available on each visited node. Once they have no more feasible hops, they evaluate the solution and the best (or the elite) ants update the pheromone trail. A set of centralized actions, such as evaporation, also takes place, which may also result in updation of these pheromone tables. The order of these tasks is dependent on the optimization problem in hand. In case of network routing, ACO is applied in a decentralized fashion. Hence, more than one ant is allowed to updated the pheromone tables.

4.2 Heuristics for Network Routing

The routing algorithms for telecom networks/ the Internet have been traditionally based on packet-switched methodology. These algorithms have a global view of the network and are deterministic in nature. Routes are determined generally through Dijkstra's or Bellman-Ford shortest path algorithms, which require a global network view for execution. This network information is acquired through neighbouring nodes. The updating of this information is not only slow but also has traffic and space overheads and consumes a lot of effort in synchronizing /updating network information at any given instance. Network resources are therefore underutilized. Routing is on single paths and usually algorithms do not maintain multipaths between a node and the destination. Few initiatives have been published where routing uses multi-paths, e.g.: Chen et al. [24] and Vutukury and Garcia-Luna-Aceves [133]. A review of multi-path based algorithms was done by Vutukury and Garcia-Luna-Aceves [134]. However, a single-path approach, such as OSPF, adopted by the net-

working community, has an obvious limitations (pointed out in Chapter 2) from an optimization and multipath self-correcting perspective.

A survey done by Wedde and Farooq [136], focused entirely on telecommunication networks and provided a perspective based on the design philosophy of the development of the routing algorithms coming from nature inspired methodologies. They argued that nature inspired methodologies, such as EA and swarm intelligence, are in actual fact the off-shoots of AI. They also compared the performance of heuristic methods with exact (deterministic) methods of network routing. The nature-inspired stream of research, focused on the methods of artificial intelligence. Both machine as well as agent-based learning were employed. The resulting algorithms were adaptive and dynamic in nature. They were decentralized and made use of local information and the routing decisions were deterministic. The agent-based systems were autonomous and the agents left the network information on the nodes, which was then used to make routing decisions.

These heuristic algorithms do not guarantee optimality, They find a good (near optimal) solution in a reasonable time when the network size gradually increases. On the other hand, the performance of exact algorithms deteriorates with the increase in an size of network. In this Section, we present a brief overview of advances in network routing using EA, ACO and other local search based meta-heuristics.

4.2.1 Evolutionary Algorithms for Network Routing

In 1999, Sinclair [120] provided the first summary of the application of evolutionary computations (EC) to telecommunication networks. He did break-

down the problem into six elements: dimensioning, node location, topology, trees, routing/ restoration/ call admission and wavelength/ frequency allocation. It appears from his review that routing, restoration and call admission had received the greatest attention (36 papers), to be followed by papers on dimensioning of the telecommunication networks (27 papers).

In 1997, Munetomo et al. [95] proposed a genetic algorithm called "path genetic algorithm" for adaptive network routing. The term adaptability was used to emphasize the fact that the algorithm could handle large networks, handle uncertainty and work in parallel on different processors. They proposed three algorithms, one for circuit switched networks, a fuzzy classifier system for packet -switched networks and the third algorithm called Genetic Adaptive Routing Algorithm (GARA), which was meant for Internet based traffic. Here we describe GARA is detail which has formed the basis for other EAs for routing.

GARA determines alternative paths for Internet traffic. The algorithm is distributed in nature and does not require central control. It uses all the three genetic algorithm operators, i.e.; crossover, mutation and selection. The crossover operator plays the central role. This scheme adaptively maintains a known number of alternative paths/routes by using the network information. This information is kept only for the frequently used destinations. For each route, delay is calculated by sending a test packet. Based on the delay, weight is assigned to each alternative path. The higher the delay, the smaller the weight assigned to that path. New routes are found by using crossover and mutation operators.

Crossover takes two paths and uses one of the common nodes as a crossover point. This operator cannot be applied if there is no common node.

Mutation operator takes a candidate node in a route and selects one of its neighbours randomly. Then, the shortest path is found from this neighbour-node to source and destination nodes, using Dijkstras algorithm. If the route contains duplicate nodes, the new route is discarded. Otherwise, the newly formed (after mutation) route is included in the routing table and its fitness is evaluated– along with other routes. Fitness evaluation is done on the basis of the delay in the route. When a new route is to be inserted and the routing table is full, the selection process removes the lowest-weight path.

If a packet arrives with a destination that was not used previously, the routing table will have no *a priori* routes for this destination. The first route to this new destination is calculated using Dijkstra's shortest path algorithm, and is added to the routing table. The other alternative routes are generated by using genetic operators.

Compared to the other protocols used over Internet such as RIP (Routing Information Protocol) and SPF(Shortest Path First), GARA has very little operational overheads. RIP and SPF send $O(n^2)$ messages, while GARA sends O(n) messages with O(1) size – as compared to O(n) for RIP and O(1) to O(n) for SPF. Hence the total overheads for GARA are O(n) – while RIP is $O(n^3)$ and SPF is $O(n^2)$ to $O(n^3)$.

Based on GARA, Munetomo et al. [96] later also proposed a migration scheme GRM – Genetic Routing algorithms with Migrations. It is meant for parallel genetic algorithms. Instead of re-calculating all the paths, a source node can get routes migrated from other subpopulations, if routes going towards the same destination have already been calculated previously. The addition and deletion operators are applied to make the migrated string meaningful for the receiving node.

Liang et al. [84] later tried to improve upon GARA and proposed Distributed Genetic Algorithm (DGA). They proposed a network routing scheme, where nodes do not have information regarding the network status and do not need to maintain the extensive route details. Only the source node's routing tables are updated. They have proposed GA-agents for this purpose. The crossover, mutations and fitness evaluation criteria are also designed. The GA-agents are chromosomes, in which each element depicts the next hop in the path. Each node maintains a set of GA-agents. The next hop information is within the GA-agents, hence it does not need any information from the node it is visiting. Hence this algorithm differs from GARA, where each node specifies the next hop for the path. GA-agents make a return trip - like ants in AntNet. When GA-Agent completes a round trip, only the routing tables at the source node are updated. To reduce the processing, if two neighbouring nodes share a part of the route, useful chromosomes are passed to neighbouring nodes. The neighbour node applies deletion, addition operator to make the received chromosome valid.

For IP routing optimization GAs have been proposed in [93], [111], [109], [46],[94], [112] and [135].

4.2.2 ACO based heuristic solutions

The collective intelligence of groups of fish, insects and other biological organisms has always inspired researchers. Bonabeau et al. [15] were among the first researchers who used this Swarm Intelligence (SI) for problem solving. The fundamental characteristic of SI is the absence of a central controller. These algorithms are decentralized and can adapt to the changes in the system

(nature) and make probabilistic, rather than deterministic decisions. Among these SI meta-heuristics, ACO has been widely used to propose adaptive routing for the networks. Although ACO is a centralized meta-heuristic, it acts in decentralized manner when applied to network routing.

The network information such as data traffic and network topology change over time. The routing on the network has to adapt to these changes. This problem can be represented in ACO paradigm. The components are the nodes of the network and connections are the links between different nodes. The nodes store pheromone tables and routes to each destination. The time taken to complete each route is also stored.

The ants choose the next hop stochastically, based on the probability data available in the pheromone table of the node visited most recently. This way, ants may choose different hops each time and can explore multiple paths. The repeated path sampling mechanism is used to update information about the state of the network. The algorithm is based on the concepts of positive feedback and cooperative behaviour.

Theraulaz and Bonabeau [124] discuss the strengths and weaknesses of ACO and describe a possible stagnation. This may happen if the ants initially choose a non-optimal solution and then it is reinforced strongly and adopted by all the following ants. If, however, a shortest path is found by the ants, then that path may become congested with overuse and may not remain the optimal selection. To overcome these problems, different strategies have been suggested. Sim and Sun [119] have discussed some of these strategies, such as: evaporation, aging, limiting and smoothing pheromone, pheromone-heuristic control and privileged pheromone laying.

A summary of ACO based methods is given in Table 4.2. Column II

gives the year of publication, column III describes if the method is for circuit-switched or packet-switched network. Column IV shows if the load balancing feature is proposed in the algorithm and column V shows if multiple paths can be used for one demand.

Table 4.2: Summary of meta-heuristic methods using the concept of Ants

I Method	II Year	III CS/PS	IV LB	V MP	VI Remarks
A.B.C [114]	1996	CS	yes	no	-
ASGA [138]	1998	CS	no	yes	-
SynthECA [137]	2000	CS	yes	no	fault detection
MACO [118]	2003	CS	yes	yes	disjoint paths
AR [121]	1997	PS	no	no	-
AntNet[36]	1999	PS	yes	yes	-
CAF [69]	1998	PS	yes	no	-

The routing algorithms based on ACO for circuit-switched and packet-switched networks are explained in the following Sections.

Routing for circuit-switched (connection-oriented) networks

Schoonderwoerd et al. [114] proposed a routing algorithm called Ant Based Control (ABC) for load balancing of circuit-switched networks in 1996. The ants are periodically forwarded to randomly selected destinations. They are assigned an age. At each hop, the age of the ant is increased proportional to the usage of the node they are visiting. The ants update the pheromone table using the aging information. The amount of pheromone laid shows the quality of a path from the current node back to the source node. The ants are killed as they arrive at the destination.

Ant System Genetic Algorithm (ASGA), introduced by White and Pagurek

[138] was also meant for circuit switched networks. It works for single as well as multi-paths. It combines the ants and genetic algorithm methodology. Initially a population of ants traverses the paths with random values of pheromone and after each hop updates the path costs. On the way back, the pheromone tables are modified. The paths found are stored in another table along with the values of pheromone and costs. Path selection, mutation and crossover for the next generation are done. The paths with low fitness values survive for the second generation of the ants. In order to prevent stagnation, pheromone evaporation, heuristic pheromone control and privileged pheromone laying methods are used.

Unlike Ant Based Control (ABC), ASGA does not provide load balancing. Its variation called Synthetic Ecology of Chemical Agents (SynthECA) supports load balancing (White [137]). SynthECA framework uses different types of agents, which are like ants. Each one has one of the designated tasks i.e.: finding routes, setting up connection or finding faults in the links. These agents have associated chemical features, which make changes in the environment. These agents also go through evolutionary changes. A detailed description of this algorithm can be found in [137].

The Multiple ACO (MACO) paradigm proposed by Sim and Sun [118] is for finding a routing with balanced loads. For this purpose, MACO maintains multiple probabilistic routing tables. SynthECA and ABC use a single probabilistic routing table and if there are more than one optimal paths, only one will be chosen. On the other hand, MACO uses multiple probabilistic routing tables and supports multiple paths for the same source and destination. In order to make paths disjoint, MACO uses colonies of Ants and colours them and introduces the concept of 'repulsion' between colonies. If one colony

chooses a path, then the other colonies will be repulsed by using the same path.

Routing for packet-switched (connectionless) networks

Since 1992, when Dorigo first introduced ACO, its different variations have been proposed for load balancing and routing. AntNet was one of the first mechanisms that was based on ACO. See Ducatelle et al. [41]. Most of the other methods proposed extend AntNet's methodology. AntNet uses the shortest path characteristic of the ACO paradigm. See Di Caro [36]. To explore the paths between each source and destination, artificial ants are created. These ants travel from source to randomly selected destinations. Each node maintains a routing (pheromone) table and traffic statistics. The routing tables store the shortest path from a node to any destination. The decision to use the next hop is probabilistic, which is based on the pheromone quantities and heuristic function. This heuristic function takes into account the queue length for each outgoing link. The ants, while moving towards their destination, capture the path statistics, such as the delay experienced. Once they reach the destination they become backward ants and trace back the same path. On their way back, using the time taken to travel on the path, they update the traffic statistics and pheromone table on each visited node. The data packets also use these pheromone values and the next hop is selected stochastically. This allows the data traffic not only to adapt to the changes in network status, but also to balance the traffic on the network. All the packets to one destination need not follow the same path. Different packets may use different paths. AntNet has been tested on network sizes such as: 8,13 and 57 nodes.

Sim and Sun [119] provide a comparison between ACO, OSPF (link state algorithm) and RIP (vector-state algorithm). For both OSPF and RIP, a complete routing table is required. In case of OSPF, this information is collected by sending Link State Advertisement (LSA) packets. In RIP complete routing tables are exchanged. On the other hand, AntNet requires less processing as well as storage space. The ants explore paths without any prior information, and are comparatively smaller and can use data packets, when available, to piggyback on them.

Using the idea of ABC for circuit-switched networks, Subramanian and Druschel [121] proposed Ants Routing (AR) algorithm for packet-switched networks. Heusse et al. [69] proposed the cooperative asymmetric forward (CAF) algorithm for routing, for networks with asymmetric link costs. While traveling from one node to another, the data packets leave a time stamp, i.e. time taken to travel on the link between these two nodes. The backward traveling ants, when passing through these nodes, will use this information to update the pheromone table. Boyan and Littman [17] proposed an approach to incorporate the link and node failures. The pheromone table maintains a probability distribution to select the neighbouring node. In case of a failed node/link, the probability of the failed element is set to zero.

4.2.3 Local search meta-heuristics

In this Section we discuss the use of meta-heuristic to solve various well-known problems which are related in one way or another to the network routing.

Design and route allocation of an IP network

The network design problem has been dealt with through different search meta-heuristics. Using simulated annealing, Randall et al. [108] solved the problem to find the minimum cost flow for a given demand matrix. Different variations of network design problems have been dealt with Tabu Search meta-heuristic. For example Cox-Jr and Sanchez [31] solved the design of a wireless telecom network, under survivability and capacity constraints. Girard et al. [58] solved an access network design problem that uses SONET channels and ADM equipped nodes. Xu et al. [140] solved a telecom network design problem where the alternative paths can change every hour. Other techniques used for the design problem were GRASP and Variable Neighbourhood Search (VNS), which were used by Canuto et al. [20]. Gabrel et al. [55] used greedy heuristics for discrete cost network optimization.

Gabrel et al. [55] provide a comparison of heuristics for the fixed cost network design problem. Chamberland [23] solve the problem for two-level network design to find an optimal topology for IP over SONET network and then find an appropriate weight system for OSPF routing. They have used different neighbourhood search strategies. The main problem is divided into smaller problems which are solved with TS. Nucci et al. [99] propose a TS based algorithm for the design of fault-tolerant logical topologies in WDM networks that use OSPF routing.

Routing and Wavelength Allocation (RWA)

For problems that relate to routing, wavelength or frequency assignment various meta-heuristics have been used. Kim et al. [78] proposed an SA algorithm

to allocate cells to the mobile radio system with minimum average blocking of the system. Hao and Galinier [66] proposed a TS algorithm for frequency assignment in mobile radio networks. A GRASP algorithm was proposed by Prais and Ribeiro [106] for traffic assignment in TDMA communication systems.

Optimizing link weights for IP routing

For OSPF routing protocol, a weight system for links is required that can be used for shortest path calculations. The routers use these shortest paths to forward the packet-based traffic. The network administrators try to assign such weights to the links that result in such paths that do not cause congestion or are cheap to use. Fortz [51] showed that finding an optimal combination of link weights is NP-hard and was among the first who employed meta-heuristics to find an optimal weight system for OSPF routing protocol. Fortz [51] proposed a tabu search algorithm for setting of OSPF weights, such that the maximal link utilization over all links was minimized. He used a cost function that penalizes link utilization more aggressively when it approaches its maximum available capacity. Nucci et al. [100] proposed a TS heuristic algorithm that finds a link weight system to cover the transient link failures. Balon and Leduc [11] later extended the work of Fortz [51] and proposed a heuristic algorithm that is aware of the intra-domain traffic matrix and other BGP data. In this algorithm the link-weight system for inter-domain routing is optimized for given intra-domain and inter-domain data. Buriol et al. [18] used a GA with local search procedures to obtain improvements in solutions obtained through crossover to solve the weight setting problem for OSPF. Harmatos [67] proposed a simulated annealing algorithm for the same problem. He and Mort [68]

proposed a genetic algorithm to minimize the number of hops and congested nodes/links.

Load balancing

Load Balancing on the network is another problem of interest. Lin and Meng [85] have proposed a genetic algorithm for network routing with an additional requirement to balance load on the network. For each demand, there is a feasible route set numbered 1,..., k. Each chromosome is made up of a sequence of these numbers for each demand. So, the length of a chromosome is equal to the total number of LSPs (demands). The initial population is obtained randomly. The fitness function penalizes for the highest link utilization rate among all demands. The crossover and mutation operations are applied probabilistically, with probabilities adjusted according to the solution quality. When crossover is applied, the link with the highest utilization rate is chosen and the demand least contributing to this highly utilized link is chosen as a crossover point. Based on the fitness value, the proportional selection method is combined with elitism, i.e; the individual with highest fitness value is preserved in the population. The computational experiments conducted by the authors suggest that, compared to SPF (which is used in OSPF), this EA algorithm results in a more balanced network.

Path protection problem for dynamic traffic

Shao et al. [116] looked at this problem. For large Shared Risk Link Groups (SRLGs), finding backup paths that provide 100% protection is impractical. If a fully reliable backup path is not available, then the most reliable partial path may be used to backup the failures. The authors proposed a heuristic

method to identify a backup path that is disjoint to the original path. The algorithm first strives to find full backup paths with 100% survivability. But if that is not feasible, the second best path that has maximum reliability is chosen as backup.This reduces the impact of SRLG failures. The heuristic is able to find a (best) partial protection configuration quicker than the time it takes with an exact method.

Survivability mapping problem

The survivability mapping problem is to map the physical network on to a virtual topology, such that the logical topology is available even in cases of failures in the underlying physical layer. The survivability mapping problem is known to be NP-complete [115]. Ducatelle et al. [40] presented algorithms for protection against failures in large IP-over fiber networks. Their finding was that TS is one of the most effective techniques to solve this problem. Armitage et al. [9], Giroire et al. [59], Crochat and Boudec [32], and Nucci et al. [99] used TS to solve this problem. Fumagalli and Valcarenghi [54] solved this problem using SA. Ducatelle et al. [40] extended the TS algorithm of Armitage et al. [9] and proposed algorithms to find a survivable mapping and also to identify the vulnerable points in the network. Two algorithms were proposed to tackle the problem: the first one called SMART confirms the existence of a solution/ survivable logical mapping, and the second one called FASTSURV is a heuristic search for an optimal virtual topology. The authors claimed that these algorithms were the fastest and most scalable algorithms at the time of its publication. They compared their results with Modiano and Narula-Tam [92], who solved the same problem using ILP.

Ergin et al. [45] tried to solve the problem of survivable virtual topology mapping for WDM technology with the additional constraint to minimize the resource usage.They proposed heuristic solutions based on EA and ACO. For the evolutionary method, the solution is represented as a string of numbers of length l, where l is the total number of logical paths to be mapped. Each logical path has k pre-defined shortest paths over the physical network. Each element in the solution string represents the index of the selected path from the set of k paths. Fitness evaluation checks for violations of survivability and capacity constraints. The algorithm was run until fitness evaluation was carried out for a predefined number of times. For ACO, the ants check each shortest path in the predefined set for each logical path. If the capacity or survivability constraint were violated, the ants updated the pheromone tables. The time required to solve the test problems (based on 24-node and 48-node graphs) with the heuristic algorithms was much less than what was obtained through ILP. Among the two heuristic approaches, their evolutionary algorithm showed better performance.

Network loading

Network loading is a design as well as routing problem. The given demands must be met by installing (loading) facilities on the arcs. Installation of a facility on a particular link has both facility-specific as well as link-specific costs. The objective of this optimization problem is to load the network at a minimum cost by selecting appropriate facilities from a given set of alternatives.

Campanale et al. [19] solved the same problem using tabu search algorithm and Giovanni et al. [57] suggested a neighbourhood search method.

Gendron et al. [56] used local search and GRASP metaheuristics to solve the network loading problem.

4.3 Hybrid Methods

Unlike exact methods, heuristics cannot guarantee optimality. Yet, when an exhaustive search is not feasible, heuristics can yield a very good quality solution in acceptable time limits. The idea of combining heuristics and exact methods is not new. By introducing heuristics in an otherwise exact algorithm, the solution times can often be improved. This combination has been so successful that a new area of research called matheuristics has emerged in recent years. *Matheuristics is the combination of meta-heuristics and mathematical programming.* It is the methodology of choice in Chapter 7 in our hybrid algorithm to solve the routing problem.

Almost all branch and bound methods use heuristics at multiple stages [107]. The initial solution is arrived at heuristically in all optimization engines such as CPLEX. At each node in the branching tree, heuristic algorithms such as (primal heuristics, dual heuristics) are used to reduce the search space, prune the searching tree and improve the upper bounds. Similarly heuristics are combined effectively with other exact techniques like branch-and-cut, cutting planes, branch-cut-and-price and column generation.

The frameworks to combine heuristics with exact methods to solve combinatorial problems have been presented by Puchinger and Raidl [107] and Dumitrescu and Stützle [42]. They have also presented structural models for combining exact and heuristics methods. Puchinger and Raidl [107] divide the hybridization in two main classes: collaborative and integrative. In collabora-

tive combination, both exact and heuristic algorithms exchange information, while in integrative combination, either the exact or the heuristic algorithm is the master and the other one is a subordinate technique. Dumitrescu and Stützle [42] suggests five strategies of hybridization.

- Main algorithm is local search and very large neighbourhoods are explored with exact method
- Optimization process is divided in two stages. Firstly, a local search method is used to find high quality solutions, then, using these solutions, smaller problems are constructed to be solved exactly.
- First, lower bounds are obtained by exact method and then these bounds are used to guide the constructive heuristics.
- LP relaxations are solved and then heuristic algorithms are run using the obtained information
- Specific procedures are solved exactly in an otherwise heuristic procedure.

Puchinger and Raidl [107] and Dumitrescu and Stützle [42] also provide examples based on various combinatorial optimization problems where such hybridization is applied. Fernandes and Loureno [48] review both the frameworks and draw similarities and differences between the two and cite multiple examples from the literature that fall under each specified category. However, none of these authors have cited any examples of hybridization in the area of network routing. Nevertheless, some instances have been found by us through careful scrutiny of the extant literature. Here we present spe-

cific examples from the literature where a combination of exact and heuristic techniques has been used to solve the problems related to network routing.

Using GA with LP to solve multi-objective optimization problem of minimum cost constrained multipath with load balancing for an MPLS network

El-Alfy et al. [43] use multi-objective constrained programming. Using LP of the relaxed problem, an initial population of solutions is generated. This population is fed into a GA algorithm, which searches for an optimal solution. For fitness evaluation that determines the probability of individuals to survive and reproduce, a linear function combining link's utilization and cost is used. They have performed tests on 10-node network and found that the hybrid approach provides superior solutions to a pure GA.

Mixing column generation and cutting plane with meta heuristics for network design problem

Chabrier [22] uses the branch-price and cut framework and enhances usual column generation with other techniques such as cutting plane generation and meta heuristics. These alterations have helped in finding good lower bounds, gaps and upper bounds for the network design problem. The problem looks at identifying optimal paths between node-pairs and capacity requirements of each path. For column generation, a pool of paths is maintained. This alleviates the need to find paths at each iteration. The quality of solutions obtained through branch & price is poor. The bounds are also not very tight. The algorithm is further enhanced by adding cutting planes. Cuts are added not only at the root, but to the subsequent nodes as well. The cut modifies

the pricing problem and the candidate column's reduced cost must take it into account. The improvement in solution and bounds are obtained by meta-heuristics.

Using neighbourhood search with ILP for IP design problem with reliability and routing problem

Design and route allocation of an IP network with additional constraints of reliability and hop limit is one of the well-known routing problems. All traffic must be sent over the network even in cases of failures. This problem is known to be NP-hard [34]. Bley et al. [14] have proposed an MILP algorithm and heuristic local search algorithms for the OSPF protocol for this problem. It searches through different kinds of neighbourhoods and identifies an optimal survivable routing for a given network. Gabrel et al. [55] propose a heuristic algorithm for discrete cost multi-commodity network optimization problems that is based on the the bender's cutting plane technique.

De Giovanni and Tadei [34] have worked on the same problem and have presented two algorithms: one using local neighbourhood search and the other using TS. For the Interior Gateway Protocol (IGP) within an Autonomous System (AS) over network layer, the authors look at the problem of capacity allocation with minimum cost-under QoS constraints. QoS is measured in terms of number of hops, protection strategy and maximum transmission delay. The main emphasis is to provide protection against node-failure scenarios while using the equal splitting rule of OSPF routing protocol. Among the two, tabu search gives better results - especially when small sized neighbourhoods are used in the diversification step. They have proposed and tested with different initial solutions that are generated in greedy or random ways. They generate

reduced-size neighbourhoods, that facilitate in exploring large solution spaces. Using tabu search, they are able to solve some of the real-life instances.

Mixing column generation with local search with path-relinking for minimization of maximum link utilization

Santos et al. [113] have looked at the minimization of maximum link utilization problem. They have proposed a hybrid algorithm, where column generation is used along with local search with path-relinking to solve the otherwise computationally hard optimization problem.

Mixing exact IP routing with non-deterministic tie-breaking in case of Equal Cost Multipaths (ECMP)

Hock et al. [70] propose a heuristic optimizer to be used within IP-based routing protocols such as IS-IS, OSPF and MPLS to obtain resilient routing. Their objective is the minimization of maximal link utilization.

Using SA with LP solve multi-objective optimization problem for traffic engineering of MPLS network

Cerav-Erbas [21] try to achieve two traffic engineering objectives: minimization of cost and minimization of maximum link utilization. They start with a relaxed LP problem, and then divide the feasible set into smaller sets. SA framework is used to solve each of the subproblems separately.

Mixing ILP with GA to solve the Routing and Wavelength Allocation problem

Barpanda et al. [12] solved the ILP formulation for Routing and Wavelength Allocation problem with a GA and were able to solve the problem in polyno-

mial time and obtain near-optimal solution. They used multiple objectives of minimizing the congestion on the links as well as minimizing the number and length(number of hops) of the light paths.

4.4 Summary

In this Chapter we have given an overview of heuristic techniques and of those matheuristic methods combining heuristic with exact methods. Applications and algorithms in the area of network routing using these heuristic and matheuristic methods have been presented. These algorithms have shown promise in the simulated environment, but for almost all of them, implementation in real-life (on physical network) is still a challenge [41].

The applications where EA or ACO are used are plentiful. Other local search meta-heuristics are also being applied to different problems related to network routing. Combination of meta-heuristics and deterministic methods is also being used and shows great promise for future research. As pointed out by Puchinger and Raidl [107], hybridization of heuristic and exact methods is a promising area of research and there is a need to exploit this potential in the area of network routing as well. We have taken up this challenge in Chapter 7.

Except for the traditional use of heuristics within exact methods (such as heuristic search within branch and bound), the mixing of exact and meta-heuristic methods is a recent trend. In the area of network routing in general, except for [14], all the algorithms cited in Section 4.3 have been developed in the last 10 years. The work done in the area of network routing, with survivability constraint and multiple paths, based on hybrid approaches is very limited. Our contribution of the algorithm developed in Chapter 7 is an

attempt to add to this insufficient reservoir of knowledge.

5

Solving Routing Problems with ILP

In this Chapter we discuss in detail the problem of network routing and its variants. We present different ways to formulate the problem and analyze the characteristic of each formulation. Starting with a very basic single commodity flow problem, we build upon it to present a more challenging multi-commodity flow problem that uses multi-paths. We then discuss the complexity of this problem, before adding more restrictions on the flow allocation.

A network is defined as a set of nodes and links /edges. An edge or a link is a single bi-directional connection between two nodes. Let us assume

that there are no parallel links between any two nodes and there are no self-loops[1]. Furthermore, assume that all nodes are reachable, i.e. there exists a path from every node to every other node.

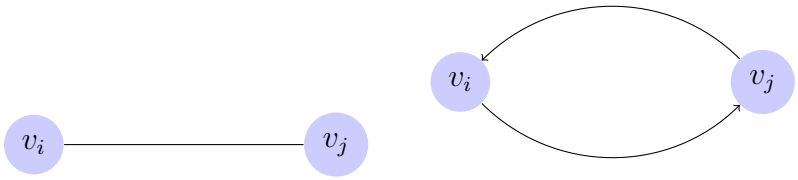
For a demand from a specified source node to a target node and with a given demand value, the single-commodity flow problem is to find a feasible flow from source node to target node, such that the flow on any edge is less than or equal to its total capacity. The underlying assumption is that the individual edge capacities are more than the demand value. If there are more than one commodities to be routed, the problem is called multi-commodity flow or network routing problem.

5.1 Single Path Allocation Problem

Let $G = (V, E)$ be a graph representing a network with $|V|$ nodes and $|E|$ edges. The node degree of a node i is the number of nodes directly connected connected to it. Each edge has a capacity b_e. The set of demands that are required to flow from v_i to v_j is denoted as D. For each $d \in D$ there is an associated flow value h_d to be routed on a single path. The single path allocation problem for a given demand set can be represented as an LP formulation, where we only need one set of constraints to check that the capacity usage is less than or equal to the available capacity on each edge. We do not need any other explicit objective for this formulation. The network flows can be modeled in two ways as described in Pióro and Medhi [105]:

1. by considering each individual node and evaluating the total flow passing through it via the links that are directly connected to it. The flow gen-

[1] a node connected to itself with a single link



(a) An undirected link

(b) An equivalent directed representation

Figure 5.1: Directed representation of an undirected link

erated by each demand is looked at individually as well. This approach is called node-based approach.

2. by considering each path for every demand and allocating the demand flow to one of its paths, while respecting the capacity constraints. This method is called path-based approach.

5.1.1 Node-based Approach

In order to distinguish between inflows and outflows, the node-based approach would require a network with directed links. An undirected link, as shown in Fig 5.1(a) can be represented as two directed arcs as shown in Figure 5.1(b).

For any demand, the total flow that can go out of its source node s must be equal to h_d. Similarly, for the target node t, total out-flow must be $-h_d$. For all other nodes, every flow that enters a node must also leave it, resulting in zero net flow. This is referred to as the flow conservation law, and is shown mathematically below in Eq. (5.1).

$$
\begin{aligned}
\text{if } v = s \quad & \sum_{e \in \delta^+(v)} f_{d,e} - \sum_{e \in \delta^-(v)} f_{d,e} = h_d \\
\text{if } v = t \quad & \sum_{e \in \delta^+(v)} f_{d,e} - \sum_{e \in \delta^-(v)} f_{d,e} = -h_d \\
\text{if } v \neq s,t \quad & \sum_{e \in \delta^+(v)} f_{d,e} - \sum_{e \in \delta^-(v)} f_{d,e} = 0
\end{aligned}
\tag{5.1}
$$

Here the flow variable $f_{d,e} \in \mathbb{R}^+ \cup \{0\}$. $\delta^+(v)$ is the set of edges incoming on node v and $\delta^-(v)$ is the set of edges going out of node v. A capacity constraint (5.2) can be added to complete the LP formulation that can find a feasible flow for each demand.

$$
\sum_{d \in D} f_{d,e} \leq b_e \quad \forall e \in E \tag{5.2}
$$

A feasible path needs to be constructed using the values of $f_{d,e}$. There is no guarantee that the generated path is indeed the best, i.e., that it is the shortest. The control over the path selection is not available in this formulation. To obtain such flexibility, we can refine this model by introducing a binary variable $\mu_{d,e} \in \{0, 1\}$, which is set to 1 if an edge e is used for demand d. See the modified formulation (5.3)-(5.4). This modification allows us to introduce the constraints that can ensure the selection of paths in accordance with a pre-defined criterion.

$$\left.\begin{aligned} \text{if } v = s \quad & \sum_{e \in \delta^+(v)} \mu_{d,e} - \sum_{e \in \delta^-(v)} \mu_{d,e} = 1 \\ \text{if } v = t \quad & \sum_{e \in \delta^+(v)} \mu_{d,e} - \sum_{e \in \delta^-(v)} \mu_{d,e} = -1 \\ \text{if } v \neq s,t \quad & \sum_{e \in \delta^+(v)} \mu_{d,e} - \sum_{e \in \delta^-(v)} \mu_{d,e} = 0 \end{aligned}\right\} \forall v \in V,\ d \in D \tag{5.3}$$

$$\begin{aligned} \sum_{d \in D} h_d \mu_{d,e} \leq b_e \quad \forall e \in E \\ \mu_{d,e} \in \{0,1\} \end{aligned} \tag{5.4}$$

With this modification we can now add constraints to restrict the selection of paths. For example, to limit the path length to a maximum of h hops, following set of constraints can be introduced in the model (5.3)- (5.4):

$$\sum_{e \in E} \mu_{d,e} \leq h \quad \forall d \in D \tag{5.5}$$

5.1.2 Path-based Approach

For this approach, a set P_d of candidate paths for demand $d \in D$ is a prerequisite. The candidate path-set is normally generated through one of the shortest-path algorithms, such as Dijkstra's, Surrballe or k-shortest path algorithm. In order to create a relationship between these paths and edges, a parameter ρ_{edp} is used, which shows whether a path p_d passes through an edge e. If a path is selected to route the flow for a demand, then the corresponding binary variable χ_{dp} will be set to 1. This single path allocation network flow problem can be formulated as shown in Eq. (5.6).

$$\sum_{p \in P_d} \chi_{dp} = 1, \quad \forall d \in D \tag{5.6a}$$

$$\sum_{d} \sum_{p \in P_d} \delta_{edp} h_d \chi_{dp} \leq b_e, \quad \forall e \in E \tag{5.6b}$$

The constraint (5.6a) restricts the number of selected paths per demand to one. The constraint (5.6b) enforces that the capacity is not assigned beyond its availability.

Comparison of number of variables and constraints of node-link and path-link formulations

The number of edges for a directed graph, with average node-degree as $\bar{k}$, can be calculated as $E = \frac{1}{2}(\bar{k} \cdot V)$. The maximum number of node-pairs that can act as source and destination for demands are $|V|$ $(|V|-1)$, when we assume that every node has a demand for every other node. The average number of paths per demand is $\bar{p}$. With an increase in the number of nodes, we can safely assume that the average node degree [2] does not increase. Pióro and Medhi [105] have also made this assumptions as this often is the case with real-life networks. For an example, with a network of 10 nodes with an average node degree of 3, adding a new node with node degree 3 will again result in an average node degree of 3 for the network of 11 nodes. Furthermore, assume that the average number of paths per demand is also kept fixed with an increase in the network size, the number of variables for both formulations are:

Node-based formulation:

[2] Node degree is the number of direct links from a node to other nodes in the network

$\mu_{d,e}$ variables will result in $|V|(|V|-1) \cdot \frac{1}{2}(\bar{k} \cdot |V|) \approx O(|V|^3)$

Path-based formulation:

$\chi_{d,p}$ variables will result in $|V|(|V|-1) \cdot \bar{p} \approx O(|V|^2)$

The number of constraints for both formulations are as follows:

Node-based formulation:

$|V| \cdot |D| + |E|$ constraints will result in $|V| \cdot |V|(|V|-1) + \frac{1}{2}(\bar{k} \cdot |V|) \approx O(|V|^3)$

Path-based formulation:

$|D| \cdot \bar{p} + |E|$ constraints will result in $|V|(|V|-1) \cdot \bar{p} + \frac{1}{2}(\bar{k} \cdot |V|) \approx O(|V|^2)$

As we can see, if we fix the number of paths per demand, so that the number of paths per demand does not increase with the size of the network, the path-based approach is asymptomatically better than the node-based approach, as the number of variables and constraints required for this approach are one order of magnitude less than those of the node-based approach. In addition, the path-based approach is more flexible:

- With the node-based approach, we do not need to have a list of candidate paths, which is a pre-requisite for the path-based approach. However, depending on the need, it can be seen to be advantageous to have control over the path selection. With the pre-computed lists we can disallow certain edges, if we wish so, to be used by certain demands. We can limit the number of hops by selecting only the shorter paths. We can also restrict the path list to disjoint paths.

- The path-based approach allows us to have two parallel demands i.e., between the same node pair. This type of demands may be required when we want to provide different QoS between the same node-pair.

– The path-based formulation can be restructured to represent uni-cast and multi-cast flow routing.

Considering the flexibility and reduced complexity of the path-based approach, it seems to be a better choice for modeling exercise.

The network flow problem presented so far allocates single paths while satisfying the capacity constraints. We can extend this problem by introducing an objective in accordance with our networking needs. The most commonly used objectives are cost reduction, flow maximization or congestion minimization (maximum link utilization).

– In order to reduce the cost of the link installation and usage, an objective to minimize cost is used. Although installation costs are a concern at the network design stage, usage / leasing cost minimization can be an objective for path allocation. If the per-unit usage costs for each link are known, say c_e, then the objective is to minimize $\sum_{e \in E} \sum_{d \in D} \sum_{p \in P_d} c_e \delta_{d,p,e} \chi_{d,p}$. For an existing network it becomes significant only when routing costs are involved.

– Given the available capacity, one objective could be to find a routing in which maximum flow is achievable. This objective function is useful when the demand values are more than the available capacity. By maximizing flow, we can find the maximum number of demands realizable on the network. The objective will be to maximize $\sum_{d \in D} \sum_p h_d \chi_{d,p}$. The constraint (5.6a) will change to an inequality

$$\sum_p \chi_{dp} \leq 1, \quad \forall d \in D$$

– In order to have smooth flow over the network, we might want to minimize the congestion on any link. Congestion is measured as a function of

link usage. It is the amount of traffic passing through any link. We can measure congestion(ρ) in absolute terms as $\max_e \sum_d \sum_{p:e \in p} h_d \cdot \chi_{d,p}$. A relative measure of congestion (ρ_{max}) can be obtained as $max_e[\frac{\sum_d \sum_{p:e \in p} h_d \chi_{d,p}}{c_e}]$. An optimization objective of minimizing this ρ_{max} identifies a solution that will balance the load over all the links. By bringing the maximum congestion value down, we can reduce the difference between minimum and maximum link utilization, resulting in a more balanced network traffic.

The flow maximization and minimizing the maximum link utilization are closely related objective functions. If we want to maximize the flow, we are in effect trying to reduce the un-utilized link capacity; hence maximizing the link utilization. If, however, the capacities are much larger than the demands to flow (which is generally the case in practice), then the flow maximization may yield inefficient solutions with respect to link utilization. The worst case scenario is that one link utilizes 99.9% of its capacity, while others are severely under-utilized. So, a slight increase in demand will result in congestion. On the other hand, minimizing the maximum link utilization, will result in a more balanced network usage, and therefore, not only can we maximize the flow, but we can also estimate the possible growth margin in demand values that can be achieved without causing congestion.

5.2 Multipaths in Routing

5.2.1 Multipath structures for network resilience

The routing efficiency can be increased by using multipaths instead of one path per demand. We have already provided an account of these methods in Chapter 3. The use of multi-paths can have number of potential advantages, such as:

- The use of multi-paths can improve the utilization of network resources (Ahuja and Ramasubramanian [7]). The use of multipaths reduces backup capacity requirement by allowing sharing of different connections in different failure scenarios (Menth et al. [90])
- The multi-paths have been proposed specifically for routing where the aim is to improve the resilience of the network against failures. There have been a number of models presented in the literature for failure protection using multi-path routing, such as Cherubini et al. [25], Alouneh et al. [8], Menth et al. [88, 89]
- For wired networks, multi-path routing can reduce the connection-establishment time as compared to single path routing (Cidon et al. [27]), while for mobile and ad hoc networking, multi-paths are found to decrease the packet dropping and hence help in combating the inherent unreliability of these networks.(Tsirigos and Haas [126])
- There is a lot of research showing the potential of multi-paths for QoS based routing. See for example [117],[101], [97], [142],[24], [123] and [132]

– Multi-paths result in a more balanced network utilization, while providing QoS and protection against failures. Van Do et al. [128] showed it on MPLS based systems.

The potential disadvantage of multi-paths is the need to concatenate the bifurcated traffic and re-order it at the receiving end, which can cause some delay in transmission. Re-ordering of the packets at the receiving end would also require some additional buffer space. There are a number of proposals to get around this problem. For example, Yabandeh et al. [141] propose packet scheduling to avoid delays among these multi-paths.

5.2.2 Self-Protecting Multipaths (SPM)

In Chapter 3 we introduced a multi-path scheme called "Self-Protecting Multipaths (SPM)" proposed by Menth et al. [88, 90] to provide protection against link and node failures. They argue that the use of multi-paths reduces backup capacity requirement by allowing sharing of different connections in different failure scenarios. In this Section we describe SPM in greater detail, as we will build upon this model and will introduce path selection feature.

SPM supports packet flow between source and destination nodes, using end-to-end multi-paths. In this model, an SPM is allocated to each source and destination node-pair having a positive demand. Each SPM consists of a predetermined number of disjoint paths. Traffic is distributed on these paths through a load balancing function. In case of a (single link/node) failure affecting the demand, one of the paths becomes unavailable. The load is then redistributed to the rest of the paths.

For each failure scenario, SPM uses three types of constraints to assign

load $\ell^s_{d,p}$ to each path in the multi-path:

- Load distribution on all paths must add up to 100% in all failure scenarios.
- No assignment of load to a failed path.
- Capacity constraints

Load balancing can takes place in three different ways: (i) equal load balancing, (ii) reciprocal to the path length, and, (iii) an optimized load balancing function for SPM (o-SPM). Menth et al. [88, 90] have also proposed two further variations of SPM: integer SPM(i-SPM) and failure-specific SPM(f-SPM).

Integer SPM algorithm does not allow for bifurcation of traffic on multi-paths. Instead, it takes only one path out of many for traffic flow. In effect, i-SPM acts as a path selection protocol. In case of a path failure, i-SPM can either select any one path of the remaining paths within SPM, or distribute the traffic using a load balancing function. f-SPM takes into account each failure state and provides a load balancing function specific to each failure location in an affected (failed) path. Hence, for one failed path there can be different load distributions corresponding to each failed link. In o-SPM, failing of a path will result in only one specific load distribution irrespective of which edge caused the failure. Although in f-SPM, there are more degrees of freedom for optimization, it is computationally more complex to solve. i-SPM has been found to be a little less efficient than o-SPM; yet it is computationally faster than o-SPM (Martin et al. [86]), and, therefore, the performance of i-SPM for larger networks can be better than that of o-SPM. Menth et al. [90] found

that optimizing the load balancing functions results in a most efficient capacity usage. An additional 20% capacity provides sufficient protection against all single link/node failures. o-SPM can carry 50% to 200 % more protected traffic than IP re-routing that uses OSPF.

We will use o-SPM approach to formulate a more flexible path selection model for multi-path routing.

5.3 Path Selection for SPM

For SPM, a set of k disjoint paths for each demand is a pre-requisite. Therefore, disjoint paths are identified at first– through repetitive application of a shortest path algorithm. This pre-determination can act as a bottleneck for the whole optimization process. Since the aim is to optimize the link utilization of the overall network, the shortest paths with regard to one demand might force other demands to use longer paths resulting in higher link utilizations. Therefore, instead of using pre-determined set of paths, as in SPM, we allow the optimization process to choose among a large set of paths. This extension is based on the assumption that there exist at least 2 disjoint paths for each demand, as otherwise, the concept of multi-paths cannot be implemented and failure against links cannot be guaranteed. The optimization process selects not more than k disjoint paths per demand.

In case of SPM, the load is distributed to all available disjoint paths. However, when path selection is also part of the optimization process, there is a need to introduce a binary variable to determine whether a path is selected for routing or not, and to ensure that the selected paths are indeed disjoint.

In order to provide resilience, we need to specify the failure scenarios

that must be protected. We can choose nodes , links, or both, to be protected. The links span over large areas, both underground and underwater. The operational costs to recover a failed link is much higher than that of recovering from a failed node. Also, at the time of installation, most nodes are provided with a backup, which can be switched over in case of a node-failure. The link failures, on the other hand, are harder to detect and recover. We therefore concentrate on protection mechanisms against link-failures.

5.3.1 Model description

The path selection for multipath routing can be modelled with the help of the path-based formulation. Let $G = (N, E)$ be an undirected graph representing a telecommunication network, where N is a set of nodes in the network, and E a set of links between the nodes. Let b_e be the capacity of link $e \in E$. The set D represents the demands between node pairs, with h_d volume of demand to flow. The set of paths between source and destination of demand $d \in D$ is denoted by P_d. We only take single link failures into consideration, but the model can be generalized to more than one simultaneous link failing scenarios. To protect against single link failures, we must at least have two disjoint paths for every demand. In general, if we want to protect m simultaneous link failures, we need to ensure that the path set contains at least $m + 1$ disjoint paths with sufficient spare capacities.

A set S consists of all single link failure scenarios, hence $S := E$. We also consider the non-failure scenario s_0, denoted by '0'. Hence, we define a set S_0, where $S_0 := E \cup \{0\}$.

The edges on paths for each demand are given as input to the model

with the help of a three dimensional matrix δ. Element δ_{edp} is set to 1, if and only if link e is on path p of demand d.

Binary variable χ_{dp} is set to 1 if path $p \in P_d$ is used for demand $d \in D$ in any of the scenarios, and 0 otherwise. For each scenario $s \in S$, the model determines a *load distribution* for demand d across paths P_d. The flow variable $l^s_{dp} \in [0,1]$ denotes the fraction of the traffic routed along path $p \in P_d$. The complete formulation is presented at (5.7).

$$\min \rho_{\max} \tag{5.7a}$$

$$s.t. \sum_{d \in D} \sum_{p \in P_d} h_d \delta_{edp} \ell^s_{dp} \leq b_e \rho_{\max} \quad \forall e \in E, s \in S_0 \tag{5.7b}$$

$$\sum_{p \in P_d} \ell^s_{dp} = 1 \qquad \forall d \in D, s \in S_0 \tag{5.7c}$$

$$\ell^s_{dp} \leq \chi_{dp} \qquad \forall d \in D, p \in P_d, s \in S_0 \tag{5.7d}$$

$$\sum_{p \in P_d} \delta_{edp} \chi_{dp} \leq 1 \qquad \forall d \in D, e \in E \tag{5.7e}$$

$$\delta_{sdp} \ell^s_{dp} = 0 \qquad \forall d \in D, p \in P_d, s \in S \tag{5.7f}$$

$$\sum_{p} \chi_{dp} \leq k, \qquad \forall d \in D \tag{5.7g}$$

$$\chi_{dp} \in \{0,1\}, \ell^s_{dp} \geq 0 \tag{5.7h}$$

Constraints (5.7b) evaluate the load on each edge and in each scenario and determine the maximum load congestion. For each demand and each scenario, constraints (5.7c) model the load distribution among the paths. A path can carry traffic only if it is one of the selected paths (cf. (5.7d)) and the selected paths for a demand have to be edge disjoint (cf. (5.7e)). These disjointness constraints ensure that at most one path is selected from all paths

containing a specific edge. Constraints (5.7g) restrict the maximum number of disjoint paths to be k. Finally, constraints (5.7f) state that a path cannot be used if one of its edges fails.

5.3.2 Complexity of the path selection problem

The path selection problem, as presented in the previous section, can be considered a splittable multi-commodity problem, where each demand's volume can be split on at most k-disjoint paths. Hence, for each demand, it is the problem of finding a set of k disjoint paths that can carry the demand volume. But it does not put any restriction on sharing of edges among different demands. There are a number of variants of the classic disjoint path problem.

The disjoint path problem is one of Karp's original NP-complete problems. Given an undirected Graph $G = (V, E)$, and a set of m node-pairs D, where each pair is $s, t \in V$. D is realizable in G, if there exist mutually edge-disjoint paths $p_1 \ldots p_m$, such that s_i, t_i are the endpoints for p_i. A variation of this problem is where there is a demand value attached to each node-pair and each edge has a capacity. Then D is realizable only if each demand can be sent over G, where available capacity on the edge is the upper bound on demand volume that can be sent over that edge. The special case where all s_i, t_i are the same, has been solved through Menger's and max-flow-min-cut theorems. Later, it was shown that for $m \leq 2$ (two or less commodities), the problem is solvable in polynomial time.

Kleinberg [80] has also studied three NP-complete variants of this problem:

– Unsplittable flow problem: Assume that a β_i is associated with each

demand-pair to show the demand value to be routed and the capacity of each edge is c_e. The problem is to find a subset of demands realizable on G, while satisfying the capacity constraints which is: $\sum_{i:e\in P_i} \beta i \leq c_e$ for every edge.

– Rounds: Minimum number of rounds to send D over G, where in each round the paths are disjoint, and

– Congestion: Assume that a β_i is associated with each demand-pair to show the demand value to be routed and the capacity of each edge is c_e. Then congestion is defined as $max_e \sum_{i:e\in P_i} \beta i$. The problem is to find a routing with minimum congestion possible. If the capacities are arbitrary, relative congestion can be defined for each edge by dividing congestion value by capacity of the edge, i.e.: $\max_e \sum_{i:e\in P_i} \beta i/c_e$. Then the problem is to find the minimum relative congestion possible while routing D over G.

The problem of minimum relative congestion restricts one path for each demand-pair. Our path selection problem has the same objective of minimum relative congestion, but the demand value is splittable among k disjoint paths. The other important difference is that we do not impose the disjointness constraint for paths used for different demand-pairs.

Baier et al. [10] have worked on the k-splittable flow problem, which allows splitting of demand over k not necessarily disjoint paths. For a single commodity, the maximum k-splittable s-t flow problem asks what is the maximum flow possible from s to t on k paths. If $k \geq |E|$, the problem can be solved efficiently. However for $k < |E|$, they argue that, the problem is already NP-Hard, as it a special case of single-source unsplittable flow problem.

Baier et al. [10] show the hardness of their problem by reduction from SAT problem. Using similar reasoning, we can show that SAT is reducible to our path selection problem as formulated in (5.7). As explained earlier, to protect single link failures, we need at least two disjoint paths for every demand. Once we have two disjoint paths with enough capacity, a feasible load distribution can be determined. The feasibility of the problem depends on the existence of two disjoint paths. We will discuss the complexity of path selection problem for the case where $k = 2$.

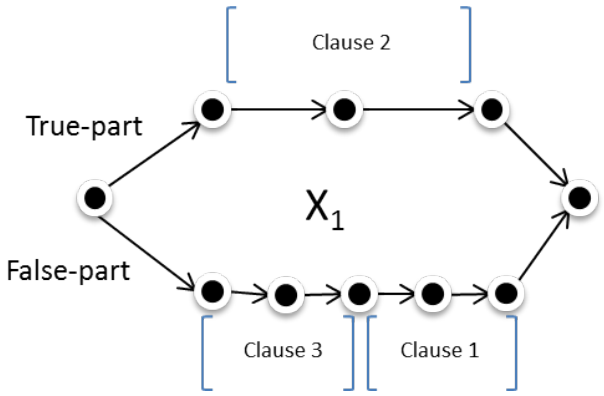
The Satisfiability (often written as SAT) problem is the decision problem to find if any boolean expression in conjunctive normal form (product of sums) has a true value. The boolean expression consists of conjunctions (AND) of clauses and each clause comprises of disjunctions (OR) of boolean variables. SAT is one of the classic NP-Complete problems presented by Cook [29]. An instance is said to be satisfiable, if its boolean variables can be assigned values such that the resultant value of the whole instance is true.

Proposition 1. *The path selection problem is NP-complete even in the case when only one node-pair has to be connected.*

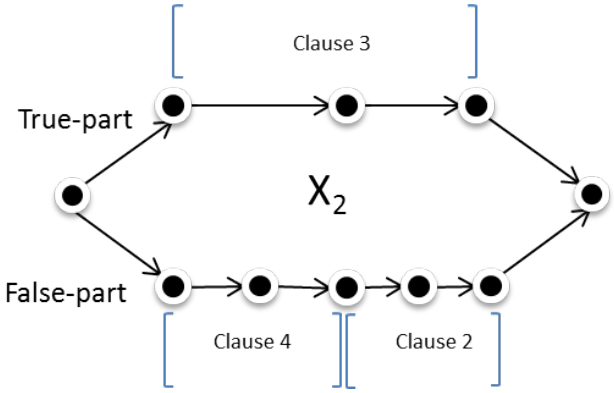
Proof. We consider D comprising of a one node-pair. We prove this by reduction from the SAT problem. If SAT is satisfiable, we can have a feasible solution to the path selection problem. If SAT is not satisfiable, then D is not realizable and the path selection problem does not have a feasible solution.

Consider a SAT instance, with $x_1, \ldots, x_n$ variables and m clauses.

Let us construct a graph for a demand from s to t using the SAT instance. Figure 5.3 shows a graph constructed using one example SAT instance:



(a) Variable subgraph for X_1



(b) Variable subgraph for X_2

Figure 5.2: Representation of variables

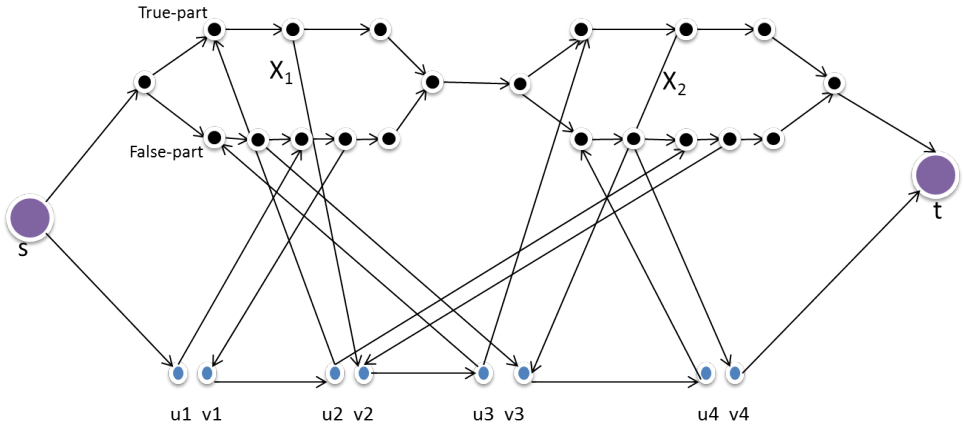


Figure 5.3: Graph for SAT instance: $(\bar{X_1}) \wedge (X_1 \vee \bar{X_2}) \wedge (\bar{X_1} \vee X_2) \wedge (\bar{X_2})$

$(\bar{X_1}) \wedge (X_1 \vee \bar{X_2}) \wedge (\bar{X_1} \vee X_2) \wedge (\bar{X_2})$

In order to construct the graph, we first draw a subgraph for each variable to ensure that the true assignments in the SAT clauses are consistent. For each variable x_i, we check each clause in a descending order. So, if the SAT instance has 4 clauses, we will start with clause 4 and will go down to clause 1. If a clause C_j has x_i, we add two consecutive edges (one primary and one secondary edge) in the true part of the x_i subgraph, if C_j has $\bar{x_i}$, then we add two consecutive edges in the false part. See variable subgraphs for X_1 and X_2 in Fig. 5.2, which shows the construction of the subgraphs with respect to their values in each clause.

Now, to encode the clauses, we need to add new nodes. For all clauses add a node-pair, u, v. Connect source s to the first variable subgraph and connect the last variable subgraph to target node t. Furthermore, connect s u_1 and connect v_m to t. Now, connect u_i to the tail of the primary edge for each variable that is present in the clause C_i. The head of this primary edge is connected to v_i. A directed link is added between each v_i and u_{i+1}.

All of the variable subgraphs are connected to each other with a connecting link, and the source node s is connected to x_1, while x_m is connected to the target node t. The complete representation for the example SAT instance is shown in Fig. 5.3.

Let us consider the case where all edges have unit capacity, and we also want to send a demand value of 1 unit from source node s to target node t.

Now assume that the SAT instance is satisfiable. We send 1 unit of demand by fixing a satisfiable assignment. If a feasible routing exists, then we need two disjoint paths for the demand, such that the capacity constraint is satisfied. The path from source will go through variable graph, using either true-part or the false-part. The second disjoint path with the same capacity is available through nodes representing clauses. The structure of the graph is such that u_i, v_i path will be disjoint from the path passing through true or false-parts. Consequently, if the SAT instance is satisfiable, there will be two disjoint paths for the demand, with enough capacity to carry to the demand volume in case one of the two paths fails.

The second part of the proposition, that if SAT is not satisfiable, then D is not realizable and the path selection problem does not have a feasible solution, can also be shown using an example. Consider an unsatisfiable SAT instance: $(\bar{X_1}) \wedge (X_1 \vee X_2) \wedge (\bar{X_2})$
We draw variable subgraphs and add nodes that represent clauses using the same method as described above. The resultant graph is shown in 5.4.

In order to construct two disjoint paths, one path must go through the variable subgraph, while the second one should traverse the nodes representing clauses. However, any path that uses clause nodes, will either use links that belong to both parts (true and false) of variable subgraphs (either x_1 or x_2) or

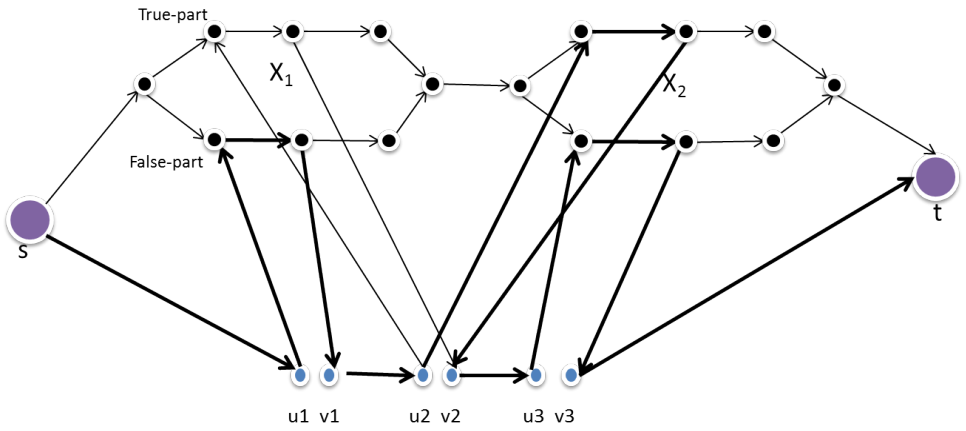


Figure 5.4: Graph for an unsatisfiable SAT instance: $(\bar{X_1}) \wedge (X_1 \vee X_2) \wedge (\bar{X_2})$

will use the link connecting two variable subgraphs. Thus, it is not possible to find a second path which is disjoint to the first one. The thick lines in the Figure 5.4 shows a path to illustrate this fact. This path uses both true and false parts of the variable subgraph. □

5.3.3 Adding scenario-specific constraints

The routing problem discussed so far does not prohibit re-distribution of load for demands that are not affected by a failure scenario. Thus, quite a few demands may change their load distribution with every failure in the network. This is not desirable from the management point of view. Switch over of load from one path to another may disrupt the flow unnecessarily. It is desirable and more practical to allow re-distribution of load only when a failure affects a demand flow: i.e., one of its multi-paths becomes unavailable. Such restrictions can be modeled though constraints (5.8).

$$\ell^s_{dp} - \ell^0_{dp} \quad \leq \sum_{q \in P_d} \delta_{sdq} \chi_{dq} \qquad \forall d \in D, p \in P_d, s \in S \tag{5.8a}$$

$$\ell^0_{dp} - \ell^s_{dp} \quad \leq \sum_{q \in P_d} \delta_{sdq} \chi_{dq} \qquad \forall d \in D, p \in P_d, s \in S \tag{5.8b}$$

$$\ell^i_{dp} - \ell^j_{dp} \leq 1 - \delta_{idq}\delta_{jdq}\chi_{dq} \forall d \in D, p, q \in P_d, i, j \in S \tag{5.8c}$$

Constraints (5.8a) and constraints (5.8b) ensure that if a failed edge is not on any of the selected paths, its failure does not affect the load distribution. The load on any path should remain identical to its load assignment in '0' (i.e. no-failure) scenario. This is the case when an edge s is not on any of the selected paths. Hence the right hand side (rhs) of the inequalities equals zero. If the edge s is on one of the selected paths, the rhs of both of these equations will equal 1, implying that these constraints have no effect. In any case, the disjointness constraint (5.7e) will not let the rhs of constraints (5.8a) and constraints (5.8b) exceed 1.

The failing of an edge, which belongs to one of the selected paths, results in the failing of that path. Constraints (5.8c) restrict the load distribution in this case. They enforce that, irrespective of the cause of the failure, when a specific path fails, the resultant load distribution on the remaining paths should be exactly the same. This condition has been imposed by considering two failure scenarios: (i) failure of edge i and (ii) failure of edge j. The rhs of this inequality will become 0 when both edges i and j are on the selected path p – thus enforcing the loads on remaining paths to remain identical in these two scenarios. If both i and j are not on the path p, the rhs will equal 1, allowing the load values to vary by at most 1, which is the maximum load

value.

The model presented at (5.7) and (5.8) constitutes the complete formulation for our considered routing problem and will be referred as E1 formulation hereafter.

5.3.4 Model improvements

From the model presented at (5.7) with additional constraints (5.8) it is clear that, given a set of paths for every demand, not all variables and constraints are needed. The variables involved in constraints (5.7f) can only take the value zero. Therefore, these variables do not have to be generated. Also, if failure scenario $s \in S_0$ (read: edge s) is not contained in any path $q \in P_d$ of demand d, inequalities (5.8a) and (5.8b) force $\ell^s_{dp} = \ell^0_{dp}$, and thus variables ℓ^s_{dp} can be handled implicitly by using ℓ^0_{dp} instead.

Next, constraints (5.8b) can be relaxed/removed without changing the problem. Note first that the right hand sides (rhs) of constraints (5.8a) and (5.8b) are equal. If it is equal to 1, the constraints have no impact since both $\ell^s_{dp} \in [0, 1]$ and $\ell^0_{dp} \in [0, 1]$. Further, the rhs do not depend on path p at the lhs. In case the rhs equals zero, by relaxing inequalities (5.8b), one might decrease but not increase the load on path p in case of failure s. However, since the sum of all paths must equal 1 (cf. constraints (5.7c)), a decrease would require an increase in another variable.

Furthermore, the right hand side of constraints (5.8c) is either 1 or $1 - \chi_{dq}$. The latter only holds if both i and j are on path q. In most cases this will not be so and thus the inequality does not restrict the load distribution in such cases and should not be generated. In our experiments (which we will

present in the next Chapter), constraints (5.8c) would make up 80% or more of the total number of constraints, slowing down the solving process considerably.

Finally, constraints (5.8c) with right hand side $1-\chi_{dq}$ can be replaced by a stronger set of inequalities, i.e., a set of valid inequalities that dominate the original constraints and have LP relaxation value not worse than the original model. For this, observe that the disjoint paths constraints (5.7e) imply that at most one χ_{dq} will be set to 1 among all paths $q \in P_d$ containing both i and j. Hence, for $d \in D$, $p \in P_d$, $i, j \in S_0$, we can subtract all paths $q \in P_d$ at once:

$$\ell^i_{dp} - \ell^j_{dp} \leq 1 - \sum_{q \in P_d} \delta_{idq}\delta_{jdq}\chi_{dq} \tag{5.9}$$

The path selection problem with additional constraints (5.8) also has a mandatory requirement of at least 2 disjoint paths, to ensure a feasible solution. Hence, an NP-complete problem is a pre-requisite for this extended path selection problem. But the effect of adding (5.8) to the constraints is not clear. It can make the problem easy or otherwise. For network design problem where stub-release mechanism is used and failure restoration is state-dependent, Tomaszewski [125] has proved that this problem is NP-hard for even a single commodity and for single failure.

5.4 An alternative Path Selection Model

The analysis of the mathematical formulation presented at (5.7) with additional constraints (5.8) for the path selection problem shows that many of the load variables are redundant, resulting in a very large model. It is due

to the fact that we have introduced one load variable for each edge-scenario. There is a variable ℓ^e_{dp} corresponding to each demand, path and scenario (i.e., edge failure) triplet. If the average size of a path is 5 edges, then there are 5 variables depicting the load on a path and through constraints we force this load to be identical, as we wanted to ensure that if a path fails, the load re-distribution is identical irrespective of the cause of a failure. Hence, only one variable could be used, meaning four out of five of ℓ^e_{dp} are redundant. In this Section we present another formulation, where we reduce ℓ variables to $|D|\cdot|P|$. The change in the load values in different scenarios is represented by another variable α^i_{dp}.

This improved model formulation uses the same notations and definitions used for presenting (5.7), with the following modifications:

- The flow variable $\ell_{dp} \in [0, 1]$ now denotes the fraction of the demand traffic routed along path $p \in P_d$ only in the no-failure state.

- The set S_0 is not defined. The no-failure is dealt with differently. As previously, the set $S = E$ defines the failure scenarios.

- A path may fail as a result of a failure of any of its component edges. We assign one path scenario corresponding to all edge scenarios $s \in S$ on each path. If there are k paths in P_d, there are k path-failing scenarios denoted as PS_d. In case of a path failure as a consequence of any of the corresponding edge scenarios, α^i_{dp} stores the change in the load value from the no-failure load value. Hence, in case of failure of path i the total load on path p will be $\ell_{dp} + \alpha^i_{dp}$. The value of α^i_{dp} can range between -1 and 1.

The improved model is formulated as (5.10) below. It will be referred to as E2 formulation:

$$\min \rho_{\max} \tag{5.10a}$$

$$s.t. \sum_{d\in D}\sum_{p\in P_d} h_d \delta_{edp} \ell_{dp} \leq b_e \rho_{\max} \qquad \forall e \in E \tag{5.10b}$$

$$load_{dpes} = \delta_{edp}(\ell_{dp} + \sum_{i\in PS_d} \alpha^i_{dp}\delta_{sdi}) \tag{5.10c}$$

$$\sum_{d\in D} h_d \sum_{p\in P_d} load_{dpes} \leq b_e \rho_{\max} \qquad \forall e \in E, s \in S \tag{5.10d}$$

$$\sum_{p\in P_d} \delta_{edp}\chi_{dp} \leq 1 \qquad \forall d \in D, e \in E \tag{5.10e}$$

$$\sum_{p\in P_d} \ell_{dp} = 1 \qquad \forall d \in D \tag{5.10f}$$

$$\ell_{dp} \leq \chi_{dp} \qquad \forall d \in D, p \in P_d \tag{5.10g}$$

$$\alpha^i_{dp} \leq \chi_{di} \qquad \forall d \in D, p \in P_d, i \in PS_d \tag{5.10h}$$

$$\alpha^i_{dp} \geq -\chi_{di} \qquad \forall d \in D, p \in P_d, i \in PS_d \tag{5.10i}$$

$$\sum_{p\in P_d: p\neq i} \alpha^i_{dp} = \ell_{di} \qquad \forall d \in D, i \in PS_d \tag{5.10j}$$

$$\sum_{p\in P_d} (\ell_{dp} + \alpha^i_{dp}) = 1 \qquad \forall d \in D, i \in PS_d \tag{5.10k}$$

$$\ell_{dp} + \alpha^i_{dp} \leq \chi_{dp} \qquad \forall d \in D, p \in P_d, i \in PS_d \tag{5.10l}$$

$$\ell_{dp} + \alpha^i_{dp} \geq 0 \qquad \forall d \in D, p \in P_d, i \in PS_d \tag{5.10m}$$

$$\sum_{p} \chi_{dp} \leq k, \qquad \forall d \in D \tag{5.10n}$$

$$\chi_{dp} \in \{0,1\}, \ell_{dp} \geq 0, \alpha^i_{dp} \geq -1 \tag{5.10o}$$

Constraints (5.10d) evaluate the load on each edge and in each scenario

and determine the maximum load congestion. This equation does not include the no-failure scenario. The correspondence between edge scenarios and path-failing scenarios is established in constraints (5.10d). Constraints (5.10b) model the capacity constraints in the no-failure state. Constraints (5.10f) model the load distribution among the paths in the no-failure scenario and constraints (5.10k) model the load distribution when there are failures. A path can carry traffic only if it is part of SPM: i.e., it is one of the selected paths (cf. (5.10g)). constraints (5.10e) restricts that the selected paths for an SPM for each demand are edge disjoint. The value of α^i_{dp} may vary between -1 and 1 for the paths that were selected for SPM for demand d. For all other paths, α^i_{dp} is forced to be 0. This condition is enforced through constraints (5.10h) and (5.10i). In case of a path failure, constraints (5.10j) put an upper bound to load that can be re-distributed over other working paths. It must equal the load lost as a consequence of a failure. Constraints (5.10l) and (5.10m) impose lower and upper bounds on the total load carried by a path. Constraints (5.10n) set the upper bound on the maximum number of paths selected for any demand.

By introducing the concept of path-failing scenarios through α^i_{dp}, the problem size is reduced considerably. Model (5.7) with additional constraints (5.8) generates approximately $2 \cdot |E|+4 \cdot |P_d| \cdot |E|+|P_d| \cdot |E| \cdot |E|$ constraints, while in the improved version presented at (5.10) we need $|E|^2+|E|+|D| \cdot (1+|E|+2 \cdot |P_d|+4 \cdot |P_d|^2)$ constraints. Taking an example of a small network with 5 nodes, 10 edges and 2 demands with 2 paths each, model (5.7) requires (40+4)=44 variables and (100+2(20+80+200))=700 constraints while with the improved version (5.10) we need 4+8+4=16 (instead of 44) variables and 172 (instead of 700) constraints.

5.5 Summary

In this chapter we have discussed the single path and multipath network flow problems in detail. We have looked at the benefit of multipath routing for enhancing network resilience. We have proposed a path selection model for multipath routing, which provides load distribution for all failure scenarios. The distribution is such that in all single-link failures, 100% traffic can flow without any loss.

The complexity of the path selection problem is discussed. We have demonstrated that the path selection problem (without scenario based constraints) is NP-complete even for the case of one node-pair.

We have discussed the extensions of the path selection problem, where we introduce more restrictions on path selection as well as on the load distribution. This model is referred to as E1 in the following chapters. With the addition of these scenario-based constraints, the number of variables and constraints increases quickly with an increase in the problem size. We have proposed an alternative formulation (E2) in which the number of variables and constraints reduce considerably.

In the next Chapter we report and discuss the empirical results of different experiments we conducted the proposed models.

6

Evaluation of the ILP Models

In this chapter we will present the results of different experiments conducted to test the performance of E1 and E2 models, which were presented in the previous chapter. We will discuss and compare results of two cases: the case where pre-determined paths are given and the case where path selection is part of the optimization process. Furthermore, we will present a comparative analysis of the performance of our proposed methodology to the well-known 1+1 protection strategy.

The E1 and E2 models require path sets for each demand as an input.

In order to generate the suitable path sets, we developed a modified breadth-first search algorithm. The total number of paths for each node-pair can be prohibitively large, hence the path generation algorithm heuristically finds a good path set. However, once the paths are selected, the rest of the processing is based on ILP models and is deterministic in nature. The description of the path finding algorithm is provided before going into the details of the model implementations.

6.1 Path Generation

The paths between a node-pair in a network can be found by an exhaustive search based method such as breadth-first search (BFS) or depth-first search (DFS). These methods view the network as a tree, with the source node at the top (level 0). In BFS, all immediate children are searched from left to right. Then, starting from the left-most child, BFS is run on each one. All paths from source node to target node are generated. In contrast to BFS, DFS searches the first child-node, then applies DFS on it and carries on until the target node is found. It dives as deep as it can go to find the target node.

The number of paths between a node-pair depends on the size of the network and its node-degree. For a densely connected network, the possible number of paths between a node-pair can be large. For example, van Hoesel et al. [129] have presented the following Lemma 1

Lemma 1. *[129] The number of distinct simple paths (a path without node repetition) between any pair of nodes in a complete graph on $|V|$ nodes equals $\lfloor (|V|-2)!\, e \rfloor$, if $|V| \geq 3$.*

Using Lemma 1, the number of paths for a complete graph are given in

Table 6.1: Paths in a complete Graph

nodes	# of paths
5	16
10	109588
15	16925042534
20	1.74017E+16
25	7.02658E+22

Table 6.1. One can see that the number of paths grows exponentially with an increase in the network size, making path generation a non-trivial problem. Since generating a complete path set for large networks is not feasible, we need to think of other ways to find a limited set of paths that are good enough in some sense. An intuition would be to use the shortest path and other paths disjoint to it.

Dijkstra's algorithm [37] is well-known to find shortest paths. It works on directed graphs where the edge costs are non-negative. A modified Dijkstra's algorithm [105] can be used to find shortest paths with negative edge costs, provided there are no negative-cost cycles. Dijkstra's algorithm has the time complexity of $O(m + nlog(n))$, where n is the number of nodes and m is number of edges. Finding the shortest path where there are negative cost cycles in the graph is NP-hard. If there is a possibility of a negative cost cycle,we can use Bellman-Ford algorithm ([13], [50]) to detect it. This algorithm finds the shortest path or detects the presence of a negative cost cycle. The complexity of this algorithm is $O(nm)$. If the network is fully connected so that $m \approx n^2$, then its complexity is $O(n^3)$. In case of sparsely connected network, it is $O(nm) < O(n^3)$. There are many other algorithms presented in the literature for k-shortest paths, such as FloydWarshall algorithm[49] and

Johnson's algorithm [77].

In order to find a disjoint path set, Suurballe's algorithm [122] can be used. This algorithm is based on the Dijkstra's algorithm and its complexity is also the same as that of Dijkstra's algorithm (when a Fibonacci heap implementation is used). It also assumes that the graph has directed edges with non-negative costs.

For the purpose of our experimentation, using a shortest path algorithm or k-disjoint path algorithms, described above, do not suffice. Our main hypothesis is that a pre-selection of a disjoint path set (w.r.t the shortest path) results in a suboptimal solution. Therefore we try to generate a path set which is good enough and which increases the probability of finding better solutions than the pure disjoint one. But at the same time, we would like to avoid an exhaustive search. So, we want to generate a path set that has:

- the shortest path
- paths disjoint to each other – not necessarily w.r.t the shortest path
- paths with limited number of hops
- no cycles

The algorithm we are proposing is based on BFS. But, instead of an exhaustive search, we apply different restrictions to limit the path set to have only good enough paths. This algorithm applies BFS at both source and target nodes. In each iteration, the tree is expanded to the next level. But the expansion is done alternatively, i.e.; if the source node is expanded in one iteration, the target node will be expanded in the next one, and so on.

This alternative expansion and this two-way search provides us with the paths that are shortest in terms of number of hops, without doing an exhaustive search. If we want to limit the number of hops in the path set, we can easily do so by changing the stopping criterion.

6.1.1 Description of the algorithm

The path finding algorithm is formally presented as Algorithm (1). It finds paths between a node-pair (s_i, t_i). The immediate neighbours of a node x is represented by $N(x)$. We use two data structures $s - visited$ and $t - visited$, which store the information about the visited nodes. For each node, these data structures store the level and the parent. The level shows how many hops it is away from the source or target. Parent node is the one that adds an entry in the $s - visited$ or $t - visited$ tables. Using these data structures, the search moves forward from source and target nodes. At each iteration, a comparison between entries in both tables is made. Only the most recent level entries are considered in this comparison. If a node is in both tables, meaning that using this node as a pivot, paths can be generated. So, we backtrack from this node towards source and target and thus generate paths. The path pruning process removes cycles and then stores unique paths only.

The path finding process can be stopped using a criterion such as: i) desired number of paths found, ii) all paths less than a specified hop length iii) all nodes explored either by source or target BFS , or iv) all nodes explored by both BFS processes. In Algorithm (1) criterion (iii) is used.

The algorithm favours shorter paths to the longer ones. If two nodes are h hops away, and max node degree is k, then the time complexity of the

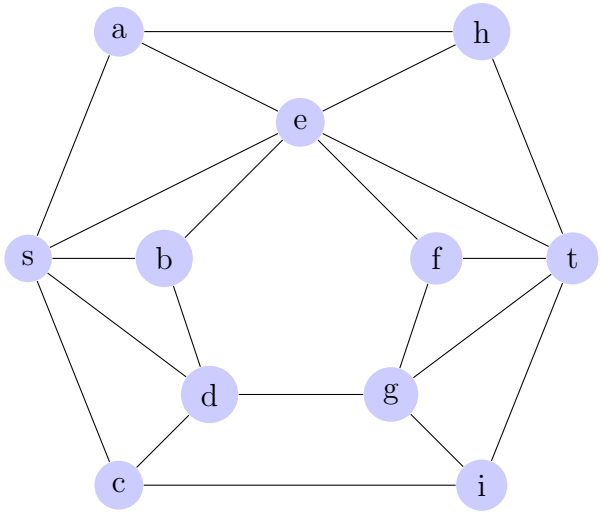


Figure 6.1: A sample graph

algorithm is $O((\frac{h}{2} \times k)^2) = O(h^2 \times k^2)$ to find the shortest path. In case of a complete network, $k = |V|-1$. So, the complexity of the algorithm is $O(h^2 \times |V|^2)$. In the worst case, every node and edge will be explored, so the complexity of this algorithm would be $O(|V| \times |E|)$ -same as that of BFS.

The path set generated has a limited number of good enough paths and we can use different combinations of these paths for our experiments.

An example

Let us see the construction of a path set from s to t on a small example based on a graph shown in Fig 6.1.

Paths Formed:

Iteration 3: s-**e**-t

Iteration 4: s-a-**e**-t, s-a-**h**-t, s-e-**h**-t, s-e-**f**-t, s-b-**e**-t, s-d-**g**-t, s-c-**i**-t

Iteration 5: s-e-**f**-g-t, s-d-**g**-f-t, s-d-**g**-i-t, s-c-**i**-g-t

Algorithm 1 FindPaths-st

```
INPUT (s, t) ν = V
Initialize s − visited = s,0,Nil
Initialize t − visited = t,0,Nil
Initialize boolean flags: s-expand = true, added = false
τ = {s, t} s-level = 0, t-level=0
while true do
  for each k ∈ s − visited at s-level do
    for each j ∈ t − visited at t-level do
      if k == j then
        generate path(s)
      end if
    end for
  end for
  if τ ∩ V == φ then
    BREAK
  end if
  if s − expand == true then
    for each v ∈ s − visited at s-level do
      for each k ∈ N(v) and k ∉ τ do
        add (k, s − level + 1, v) triplet to s − visited
        added = true
        τ = τ ∪ v
      end for
    end for
    s − expand = false
    if added == true then
      increment s − level
      added = false
    end if
  else
    for each v ∈ t − visited at t-level do
      for each k ∈ N(v) and k ∉ τ do
        add (k, t − level + 1, v) triplet to t − visited
        added = true
        τ = τ ∪ v
      end for
    end for
    s − expand = true
    if added == true then
      increment t − level
      added = false
    end if
  end if
end while
return
```

The iterations are shown in Table6.2 below:

Table 6.2: Path generation process

Iteration	$s-visited$	$t-visited$	resolved set
-	s-nil-0	t-nil-0	-
1	a-s-1		
	e-s-1		
	b-s-1		
	d-s-1		
	c-s-1		{s}
2		h-t-1	
		e-t-1	
		f-t-1	
		g-t-1	
		i-t-1	{s,t}
3	e-a-2		
	h-a-2		
	a-e-2		
	b-e-2		
	h-e-2		
	f-e-2		
	e-b-2		
	d-b-2		
	b-d-2		
	c-d-2		
	g-d-2		
	d-c-2		
	i-c-2		{s,t,a,e,b,d,c}
4		g-f-2	
		f-g-2	
		i-g-2	
		g-i-2	{s,t,a,e,b,d,c,h,f,g,i}

6.2 Computational Experiments

In order to test the efficiency of our models presented in the previous chapter, we have performed different experiments that:

1. verify our main hypothesis that the path selection when part of optimization, results in reduced ρ_{max} (maximum link utilization),

2. compare the performance of the proposed ILP formulations E1 and E2, which were presented in the previous chapter, and

3. compare the proposed multipath strategy with the well-known 1+1 protection method.

6.2.1 Testing the impact of path selection

In order to show that the optimization process with path selection improves the capacity utilization of the network, we have considered two cases. In the first experiment we consider only disjoint paths, and we compare the link utilization values for cases with and without path selection option. The second case is an extension of the first one, where we add more paths in the path set and test the performance of our model.

We have used nobel-germany, nobel-eu and germany50 network instances available at the SNDlib library [104], which has been provided by the Zuse Institute of Berlin (ZIB), Germany. All of these instances have a low node-degree which averages around 3 edges per node, therefore for many demands two disjoint paths do not exist. However, our models are based on the assumption that there are at least two disjoint paths for every demand. Also, in order to test the effect of the number of available disjoint paths, we

require higher node-degree, so that we can have more disjoint paths to test with. Hence, we have introduced more edges to each of the test instances. This has been done systematically by adding edges between nodes having a common neighbour as long as both nodes have degrees less than five. To give an example, the graphs of the original nobel-eu instance and the modified one, after adding more edges, are shown at Fig. 6.2 and Fig. 6.3. The modified networks can be viewed as logical instances superimposed on the existing physical network. The network instances have been renamed to distinguish them from the original instances available at the SNDLib.

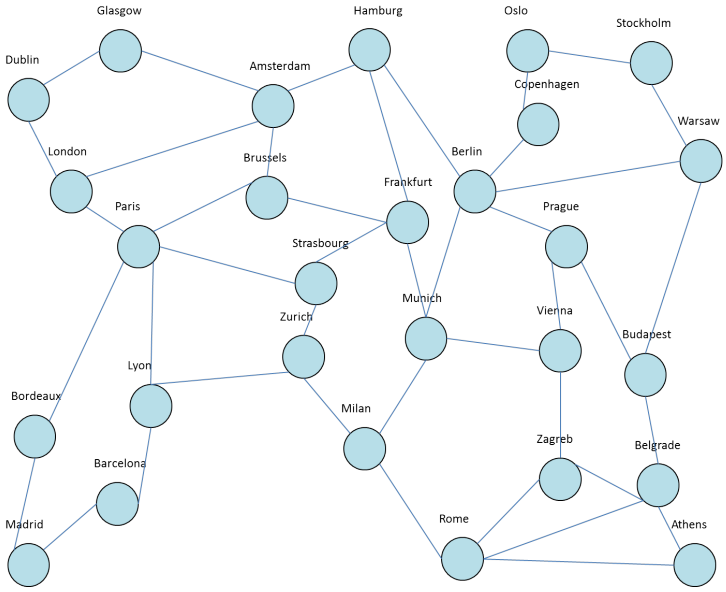


Figure 6.2: Nobel-eu network

The details of these modified instances are given in Table 6.3, which shows the number of nodes $|V|$, number of edges $|E|$ and their capacities b_e and the total number of demands $|D|$ for each instance. The edge capacities in each instance are kept identical for the purpose of testing, but variable edge capacities can also be used.

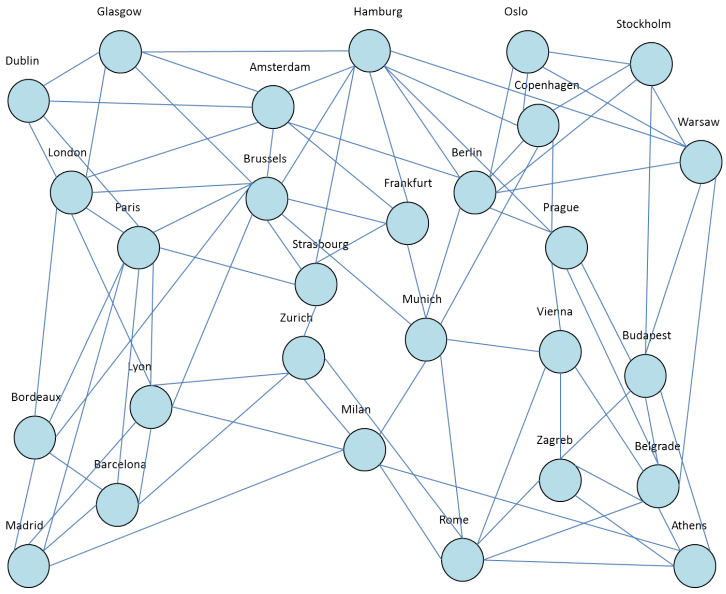


Figure 6.3: Europe28 network

Table 6.3: Network topologies, demands, and capacity per link

instance	$\|V\|$	$\|E\|$	$\|D\|$	b_e
germany17	17	52	121	60
europe28	28	84	378	120
germany50	50	157	662	100

Computational environment

We implemented the E1 formulation with the modelling language ZIMPL [81] and used IBM ILOG CPLEX 12.1 [74]. All computations were carried out on a Linux machine with 2.93 GHz Intel Xeon W3540 CPU and 12 GB RAM. A time limit of 2 hours was set for solving each problem instance. If not stated differently, all other solver settings were left at their defaults.

Furthermore, the paths for each demand are generated by the path finding algorithm (1), presented in the previous section. To guarantee that the

problem is not infeasible because of the unavailability of two disjoint paths – which is the minimum requirement – the path generating algorithm ensures that the path set fulfills this requirement.

Case 1: disjoint paths only

We conducted experiments using E1 formulation with a maximum of 2, 3, 4, or 5 available disjoint paths for each demand (for some node-pairs less than 5 disjoint paths exist, in which case only those are taken). Since the path set is already disjoint, so the disjointness constraint in E1 ($\sum_{p \in P_d} \delta_{edp} \chi_{dp} \leq 1$) does not have any effect.

Because of (6.1), a maximum number β of paths can be selected within the optimization:

$$\sum_{p \in P_d} \chi_{dp} \leq \beta \tag{6.1}$$

It may be noted that when $\beta = |P_d|$, the path selection is not required. Therefore, to evaluate the effect of path selection, we compare this case with the other cases where β and $|P_d|$ are different. The results of this testing are shown in Table 6.4. For each of the test instances ρ_{max} values obtained for different combinations of β and $|P_d|$. An optimality gap is reported when the problem could not be solved to optimality in 2 hours.

Table 6.4: Optimal/best congestion values and optimality gaps for different combinations of number of disjoint paths and maximum number of selected paths

		disjoint paths model									
# paths $\lvert P_d \rvert$		2	3	4	5	3	4	5	4	5	5
max # paths β		2	2	2	2	3	3	3	4	4	5
germany17	$\rho_{\max}$	1.4333	0.9333	0.8333	0.8333	0.9333	0.6481	0.6208	0.6481	0.6185	0.6185
	opt. gap	0.0%	0.0%	0.0%	25.8%	0.0%	0.0%	0.4%	0.0%	0.0%	0.0%
europe28	$\rho_{\max}$	1.2472	0.9945	1.0369	1.0514	0.975	0.8458	0.8426	0.8426	0.8426	0.8426
	opt. gap	0.0%	0.2%	6.3%	7.8%	0.0%	0.4%	0.0%	0.0%	0.0%	0.0%
germany50	$\rho_{\max}$	1.3100	0.8325	0.8192	0.8078	0.8067	0.7325	0.7325	0.7325	0.7325	0.7325
	opt. gap	0.0%	0.0%	10.6%	9.3%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

For all the three instance, when $|P_d|$ was greater than β, an improvement in ρ_{max} values was achieved. In particular, the case where $\beta = 2$, i.e., we want to split demands in at most two paths, a dramatic reduction in ρ_{max} was observed when $|P_d|$ was increased to 3. The cases with $|P_d| = \beta = 2$, the ρ_{max} was greater than 1, meaning that some of the links had congestion. However, when we kept $|P_d| = 3$ the ρ_{max} reduced significantly.

A further decrease can be observed if more paths are available for selection. Typically, choosing at most three paths out of five for each demand is sufficient to obtain the lowest congestion values. These findings are in line with the results of Menth et al. [88].

In addition to reducing the maximum link utilization, we also observed a load balancing over the network. The results suggest that if there are more disjoint paths than the number of paths to be selected, the load distribution seems to optimize the available capacity and thus we get the desired effect of balancing the network load over all links.

We show the results from each instance one by one.

germany17 instance

Fig. 6.4 - Fig. 6.6 show the maximum and minimum utilization of link capacities in any failure scenario for germany17 instance. These graphs also show the average utilization along with the ρ_{max} values.

In Fig. 6.4 we can observe the case of multipaths discussed by Menth et al. [88] in which the disjoint paths were pre-determined in Fig. 6.4. A higher utilization of very few links resulted in increasing the ρ_{max}, although the rest of the edges remained under-utilized even in the worst-case. However, when the path selection was introduced, a better utilization of capacities was

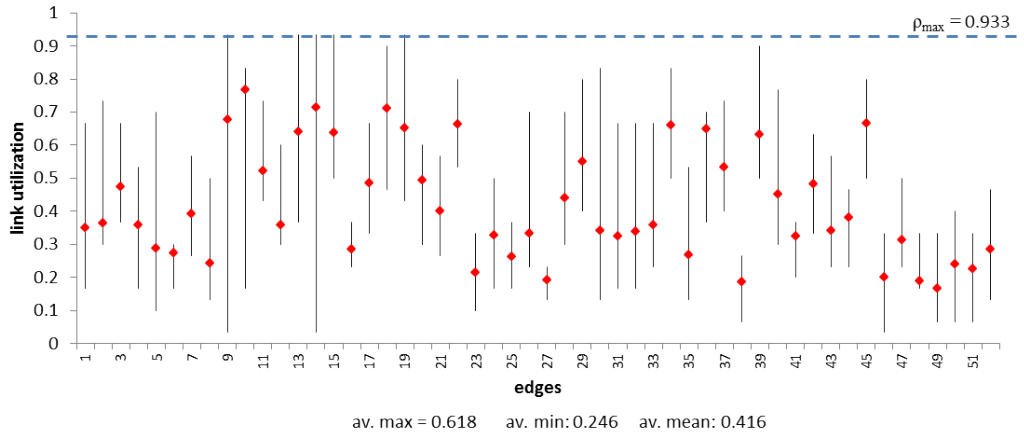


Figure 6.4: Maximum link utilization over all scenarios for germany17 with $|P_d|$=3, $\beta = 3$

observed. With an additional one disjoint path to choose from, the overall link utilization came down from 0.933 to 0.6448, which is a significant reduction of 31% in the worst-case utilization. When $|p_d|$ was increased to 5, we noticed a further reduction of 5% in the value of ρ_{max}. Another important finding is that when the path selection was introduced, in almost all the instances, the higher %age of demands used fewer paths. This result is shown in Table 6.5. When $|P_d|= \beta = 3$, 81% of the demands were routed on 3 paths, where only 65% of demands used 3 paths with $|P_d|= 4$. Hence with path selection, not only the worst-case link utilization was reduced, the demand splitting was also reduced, which is desirable for more efficient network management.

The graphs also report the mean values over minimum and maximum link utilizations. We can see that in the case of $|P_d|= 3$, the average max link utilization was much lower than the ρ_{max}, which means most of the links were not used to their maximum. The gap between ρ_{max} and average max link utilization reduced when path selection was used. It may be noted that average of average link utilization stayed almost the same in all the three cases

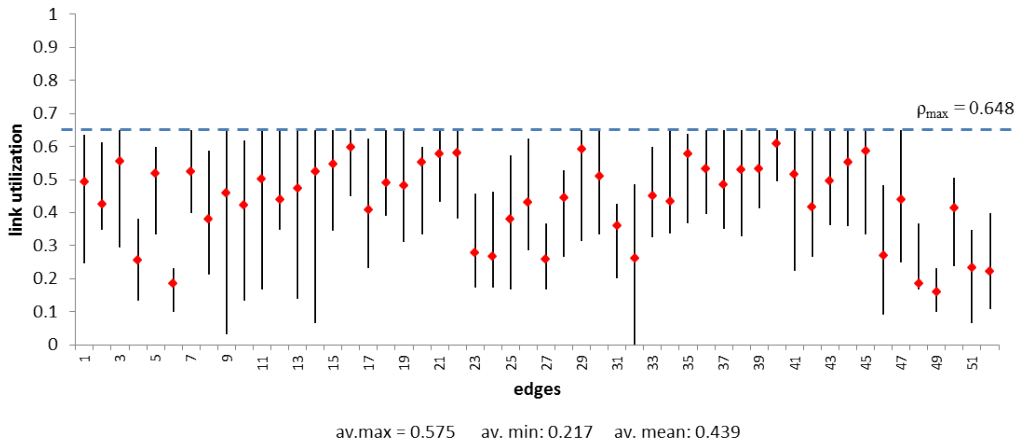


Figure 6.5: Maximum link utilization over all scenarios for germany17 with $|P_d|$=4, $\beta = 3$

($|P_d|$= 3, 4 and 5. The impact on minimum link utilization was not clear.

Table 6.5: Number of paths used by different demands for germany17 instance

# $|P_d|$	β	ρ_{max}	# of paths used	%age of demands
3	3	0.933	3	81
			2	19
4	3	0.648	3	65
			2	35
5	3	0.6185	3	93
			2	7

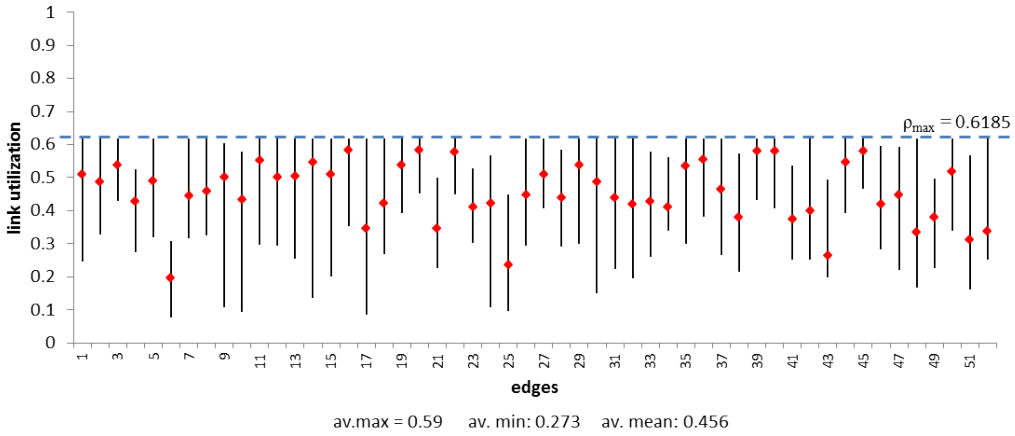


Figure 6.6: Maximum link utilization over all scenarios for germany17 with $|P_d|$=5, $\beta = 3$

Fig. 6.7 shows maximum link utilization in all the failure scenarios where $\beta = 3$ and $\|P_d|$ ranges from 3 to 5. We observed that without path selection, in only 11 out of 53 scenarios (20%), did the link utilization go up to ρ_{max} (0.933), suggesting an inefficient utilization of capacity. However, with one more path to choose from, in almost all the scenarios, the links were utilized more effectively. The maximum link utilization in the case of no-failure was also reduced from .3 to .225, when the path selection was applied.

europe28 instance

Fig. 6.8 - Fig. 6.10 show the maximum and minimum utilization of link capacities in any failure scenario for europe28 instance.

Again, these graphs show the cases where maximum number of paths $|P_d|$ was 5,4, and 3. β was kept at 3. The case where $|P_d| = \beta = 3$, the ρ_{max} value was 0.975 as shown in Fig. 6.8. When the path selection was introduced, link capacities were better utilized. It can be seen that, with $|P_d| = 4$ and $\beta = 3$, the overall link utilization reduced by 13% coming down from 0.975 to

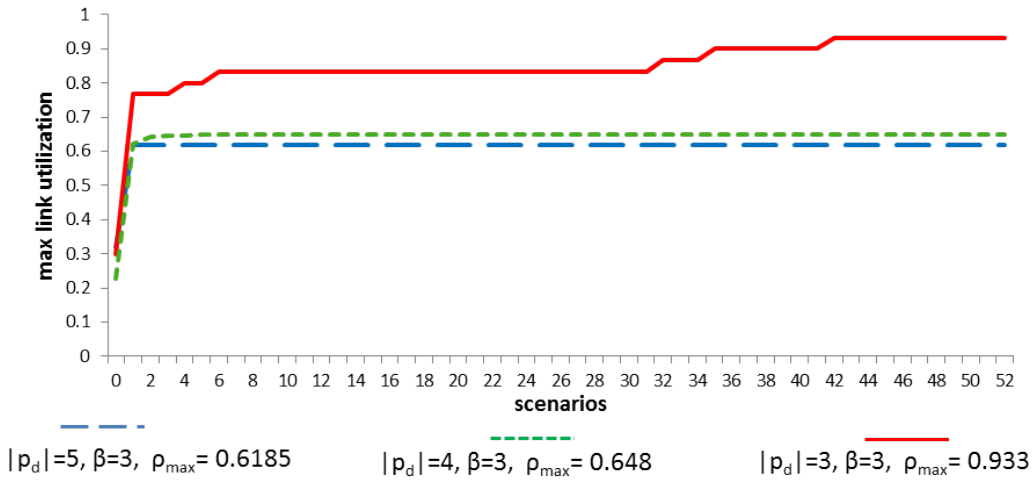


Figure 6.7: Maximum link utilization in different failure scenarios for germany17 instance

0.8426. When $|p_d|$ was increased to 5, there was no further reduction in the value of ρ_{max} (See Fig. 6.10).

When we compared the number of paths used by different demands, as shown in Table 6.6, we observed that with path selection, more demands used two paths, instead of three. It may be noted that when $|P_d|= 5$, there was no further decrease in ρ_{max}, but more demands used three paths as compared to the case of $|P_d|= 4$. As the only objective of optimization to reduce the worst-case link utilization, there was no constraint on splitting the demand upto β. In order to gain a insight, one might need to do a polytope analysis. It might be a case of degenerate solution or multi-solutions.

If there are multiple solutions with the same ρ_{max} and such optimal solutions are not too many, the network administrators can select the solution that suits their requirements. However, if we desire to minimize the splitting while optimizing, we need to solve the optimization problem in two stages. In the first stage one would minimize the ρ_{max}, while the secondary objective

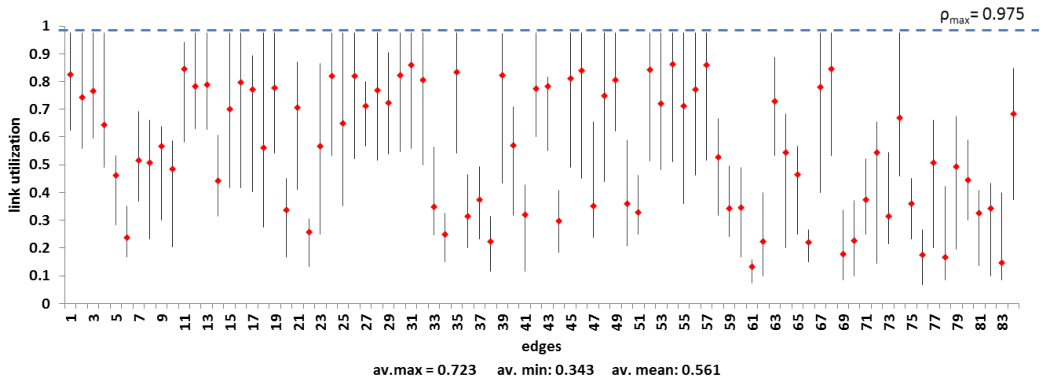


Figure 6.8: Maximum link utilization over all scenarios for europe28 with $|P_d|$=3, $\beta = 3$

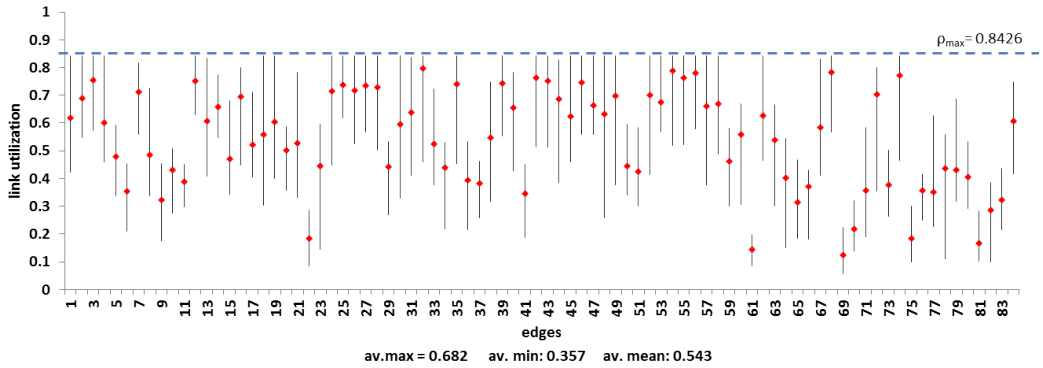


Figure 6.9: Maximum link utilization over all scenarios for europe28 with $|P_d|$=4, $\beta = 3$

would be to find the solution with the minimum demand splitting (defined over all demands) for a fixed ρ_{max}.

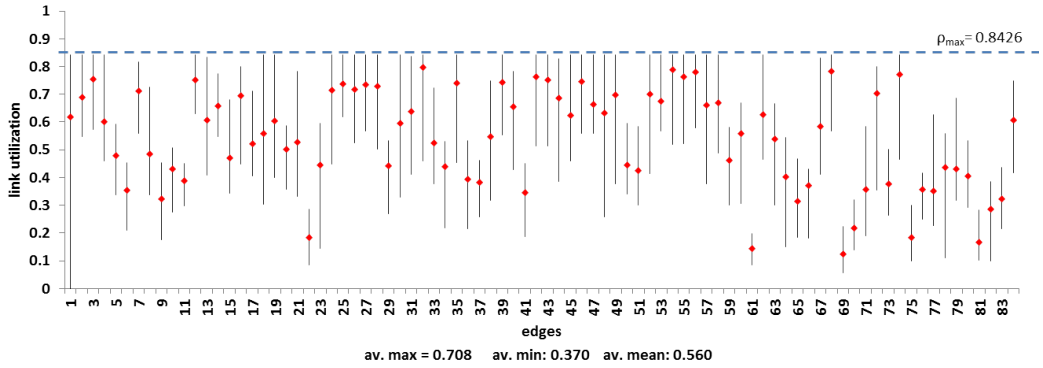


Figure 6.10: Maximum link utilization over all scenarios for europe28 with $|P_d|$=5, $\beta = 3$

Table 6.6: Number of paths used by different demands for europe28 instance

# $\|P_d\|$	β	ρ_{max}	# of paths used	%age of demands
3	3	0.975	3	84
			2	16
4	3	0.8426	3	71
			2	29
5	3	0.8426	3	83
			2	17

When the link utilization over each scenario was considered, we observed in Fig. 6.11, that shows maximum link utilization in all the failure scenarios with $\beta = 3$ and $|P_d|$= 3 to 5, that the cases where path selection was allowed, the max utilization was lower at every probability level and therefore probabilistically dominated the case without path selection.

germany50 instance

Fig. 6.12 - Fig. 6.14 show the maximum and minimum utilization of link capacities in any of the failure scenarios for germany50 instance.

We observe the similar results again in this instance as well. The graph

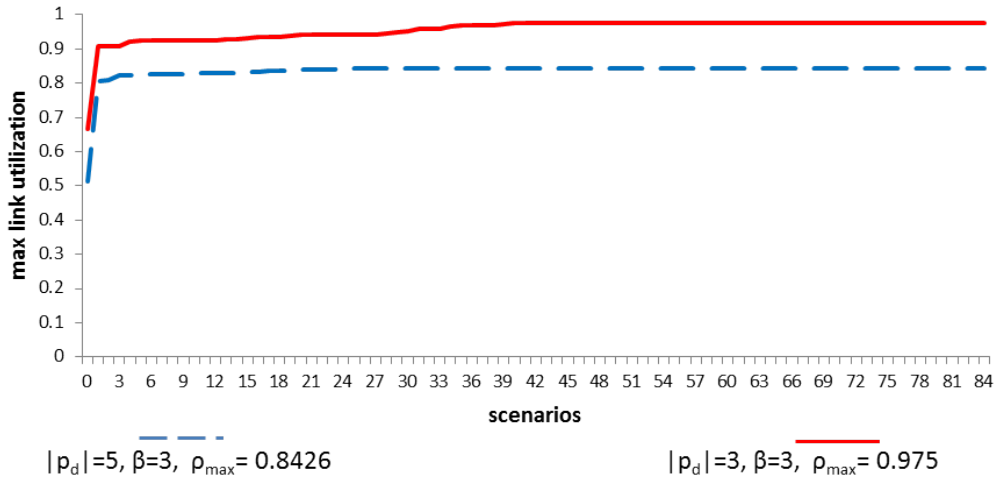


Figure 6.11: Maximum link utilization in different failure scenarios for europe28 instance

shown in Fig. 6.12 shows the link utilization when $|P_d|= 3$ and $\beta = 3$. When $|P_d|= 4$ and $\beta = 3$, as shown in Fig. 6.13, the overall link utilization reduced by 9% and came down from 0.8066 to 0.7325. When $|p_d|$ was increased to 5 (See Fig. 6.14), no further reduction in the value of ρ_{max} was observed.

When we compare the number of paths used by different demands as shown in Table 6.7, we observed that without path selection, all most all of the demands were splitting among three paths. With path selection, 19% of the demands were split in 2 paths. This shows that with path selection, a better traffic distribution plan can be identified. For $|P_d|= 5$, we compare the paths used in the cases of $\beta = 3$ and $\beta = 4$. When we allowed the demands to be split to up to 4 paths, 60% of the demands used 4 paths with the same worst-case link utilization. We need more insight to understand and explain this phenomenon.

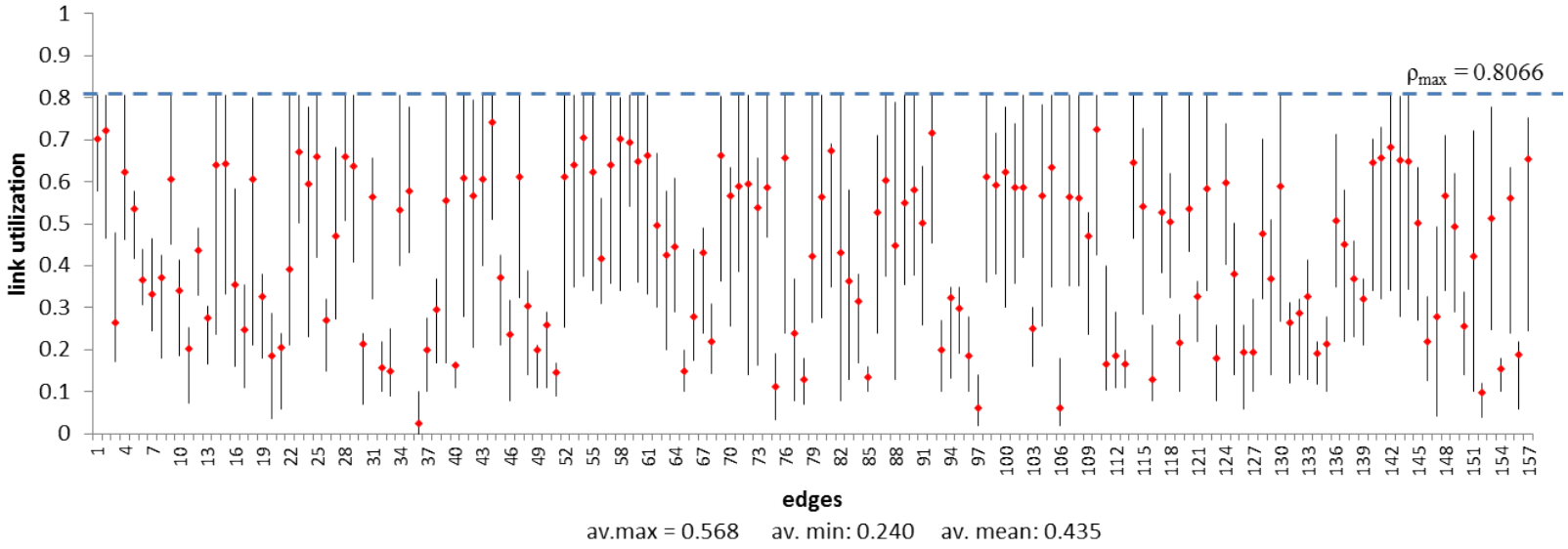


Figure 6.12: Maximum link utilization over all scenarios for germany50 with $|P_d|$=3, $\beta = 3$

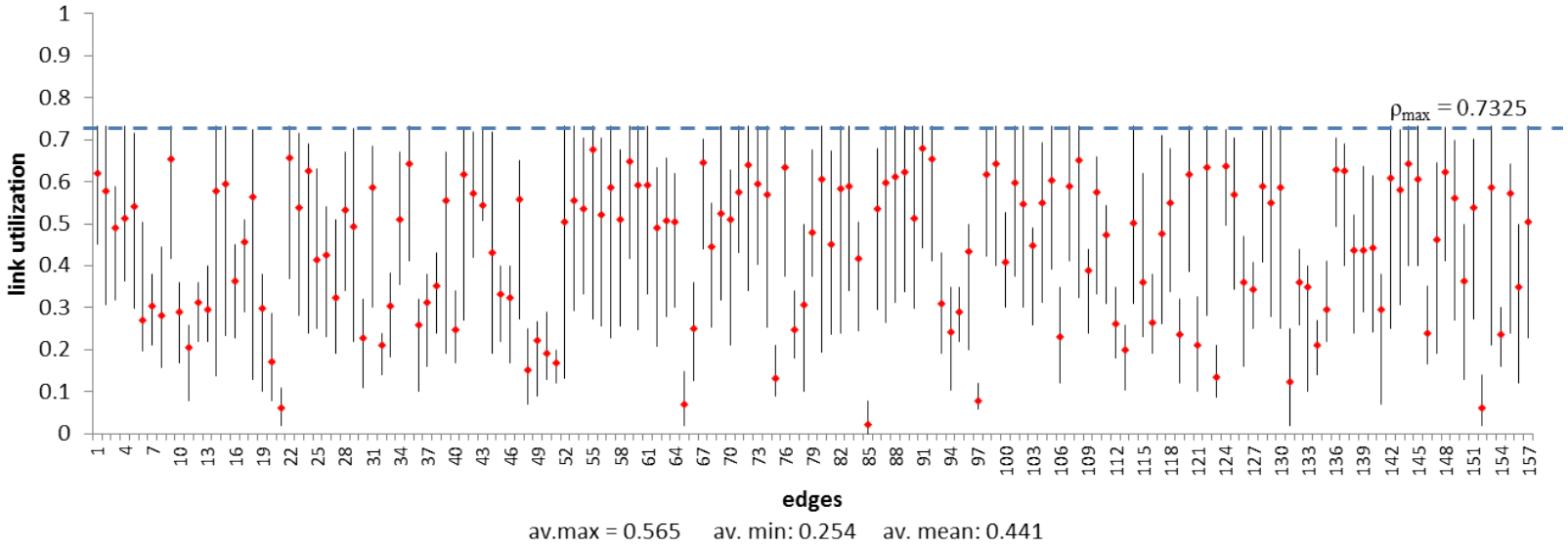


Figure 6.13: Maximum link utilization over all scenarios for germany50 with $|P_d|$=4, $\beta = 3$

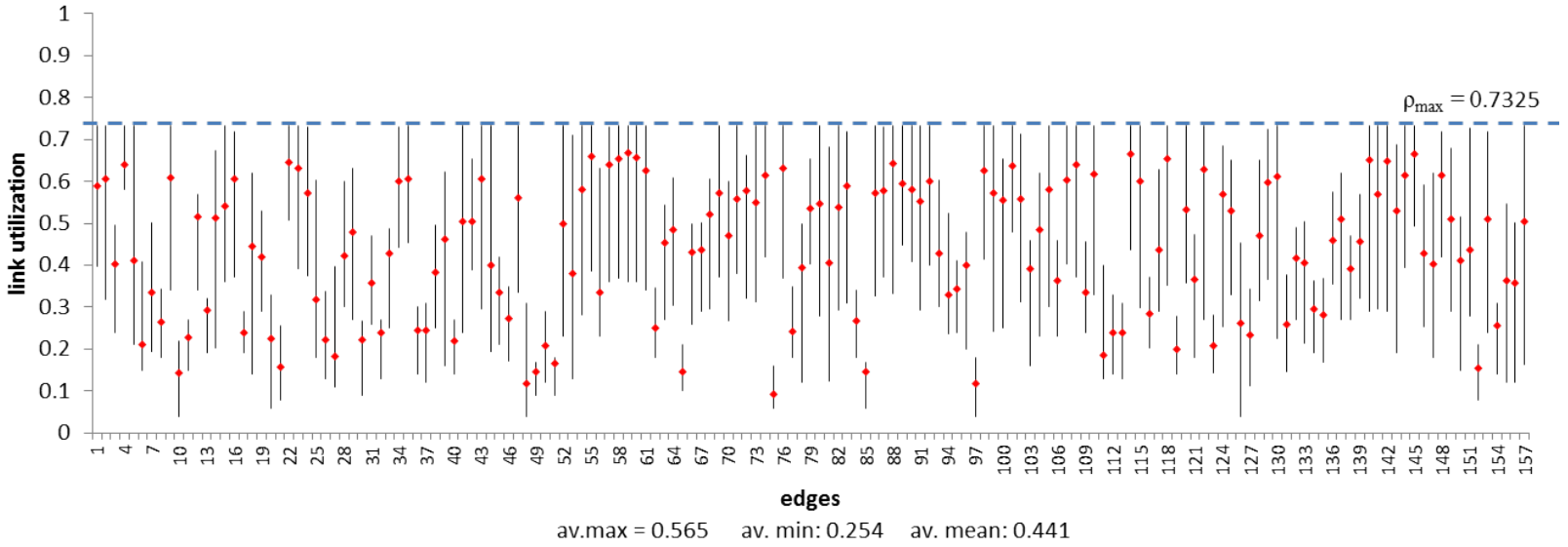


Figure 6.14: Maximum link utilization over all scenarios for germany50 with $|P_d|$=5, $\beta = 3$

Table 6.7: Number of paths used by different demands for germany50 instance

# $\lvert P_d \rvert$	β	ρ_{max}	# of paths used	%age of demands
3	3	0.8066	3	83
			2	17
4	3	0.7325	3	84
			2	16
5	3	0.7325	3	81
			2	19
5	4	0.7325	4	60
			3	31
			2	9

For germany50 instance, when the link utilization over each scenario was considered, as shown in Fig. 6.15, we observed that the $|P_d|$= 5 case probabilistically dominated the others as it has a lower max utilization at every probability level.

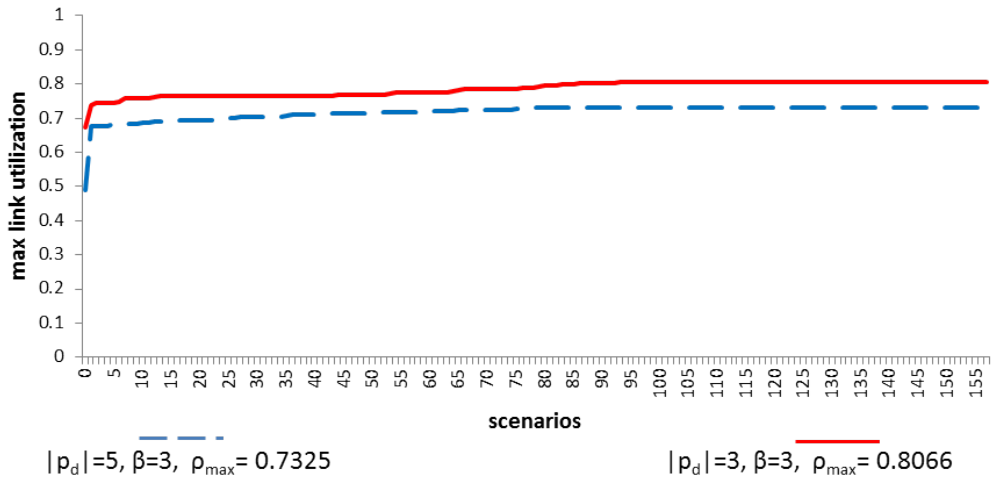


Figure 6.15: Maximum link utilization in different failure scenarios for germany50 instance

Case 2: Non-disjoint path sets

In case 2, we increased the size of path sets from 5 to 10 paths for each demand. These paths are not necessarily disjoint. Hence, disjointness constraints now play an important role in selecting disjoint paths required by SPM. Again, we compare results if we allow for selecting at most β paths per demand by constraints (6.1).

Table 6.8: Best congestion values, dual bound, gap, and CPU times for 10 paths/demands and variable maximum number of selected paths

$\rho_{\max}$		non-disjoint paths model				
# paths $\lvert P_d \rvert$		10	10	10	10	10
max # paths β		2	3	4	5	10
germany17	primal	0.9333	0.8833	0.6325	0.6193	0.6185
	dual	0.6185	0.6185	0.6185	0.6185	0.6185
	gap/time	33.7%	30.0%	2.2%	0.1%	1260 s
europe28	primal	0.9625	0.8600	0.8426	0.8426	0.8426
	dual	0.8426	0.8426	0.8426	0.8426	0.8426
	gap/time	12.4%	2.0%	4980 s	4669 s	3928 s
germany50	primal	0.9600	0.7800	0.7325	0.7325	0.7325
	dual	0.7325	0.7325	0.7325	0.7325	0.7325
	gap/time	23.7%	6.1%	3800 s	2324 s	1924 s

In almost all cases, the computation of the linear programming relaxation turned out to be extremely difficult with the default CPLEX settings (more than two hours of CPU time). For the reported results we therefore changed the linear relaxation algorithm to *primal simplex* instead of *automatic*. Nevertheless, not all instances could be computed to optimality within the time limit. The results are shown in Table 6.8.

Independent of the value of β, the dual bound after two hours of CPU time is the same and equals the value of both the LP relaxation and the best

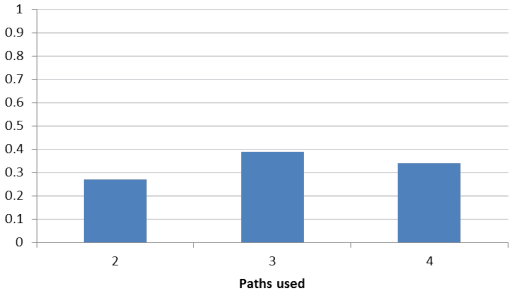
known solutions of our first study, cf. Table 6.4. However, for smaller values of β computing good primal solutions turned out to be the bottleneck of the optimization. Since the same difficulty is also observed in the disjoint case, it is difficult to judge which of the two bounds has to be improved most.

Regardless of these computational difficulties, one can observe that, with ten paths, no better results can be expected than with five disjoint paths in the first study. Whether this implies that the reported utilization values cannot be improved further remains open. To show this, all paths should be considered. Given the computational difficulties with ten paths per demand, this is only possible if a column generation (with branch-and-price) approach is applied.

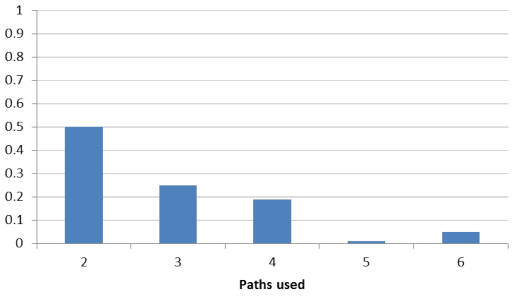
Finally, Fig. 6.16 shows a histogram of the number of used paths per demand for two germany17 cases: up to four out of five disjoint paths (case 1) and up to 10 out of 10 (non-disjoint) paths. Clearly, in the first case two, three, or four disjoint paths can be selected. In 65% of the demands less than the allowed four paths are used, in more than 70% more than two. In the second case (with the same maximum utilization) up to six paths are used. However, more than 50% of the demands only need two paths and almost 80% less than four paths, easing the management of the network. These results show that optimization of the path selection is flexible and provides a good opportunity to simplify network management as well.

6.2.2 A comparison between E1 and E2 models

In this section we present a comparative analysis of E1 and E2. Table 6.9 shows a comparison between the number of variables and constraints for the



(a) $|P_d|$= 5, $\beta = 4$



(b) $|P_d|$= 10, $\beta = 10$

Figure 6.16: Number of paths used per demand

three test cases where $|P_d| = \beta = 2$. Even for this simple case, the problem size for E1 is considerably larger than E2.

Table 6.9: Comparison of number variables and constraints in test instances

Instance	Variables	Constraints
germany17 E1	13069	48729
germany17 E2	969	6301
europe28 E1	65017	239762
europe28 E2	3025	18830
germany50 E1	210517	768848
germany50 E2	5297	45204

We have also compared the performance of both of the formulations empirically. For this purpose, the experiments were based on a C++ implementation that used CPLEX 12.3. The testing was done on a virtual machine with Windows 7 2.5 GHz Intel quard-core CPU and 12 GB RAM, which was different from the one we used for the experimentation presented in the previous section 6.2.1. A time limit of 2 hours was set for solving each problem instance. If not stated differently, all other solver settings were left at their defaults. The results are given in Table 6.10. The path set consisted of 10 paths per demand for all the three instances. Again the maximum number of paths selected was limited by β number of paths per demand. The dual values obtained are also reported. Finally, the third row states the gap between primal and dual or the computational times for each method. "−" denotes that in the given time, no integer solution was obtained.

None of the instances were solvable with E1 in the stipulated time. With E2 method, we could solve cases for $\beta \geq 4$ to optimality for germany17 and europe28. Germany50 instance was not solvable by any of the formulations.

Table 6.10: Optimal/best congestion values and optimality gaps for different combinations of number of paths and maximum number of selected paths

#paths $\|P_d\|$		10	10	10	10	10	10	10	10	10	10
max paths		2	2	3	3	4	4	5	5	10	10
method		E1	E2	E1	E2	E1	E2	E1	E2	E1	E2
germany17	primal	-	0.622	0.983	0.622	0.650	0.6185	0.6195	0.6185	0.6266	0.6185
	dual	-	0.6185	0.6185	0.6185	0.6185	0.6185	0.6185	0.6185	0.6185	0.6185
	gap/time	-	0.56%	37%	0.56%	4.8%	4387	0.16%	268 s	1.29%	562 s
europe28	primal	-	-	-	-	-	0.842	-	0.842	-	0.842
	dual	0.842	0.8426	0.8426	0.8426	0.8426	0.8426	0.8426	0.8426	0.8426	0.8426
	gap/time	-	-	-	-	-	4121 s	-	1725 s	-	1124 s
germany50	primal	-	27.55	-	27.75	-	27.55	-	27.55	-	0.732
	dual	-	-	-	0.7325	-	0.7325	-	0.7325	-	0.7325
	gap/time	-	-	-	96.6%	-	96.6%	-	96.6%	-	7200 s

But, we can see that for all the considered cases, the performance with E2 formulation is much better than that with E1 formulation.

6.2.3 A comparison of multipath and 1+1 methodology

To make a comparison between our proposed methodology with other existing methodologies, we tested the performance of our approach of multipath routing with the other approaches based on the solutions of benchmark instances. For this experimentation *dfn-bwin* and *di-yuan* instances from sndlib [104] were used. Both of these instances are solved to find the network routing with 1+1 protection and the results are reported on sndlib website. In case of *dfn-bwin*, the solution for instance *dfn-bwin U-U-E-N-S-A-N-P* was used and in the case of *di-yuan*, the solution for instance *di-yuan U-U-E-N-S-A-N-P* was used.

The results are given in Table 6.11. Since these instances are solved to find routing to provide 1+1 protection, meaning that two paths are allowed for each route. Hence, we allowed 2 paths for each demand (i.e.: $\beta = 2$).

For the capacities identified in the solution reported on sndlib website, the maximum link utilization over all states is 99.95%. With our method (E2 formulation), the maximum link utilization obtained was 61.9%, which clearly demonstrates the potential of SPM compared to 1+1 protection.

In the case of *di-yuan*, the maximum link utilization over all states is 100%. When we solved this problem using demand bifurcation, we allowed 2 paths for each demand and the maximum link utilization of 95% was achieved.

Table 6.11: Solution statistics for *di-yuan*instance

	Solution	Nodes	Links	Demands	Cost	Installed Cap	$\rho_{\max}$
dfn-bwin	sndlib	10	24	90	242025	2885000	99.9%
dfn-bwin	E2	10	24	90	242025	2885000	61.9%
di-yuan	sndlib	11	17	22	1534700	284	100%
di-yuan	E2	11	17	22	1534700	284	95%

6.3 Summary

In this chapter we have presented a path generation process that was adopted to form the path sets. The algorithm can be altered to generate different path sets of specific characteristics.

Using these path sets, we have conducted a number of experiments to assess the efficiency of our proposed methodology. We compared the cases where multipaths are pre-determined with the cases in which path selection is part of the optimization process. This endogenous path selection process introduces complexity into the problem. Nevertheless, it yields better utilization of link capacities. The multipath approach combined with this path selection feature results in a more balanced network utilization – thus making more capacity available for additional use.

We also compared the link utilization of our proposed methodology with 1+1 protection methodology using the benchmark cases. Our multipath approach required less capacity to provide protection against all single link failures than 1+1 protection scheme.

We also empirically compared the performance of two ILP models that were presented in Chapter 5 and observed that E2 formulation always outper-

formed E1 formulation. It was mainly due to the fact that E2 model resulted in much fewer number of variables and constraints as compared to E1 formulation.

7

Solution Methods for the Routing Problem

In this chapter we will explore the options to improve the solution quality that we have achieved so far. We look into the possibility of using other exact or heuristic approaches to solve our problem. Since ILP- based method on its own is not sufficient to solve the larger problems, we propose a hybrid approach, where a meta heuristic guides the search space for ILP. We will present the hybrid algorithm and the empirical results comparing the performance of this

hybrid approach with the two methods (E1 & E2) based on ILP framework, which we have already looked at in the previous chapter.

We will also discuss the column generation approach wherein we decompose the larger problem into many smaller subproblems and use a pricing methodology to add those variables into the problem space that can only improve the solution quality.

7.1 Hybrid Approach

As we have discussed earlier in Chapter 3 in detail, heuristic approaches are being successfully used in combination with exact approaches to solve the difficult problems. Although we cannot obtain a guarantee for optimality, this hybridization results in very efficient algorithms. From a managerial point of view, this approach provides a realistic and useful insight into the problem at hand, which can facilitate the decision-making process.

After exploring different possibilities of applying a heuristic approach, we have come to the conclusion that it is best to guide our ILP through a heuristic algorithm that helps the ILP by limiting the search space. It does so by adding only those variables (paths) in the search space that can potentially reduce the overall link utilization while removing those that are not useful for the optimal solution. This approach is similar to the one used in column generation, where new variables are added deterministically, while here we use heuristics to add new columns (variables). The heuristic goes one step further and also removes redundant variables. In this section we present our hybrid approach, while the column generation based exact approach is discussed in the next section.

7.1.1 Description of the Heuristic Algorithm

The algorithm assumes that a large number of candidate paths for each demand are available and can be used. The selection of the paths requires an ILP. However, adding all the paths initially in the ILP will make the problem prohibitively large and it will become unsolvable. Therefore, instead of introducing all the paths in the problem space, our proposed heuristic starts with a very small number of paths (≥ 2) per demand, which are disjoint.

Initially we do not need to make a path selection, as all the paths are already disjoint, so all we need is to run an LP problem to identify the optimal load distributions for the given paths. For this,we use E2 formulation described in Chapter 5 after relaxing the path selection imposed through disjointness constraints. The algorithm is presented formally as Algorithm (2). Let M denote the maximum number of paths available per demand. β_d denotes the number of paths used for demand d, while k is the maximum number of paths, a demand can be split into. Initially, to generate an LP solution all demands are split on a randomly selected k out of M paths for each demand. Hence $\beta_d = k$ for all $d \in D$.

The solution thus obtained can tell us which edge(s) is being maximally used. We select the edge with the highest link utilization (under any failure scenario). Let us call this edge e_{max}. If there are more than one edges with the same highest utilization, then the edge with the smaller index number is selected as e_{max} (for example, if e_3 and e_5 have the highest utilization, e_3 will be selected as e_{max}). Of course, the link utilization of e_{max} would be the same as our objective function value ρ_{max} in at least one of the scenarios.

The load allocated to e_{max} determines the quality of the solution. The

burden on e_{max} can be reduced by re-distributing its load to other paths, which are not using e_{max}. Its re-distribution can potentially reduce the maximum link utilization of the network, which is our objective function. Once we have identified the set of demands D_{max} that use edge e_{max}, we introduce more paths for these demands in the ILP by adding all the remaining $M - k$ paths. The ILP will now select paths from M available paths for each $d \in D_{max}$, while for the rest of the demands $d \in D \backslash D_{max}$, the load is distributed on the available paths. This way we considerably reduce the search space for ILP. The resulting solution will either improve or will stay the same.

This process is repeated until the same edge is identified as e_{max} twice, consecutively with the same link utilization value. This would mean that no further improvement can be expected, as the local optimization cannot further reduce the maximum utilization on this edge. Hence we stop the algorithm. It may be noted that each improvement step solves an ILP exactly to see whether the maximum utilization can be reduced by re-routing some traffic from e_{max} to other edges. Therefore the solution can never get worse from one step to the next. Figure 7.1 shows the framework that combines the exact method with a heuristic approach.

7.1.2 Experimental Results

Using the same test instances that we have used earlier in the previous chapter, we have tested the performance of the proposed hybrid approach in comparison with E1 and E2 methods. Table 7.1 shows the results of our experiments. We implemented the MILP formulation with IBM ILOG CPLEX 12.3 [75] concert technology with C++. All computations are carried out on a virtual machine

Algorithm 2 mimimze ρ_{max}

INPUT M, k
$\beta_d \leftarrow k \quad \forall d \in D$
Get solution ρ_{max} by solving LP with β_d paths for each demand d
$e_{max} \leftarrow null$
$prevEdgeIndex \leftarrow null$
$improved \leftarrow true$
repeat
  if $e_{max} \neq null$ **then**
    $prevEdgeIndex \leftarrow e_{max}$
  end if
  Identify edge e_{max} with highest link utilization
  Identify all demands D_{max} with active paths passing through e_{max}
  $\beta_d \leftarrow M \quad \forall d \in D_{max}$
  $\beta_d \leftarrow k \quad \forall d \in D \backslash D_{max}$
  Get new solution ρ'_{max} by solving ILP
  if $(\rho_{max} == \rho'_{max})$ **then**
    $improved \leftarrow false$
  end if
  $\rho_{max} \leftarrow \rho'_{max}$
until $(prevEdgeIndex == e_{max}\ and\ !\,(improved))$
return ρ_{max}

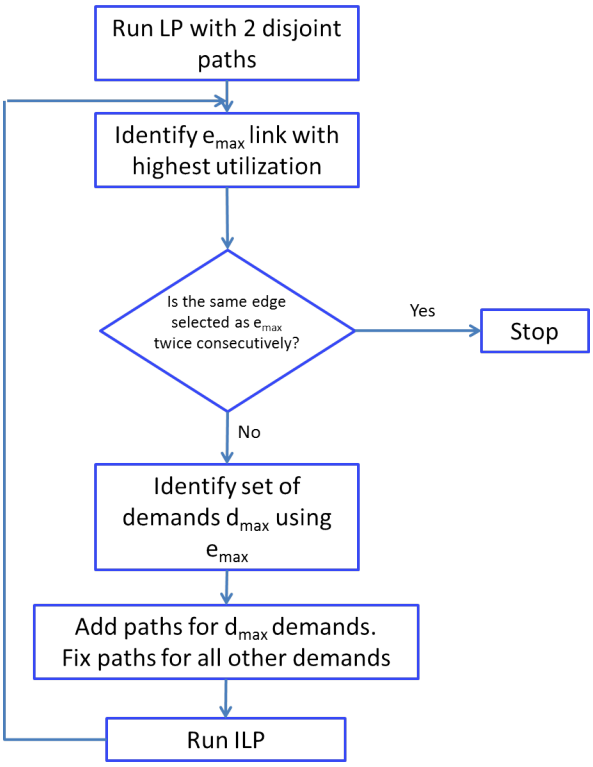


Figure 7.1: Framework of the hybrid approach

with Windows 7 2.5 GHz Intel quard-core CPU and 12 GB RAM. A time limit of 2 hours was set for solving each problem instance. If not stated differently, all other solver settings were left at their defaults.

Table 7.1: Optimal/best congestion values and optimality gaps for different combinations of number of paths and maximum number of selected paths

#paths $\|P_d\|$	10	10	10	10	10	10	10	10	10	10	10	10
max # paths β	2	2	2	3	3	3	4	4	4	5	5	5
method	E1	E2	H	E1	E2	H	E1	E2	H	E1	E2	H
germany17	-	0.622	0.833	0.983	0.622	0.7833	0.650	0.6185	0.6185	0.6195	0.6185	0.6185
dual	-	0.6185	-	0.6185	0.6185	-	0.6185	0.6185	-	0.6185	0.6185	-
Time(sec)	7200	7200	1830	7200	7200	7200	7200	4387	2512	7200	268	1979
europe28	-	-	1.097	-	-	0.8583	-	0.8426	0.8426	-	0.8426	0.8426
dual	0.8426	0.8426	-	0.8426	0.8426	-	0.8426	0.8426	-	0.8426	0.8426	-
Time(sec)	7200	7200	7200	7200	7200	7200	7200	4121	3368	7200	1725	5496
germany50	-	27.55	0.94	-	27.75	0.76	-	27.55	0.732	-	27.55	0.732
dual	-	-	-	-	0.7325	-	-	0.7325	-	-	0.7325	-
Time(sec)	7200	7200	3513	7200	7200	7200	7200	7200	7200	7200	7200	7200

Table 7.1 has three separate sets of rows for each of the test instances: germany17, europe28 and germany50. Within each set the first row indicates the ρ_{max} values obtained under each of the three methods (E1, E2, H) for the maximum number of splits allowed: i.e., β equals (2, 3, 4, and 5). The second row for each instance show the dual values obtained, if the problem was not solved in the given time (2 hours). It may be noted that in the case of the heuristic method (H), we cannot obtain a dual value. The computational times for each method is also indicated. "−" denotes that in the given time, no integer solution was obtained. For these experiments, the hybrid algorithm used a maximum of 10 paths per demand.

For *germany17*, the results of the exact formulation were better for maximum number of paths (β) less than or equal to 3. With $\beta \geq 4$, the computational time required by the heuristic algorithm was significantly less than that of the exact method. For larger instances, the hybrid approach resulted in a better solution than ILP alone. For *germany50*, near optimal solutions were obtained when paths allowed were 3 or more. A similar pattern was seen for *europe28*. For $\beta = 2$ or 3, the exact method could not find any solution. The solution obtained for 2 paths with the heuristic method was relatively far from the dual value, but with 3 paths, a solution close to the dual value was obtained. With $\beta \geq 4$, both methods could reach the optimal solution, but the computational time was less with the heuristic approach.

What we observed in the experimentation is that the heuristic algorithm sometimes finds an optimal solution, but because the stopping criterion is not satisfied, it keeps on iterating. It happens so because the heuristic algorithm has no knowledge of the possible lower bound. One possible way to avoid unnecessary iterations is to find the lower bounds through LP relax-

ation and then use these values to stop the hybrid algorithm. This way, the computational times of hybrid method may be reduced further.

Effect of different demand sizes

Finally, we also tested the effect on the computational time with the increase in burden on the network. Using *europe28* as a test case, we tested all the three methods: E1, E2 and Hybrid method for different number of demands. As it can seen in Figure 7.2, the increase in the computational time is most rapid in E1 while E2 and Hybrid method shows a slower increase in the computational time.

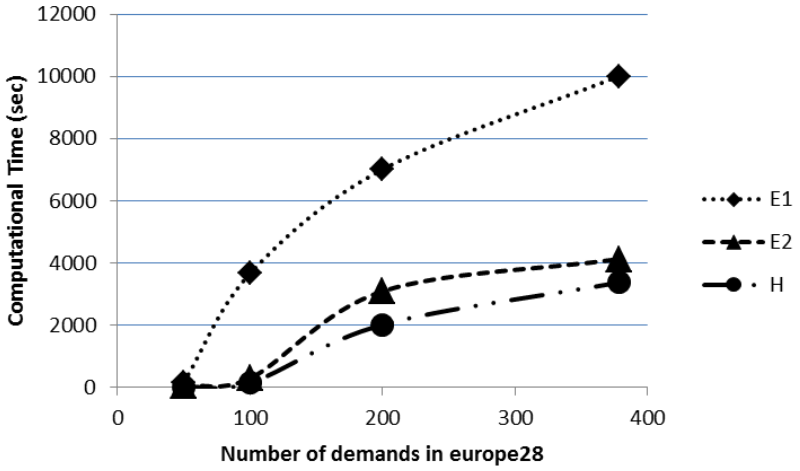


Figure 7.2: Computational time Vs number of demands

7.2 Column Generation

In this section we discuss the column generation (CG) method, also known as the branch and price algorithm to help us solve our problem. The use of dynamic CG for solving large problems is already well-known. See Chvátal

[26]. CG is typically applied to problems with a large number of variables (columns), where the number of constraints remains fixed. In case of an ILP, if the LP solution is not integral, one option is to follow a framework called IP CG or Branch and Price Algorithm (Wolsey [139]). The pictorial representation of the Framework is given in Figure 7.3. It works in collaboration with the branch and bound method to solve LPs. First, we reformulate our original problem as a Master Problem which is suitable for CG application. Then, considering a Restricted Master Problem (RMP) with a limited number of variables, we solve an LP Relaxation of this RMP. The dual values obtained from the solution of this primal problem are used to formulate a pricing problem. The variables in the primal problem are represented as constraints in the dual system. Then, the left out variables are priced. If the reduced cost of any of the variables is negative, i.e., the dual constraints are violated, adding them back to the the master problem might improve the solution quality. However, if no such variable exists, we can conclude that we have obtained an optimal solution for the current Master Problem. If the obtained LP solution is also integral, we have the optimal solution for our ILP. Otherwise, we need to apply a branch and bound method again and use the column generation method on each branching node. This process continues until an integer solution is found.

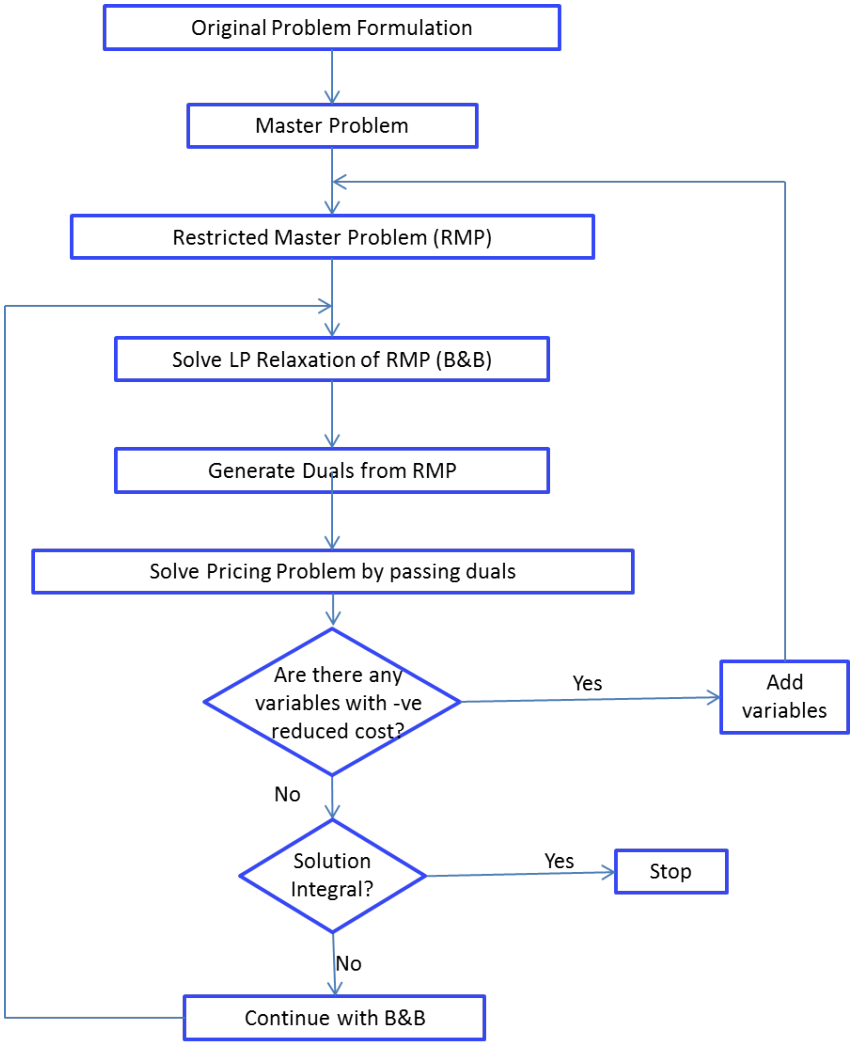


Figure 7.3: Branch and Price Framework for every B&B node

The decomposition methods have been used for network designing problems by many researchers. For example, Fortz and Poss [52] have used benders' decomposition method for survivable multilayer network design problems and Botton et al. [16] have applied it to the hop-constrained survivable network design problems. Orlowski and Pióro [103] have provided extensive discussion on the application of column generation for different network design problems with path-based survivability mechanisms. They show the complexity

of the pricing problems for different survivability concepts. In the following section, we present a column generation framework for our considered routing allocation problem.

7.3 CG Framework for Routing

In this section we present a novel formulation for resilient network routing problem using column generation method and discuss the formulation for pricing.

We start with the relaxed master problem with two disjoint paths per demand. After solving the primal problem, we use the dual information to formulate a pricing method to identify new paths that violate the dual constraints. The pricing problem can be decomposed for each demand. Hence instead of solving one large problem, we can solve one sub-problem for each demand. The violating paths can be added into the master problem at once. This process is repeated until no additional path with negative reduced cost is found; hence an optimality is ensured. Of course, for an integral solution, we need to ensure that the final solution is also integral.

We have used E1 formulation for the column generation as it is more straightforward than E2 formulation and will help us formulate a less complex pricing problem than what we will get from the E2 formulation. Also, the limitation of E1 that it has a larger number of variables might not affect its performance in column generation, since, in any case, we consider here a very limited number of columns at any one time and the resulting primal problem hopefully will not grow very large.

It may be noted that the constraints (7.1) and (7.2) were originally in

E1 formulation described in Chapter 5. Constraints (7.1) state that a path cannot be used if one of its edges fails, while (7.2) ensures that a path gets a load value only if it is one of the selected paths.

$$\sum_{s \in S : s \in p} \delta_{sdp} \ell^s_{dp} = 0 \quad \forall d \in D, p \in P_d \tag{7.1}$$

$$\ell^s_{dp} \leq \chi_{dp} \quad \forall d \in D, p \in P_d, s \in S_0 \tag{7.2}$$

In order to simplify the dual problem, we have re-formulated these constraints. Now Eq. (7.3) ensures that the load value is assigned to only those path variables that are not only on the selected paths, but are also not affected by the failure scenario. This constraint looks only at the edge-failure scenarios. The case when there is no-failure is represented by Eq. (7.4), which states that in "0" scenario (no-failure), the load can be assigned to a path if it is one of the selected ones. μ in the square brackets represent the dual variables.

$$\ell^s_{dp} \leq \chi_{dp} - \delta_{sdp} \chi_{dp} \quad \forall d \in D, p \in P_d, s \in S \tag{7.3}$$

$$\ell^0_{dp} \leq \chi_{dp} \quad \forall d \in D, p \in P_d \tag{7.4}$$

Equations (7.5) represent the complete formulation for the problem in

the standard from.

$$\min \rho_{\max} \tag{7.5a}$$

s.t.

$$[\mu^1_{e,s}] \; -\sum_{d\in D}\sum_{p\in P_d} h_d \delta_{edp} \ell^s_{dp} + b_e \rho_{\max} \geq 0 \qquad \forall e \in E, s \in S_0 \tag{7.5b}$$

$$[\mu^2_{d,s}] \sum_{p\in P_d} \ell^s_{dp} = 1 \qquad \forall d \in D, s \in S_0 \tag{7.5c}$$

$$[\mu^3_{d,p,s}] - \ell^s_{dp} + (1-\delta_{sdp})\chi_{dp} \geq 0 \quad \forall d \in D, p \in P_d, s \in S \tag{7.5d}$$

$$[\mu^4_{d,p}] - \ell^0_{dp} + \chi_{dp} \geq 0 \qquad \forall d \in D, p \in P_d \tag{7.5e}$$

$$[\mu^5_{d,e}] - \sum_{p\in P_d} \delta_{edp}\chi_{dp} \geq -1 \qquad \forall d \in D, e \in E \tag{7.5f}$$

$$[\mu^6_{d,p,s}] \ell^0_{dp} - \ell^s_{dp} + \sum_{q\in P_d} \delta_{sdq}\chi_{dq} \geq 0 \quad \forall d \in D, p \in P_d, s \in S \tag{7.5g}$$

$$[\mu^7_{d,p,s}] - \ell^0_{dp} + \ell^s_{dp} + \sum_{q\in P_d} \delta_{sdq}\chi_{dq} \geq 0 \quad \forall d \in D, p \in P_d, s \in S \tag{7.5h}$$

$$[\mu^8_{d,p,s_i,s_j}] \ell^{s_i}_{dp} - \ell^{s_j}_{dp} - \sum_{q\in P_d} \delta_{s_i dq}\delta_{s_j dq}\chi_{dq} \geq 1 \; \forall d \in D, p \in P_d, i, j \in S \tag{7.5i}$$

$$\chi_{dp} \in \{0,1\}, \ell^s_{dp} \geq 0 \tag{7.5j}$$

The dual variables corresponding to each constraint are given in square brackets. The dual system for Formulation 7.5 is given below at Eq (7.6).

7.3.1 Dual formulation

The dual of the formualtion 7.5 is given in 7.6. A set $P^{'}_{d}$ represents a set of all possible paths for the demand d:

$$\max \sum_{d\in D}\sum_{s\in S_0} \mu^2_{d,s} - \sum_{d\in D}\sum_{e\in E} \mu^5_{d,e} + \sum_{d\in D}\sum_{p\in P'_d}\sum_{i,j\in S} \mu^8_{d,p,i,j} \tag{7.6a}$$

$[\rho_{max}]$

$$\sum_{e\in E} b_e \sum_{s\in S_0} \mu^1_{e,s} \leq 1 \tag{7.6b}$$

$[\chi_{dp}]$

$$\sum_{s\in S} (1-\delta_{sdp})\mu^3_{d,p,s} + \mu^4_{d,p} - \sum_{e\in E} \mu^5_{d,e} + \sum_{q\in P'_d}\sum_{s\in S} \delta_{s,d,p}\mu^6_{d,q,s} \tag{7.6c}$$
$$+ \sum_{q\in P'_d}\sum_{s\in S} \delta_{s,d,p}\mu^7_{d,q,s} - \sum_{q\in P'_d}\sum_{e,j\in p:p\in P'_d} \delta_{e,d,p}\delta_{j,p}\mu^8_{d,q,s_e,s_j} \leq 0 \forall d\in D, \forall p\in P'_d$$

$[\ell^s_{dp}]$

$$-h_d \sum_{e\in E} \delta_{e,d,p}\mu^1_{e,s} + \mu^2_{d,s} - \mu^3_{d,p,s} - \mu^6_{d,p,s} + \mu^7_{d,p,s} \tag{7.6d}$$
$$+ \sum_{q\in S:s\neq q} \mu^8_{d,p,s,q} - \sum_{q\in S:s\neq q} \mu^8_{d,p,q,s} \leq 0 \qquad \forall d\in D, \forall p\in P'_d, s\in S$$

$[\ell^0_{dp}]$

$$-h_d \sum_{e \in E} \delta_{e,d,p} \mu^1_{e,s_0} + \mu^2_{d,s_0} - \mu^4_{d,p} - \sum_{s \in S} \mu^6_{d,p,s} + \sum_{s \in S} \mu^7_{d,p,s} \leq 0 \forall d \in D, \forall p \in P'_d \tag{7.6e}$$

$$\mu^1, \mu^3, \mu^4, \mu^5, \mu^6, \mu^7 \, \mu^8 \geq 0 \tag{7.6f}$$

$$\mu^2 : \text{free} \tag{7.6g}$$

In formulation 7.5, the constraints (7.5g), (7.5h) and (7.5i) compare the load values (ℓ) in different scenarios, making the pricing problem extremely difficult. These constraints prohibit the re-distribution of load for demands where a failure does not affect any of its paths. For the time being, we exclude these three constraints from the primal problem. The problem still remains valid - albeit less realistic. Once we succeed with column generation, without these constraints, we re-introduce them and try to re-formulate our pricing problem. However, this task would form the basis of a future research agenda, as discussed in Chapter 8.

The dual problem for formulation (7.5) without constraints (7.5g), (7.5h) and (7.5i) is given as formulation (7.7).

$$\max \sum_{s \in S_0} \mu^2_{d,s} + \sum_{e \in E} \mu^5_{d,e} \tag{7.7a}$$

$[\rho_{max}]$

$$\sum_{e \in E} b_e \sum_{s \in S_0} \mu^1_{e,s} \leq 1 \tag{7.7b}$$

$[\chi_{dp}]$

$$\sum_{s \in S} (1 - \delta_{s,d,p}) \mu^3_{d,p,s} + \mu^4_{d,p} - \sum_{e \in E} \delta_{e,d,p} \mu^5_{d,e} \leq 0$$

$$\forall d \in D, p \in P'_d \tag{7.7c}$$

$[l^s_{dp}]$

$$- h_d \sum_{e \in E} \delta_{e,d,p} \mu^1_{e,s} + \mu^2_{d,s} - \mu^3_{d,p,s} \leq 0$$

$$\forall s \in S, \forall d \in D, p \in P'_d \tag{7.7d}$$

$[l^0_{dp}]$

$$- h_d \sum_{e \in E} \delta_{e,d.p} \mu^1_{e,s_0} + \mu^2_{d,s_0} - \mu^4_{d,p} \leq 0$$

$$\forall d \in D, p \in P'_d \tag{7.7e}$$

$$\mu^1, \mu^3, \mu^4, \mu^5 \geq 0 \tag{7.7f}$$

$$\mu^2 free \tag{7.7g}$$

In Eq (7.7c), the coefficients of $\mu^3_{d,p,s}$ can be either 0 or1 depending on δ_{sdp}. For a given (d, p) pair, it is clear which values are 0 and which are 1: exactly those failure scenarios $s \in S$ not part of the path, meaning $\delta_{sdp} = 0$ for these scenarios s, will yield $1 - \delta_{sdp} = 1$ and for those with failure scenario s an

edge of the path will yield 0. So, we can re-write the first term as $\sum_{s \in S, s \notin p} \mu^3_{dps}$. Similarly the co-efficients of $\mu^5_{d,e}$ in Eq (7.7c) (δ_{edp}) can be either 1 for the edges which are on a path p, or 0 otherwise. So, we can change this term to $\sum_{e \in p} \mu^5_{d,e}$.

7.3.2 The pricing problem

The pricing problem is to identify a variable for which the dual constraint does not hold by the current dual solution. Since all variables χ_{dp}, ℓ^0_{dp}, ℓ^s_{dp} depend on demand d, instead of searching for a single variable, we can decompose the problem, by defining a pricing problem for each d separately. This simplifies the search and might speed up computations as multiple variables can be added at once.

So far the pricing values μ are given by the current master LP solution. Well, except for those $\mu^3_{d,p,s}$ $\mu^4_{d,p}$ for which paths are not yet in the master.

Using a pricing problem, we want to identify new path variables that can improve the current solution. In dual formulation (7.7), we notice that the inequalities (7.7b) can never be violated, because all the edges and scenarios are already in the primal problem. Therefore, in our pricing problem we need to consider inequalities (7.7c), (7.7d) and (7.7f).

For inequality (7.7d) and (7.7f), we can obtain values for μ^1 and μ^2 from the current primal solution, but we do not have values for $\mu^3_{d,p,s}$ and $\mu^4_{d,p}$ for $p \notin P_d$. However, we can deduce a lower bound for $\mu^3_{d,p,s} \forall s$ and $\mu^4_{d,p}$ by using Eqs (7.7d) and (7.7f):-

$$\mu^3_{d,p,s} \geq -h_d \sum_{e \in E | e \in p} \mu^1_{e,s} + \mu^2_{d,s} \forall s \in S_0 \tag{7.8a}$$

$$\mu^4_{d,p} \geq -h_d \sum_{e \in E | e \in p} \mu^1_{e,s_0} + \mu^2_{d,s_0} \tag{7.8b}$$

Substituting $\mu^3_{d,p,s}$ and $\mu^4_{d,p}$ in Eq (7.7c) with their lower bounds:

$$\sum_{s \in S} (1 - \delta_{s,d,p})(-h_d \sum_{e \in E | e \in p} \mu^1_{e,s} + \mu^2_{d,s}) - h_d \sum_{e \in E | e \in P} \mu^1_{e,s_0} + \mu^2_{d,s_0} - \sum_{e \in E} \delta_{e,d,p} \mu^5_{d,e} \leq 0 \tag{7.9}$$

Equation (7.9) can be rewritten as:

$$-h_d \sum_{s \in S \notin p_d} \sum_{e \in E : e \in p_d} \mu^1_{e,s} + \sum_{s \in S \notin p_d} \mu^2_{d,s} - \sum_{e \in E : e \in p_d} \mu^5_{d,e} \leq 0 \tag{7.10a}$$

$$\mu^1, \mu^3, \mu^4, \mu^5 \geq 0 \; and \; \mu^2 \; is \; free \tag{7.10b}$$

Any new path that violates the constraint (7.10a) prices out favourably and we can add χ and ℓ variables corresponding to this path back in the master problem.

We need to construct violating paths for which left hand side of equation (7.10a) is greater than 0. μ^1 and μ^5 are positive variables, so the sum of these variables will always be positive. The minus sign with μ^1 and μ^5 terms means these terms will always satisfy the constraint. So, we need to minimize the summation of μ^1 and μ^5 terms. μ^2 is a free variable. It can assume positive

or negative values. Thus, when the sum of μ^2 over any path is positive, and when this sum is more than the sum of μ^1 and μ^5 terms, the constraint (7.10a) will be violated. Since all three terms of (7.10a) are inter-related, minimizing one may result in an increase in the other. It may be noted here that the minimization of $\sum_{e \in E: e \in p} \sum_{s \in S \notin p} \mu^1_{e,s}$ has already been shown to be NP-hard by Maurras and Vanier [87] and Orlowski [102]. But we cannot use this result straight away, because of this inter-dependence.

The dual weights for each edge may be used to solve the pricing problem through shortest path problem. For example, Orlowski and Pióro [103] have given formulations for pricing problems in the case of network design with path-based survivability mechanisms. Most of the variants where a pricing problem lends itself to the shortest path problem, are solvable in polynomial time. However, the structure of (7.10a) is not such that any shortest path algorithm can be applied straightaway. The difficulty arises from the fact that, for any fixed demand $d = (s_d, t_d)$, we need to find a path that violates (7.10a). The weight of the path from s_d to t_d does not depend only on the individual weights of the links, but on the whole path. $\sum_{s \in S \notin p} \mu^1_{e,s}$ complicates the weight calculation, as it depends on the failure scenarios in which the path will survive.

7.3.3 Solving the pricing problem

In order to solve our pricing problem, we can formulate another ILP, with the objective:

$$\max \sum_{s\in S:s\notin p} \mu^2_{d,s} y_{d,s} - h_d \sum_{s\in S:s\notin p} \sum_{e\in E:e\in p} \mu^1_{e,s} y_{d,e} - \sum_{e\in E:e\in p} \mu^5_{d,e} y_{d,e} \tag{7.11}$$

where $y_{d,e}$ is a binary variable that characterizes a path. $\sum_{s\notin p}$ in the first term suggests that those scenarios which are not on the path should be considered. We can implement this by using $(1 - y_{d,s})$. Similarly, we can implement the condition $s \notin p$ in second term by multiplying it with $(1 - y_{d,s})$, since it only counts if failure scenario s is not on the path p but it has edge e is on it. However, this will result in a quadratic term in the second summation of Eq (7.11).

We can re-write Eq (7.11) as follows:

$$\max \sum_{s\in S} \mu^2_{d,s}(1 - y_{d,s}) - h_d \sum_{s\in S} \sum_{e\in E} \mu^1_{e,s}(1 - y_{d,s}) y_{d,e} - \sum_{e\in E} \mu^5_{d,e} y_{d,e} \tag{7.12}$$

Linearizing the quadratic term by adding a binary coupling variable, $z_{d,e,s}$ with $z_{d,s,e} = 1$ if and only if $y_{d,s} = 0$ and $y_{d,e} = 1$.

$$\max \sum_{s\in S} \mu^2_{d,s}(1 - y_{d,s}) - h_d \sum_{s\in S} \sum_{e\in E} (\mu^1_{e,s} y_{d,e} - \mu^1_{e,s} z_{d,e,s}) - \sum_{e\in E} \mu^5_{d,e} y_{d,e} \tag{7.13}$$

where $z_{d,e,s}$ are new binary variables satisfying for all s, e

$$z_{d,s,e} \leq y_{d,e} \tag{7.14a}$$

$$z_{d,s,e} \leq 1 - y_{d,s} \tag{7.14b}$$

$$z_{d,s,e} \geq y_{d,e} - y_{d,s} \tag{7.14c}$$

We further need to add constraints to construct a meaningful path. Paths can be added by using the node-based approach described in Chapter 5 earlier. The node-based approach assumes directed edges.Therefore, we need to transform our undirected edges into directed ones. For that we adopt a similar approach which has been followed earlier by [103] and will define two new binary variables $x^e_{i,j}$ and $x^e_{j,i}$ corresponding to each edge $e \in E$, to represent a flow from node i to node j and vice versa. For the set of nodes V, assume $\delta(v)$ is the set of undirected edges incident to $v \in V$.
Then, the flow conservation equations can be stated as follows:

$$\sum_{e \in \delta(v) | e = \{i,j\}} (x^e_{i,j} - x^e_{j,i}) = \begin{cases} 1 & \text{if } v \text{ is a source;} \\ -1, & \text{if } v \text{ is a target;} \\ 0, & \forall v \in V \backslash \{s,t\} \end{cases} \tag{7.15}$$

In order to link y variables with x variables, we add Eq (7.16), which states that if an arc is selected in a path, then the corresponding y value must be 1. However, since in any path an edge can be used only in one direction, the summation of x variables for each specific edge is limited to be 1.

$$y_{d,e} = x^e_{i,j} + x^e_{j,i} \qquad \forall e \in E | e = \{i,j\} \tag{7.16}$$

Example: We will show the benefit of our pricing problem with the help of a small example: Consider a graph consisting of 7 nodes and 10 edges, as shown in Figure 7.4. The capacity of all the edges is 1 . There are two demands d_0 from n_1 to n_3 and d_1 from n_5 to n_7 with $h_{d_0} = h_{d_1} = 1$. Assume that we start with the master problem with following paths for each demand:

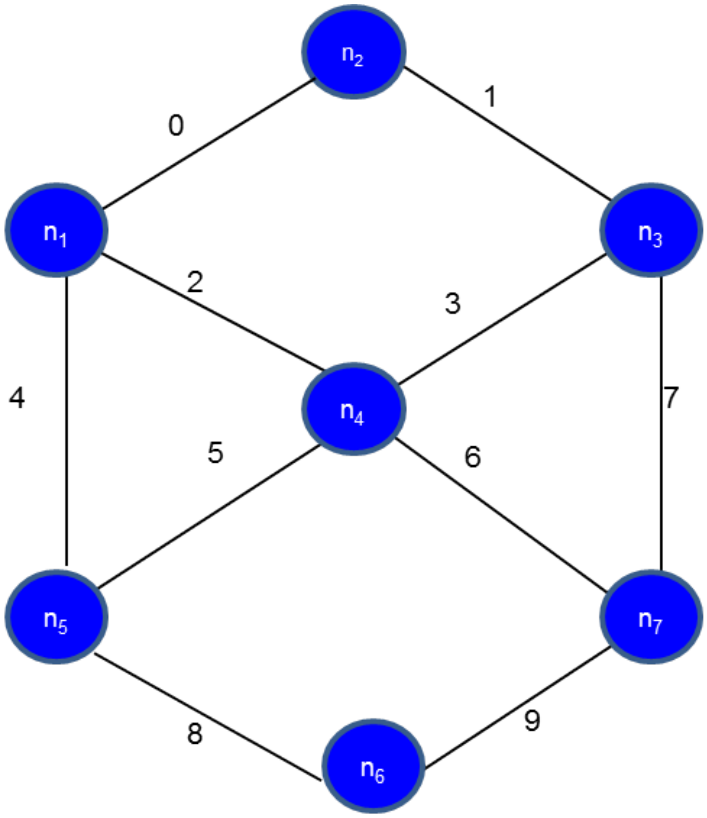


Figure 7.4: A network with 7 nodes and 10 edges

For d_0 we have two paths using the edges: (0,1) and (4,8,9,7).

For d_1 we have two paths using the edges: (8,9) and (4,0,1,7).

The solution value of the primal problem will be $\rho_{max} = 2$. We can see in the Figure 7.4 that in failure scenarios {0,1,8,9}, some of edges will be used by both demands resulting in the highest utilization of 2 units (hence resulting in congestion).

From the solution of the primal problem, we obtain the dual values μ. Most of the values are zero, except for following: $\mu^1_{0,8} = 0.2$, $\mu^2_{0,8} = \mu^2_{1,8} = 1$, $\mu^3_{0,1,8} = \mu^3_{1,0,8} = 1$

In order to improve the ρ_{max} with 2 paths per demand, let us consider

a new path (2,3) for d_0. From the pricing problem (7.13) to (7.16), we can calculate the values for y and z variables. For path (2,3) $y_{0,2} = y_{0,3} = 1$, while the y variables for rest of the edges are 0. Similarly, from Eq. (7.14) we obtain the z values. $z_{d,e,s}$ can be 1, if $y_{d,e} = 1$ and $y_{d,s} = 0$. Therefore z variables can have value 1, only for two $e = 2$ and $e = 3$ cases. However, z values cannot be 1 for those s values for which $y_{d,s} = 1$. Hence, for $e = 2$ and $e = 3$, the z values are as follows:

$z_{0,0,2} = z_{0,1,2} = z_{0,4,2} = z_{0,5,2} = z_{0,6,2} = z_{0,7,2} = z_{0,8,2} = z_{0,9,2} = 1$, and

$z_{0,0,3} = z_{0,1,3} = z_{0,4,3} = z_{0,5,3} = z_{0,6,3} = z_{0,7,3} = z_{0,8,3} = z_{0,9,3} = 1$, while

$z_{0,2,2} = z_{0,3,2} = z_{0,3,2} = z_{0,3,3} = 0$

All other z values are 0.

Using these values, we calculate the reduced cost for this new path (2,3) from Eq. (7.13), and we obtain a value 1. Hence we can conclude that the path (2,3) violates the current constraints as the reduced cost is greater than zero. Therefore, adding this path variable to the primal problem can potentially improve the solution.

Hence, we add path (2,3) for demand d_0 in the master problem. The maximum link utilization reduces to 1. The routing for demand d_0 is through paths (0,1) and (2,3), while demand d_2 uses paths (8,9) and (4,0,1,7). Now the failure scenarios 0,1 will affect both demands simultaneously, but still there are non-overlapping paths that can be used by both demands. Resultantly, the ρ_{max} in this situation is reduced from 2 to 1.

7.4 Summary

In this chapter we have discussed two approaches to solve our routing problem more efficiently than our previously proposed ILP methods. Firstly, we have proposed a hybrid approach, where an ILP is combined with a heuristic algorithm. We have given an empirical comparison between exact and heuristic methods.

For large instances, in a given time limit, the heuristic approach gives good quality solutions. For most instances where a solution was found using E2, the computational times taken by using the heuristic approach were less than those of the exact method. We have also looked at the effect of variation in demand sizes on the performance of our proposed methods.

Our second approach uses column generation. We have presented a pricing problem without the scenario based constraints. The pricing problem has a difficult structure. Orlowski [102] and Maurras and Vanier [87] have proven the NP-hardness of problems with similar structures, but because of the inter-dependence of different terms in the pricing problem, we cannot apply their results straightaway. We have proposed an original ILP formulation to solve this pricing problem and have shown its benefit with the help of an example.The implementation of this column generation based method is an area that constitutes work in progress.

8

Conclusions

In this thesis we have looked at the problem of network routing with the view to provide resilience against single link failures.

For voice grade communications, where fast recovery from failures is required, the circuit-switched Synchronous Optical Networking (SONET) protocol has been traditionally used. Because of its ring architecture, it assures 50 ms recovery time. Over time, Gigabit-Ethernet has appeared as a cheap competitor with more flexible architectural options. But it does not provide the same level of failure protection. Ethernet-based networks therefore rely on

layer-3 protocols to provide such a protection.

We noted in our overview of the characteristics of OSPF and MPLS layer-3 protocols in Chapter 2 that the failure detection and signalling, which are the two major contributing factors of delay in failure restoration, are particularly slow in OSPF. The link-state routing tables take time to react to the failure. Their convergence is slow and resource intensive. ECMP provides a way out, but the availability of equal cost paths is not always possible. MPLS, on the other hand, can be configured to provide manual protection switching in case of link and/ or node failures. Automatic protection can also be configured. However, one issue with MPLS is that it too relies on the routing table information to establish the local repair paths. Therefore, with local repairs, it has to wait for the routing vectors to converge, which alone can take several milliseconds. The Fast Reroute mechanism in MPLS preallocates the detours in case of failures and therefore can react faster. It promises 50 ms recovery time. But the detours may become inconsistent by the time routing tables converge. The pre-allocation of detours is also cumbersome and resource intensive, as each link in the protection domain has to have a detour.

In order to provide faster recovery with reduced capacity overheads, we used a multipath based traffic distribution on end-to-end paths, where all the paths in a multipath are edge-disjoint. In our model, the traffic distribution is based on a demand matrix. For each demand, the end routers maintain the traffic distribution patterns in all failure scenarios. With the paths already pre-determined, this method does not require waiting for the routing vectors to converge. A control signal sent to the ingress router can initiate the redistribution of the load on the surviving paths. This multipath strategy, firstly, reduces the impact of a failure on any demand by a k-th factor, where k is the

number of paths in the multipath. Secondly, the time required to recalculate a backup path is saved and the routers do not need to wait for the routing vectors to converge to start recovery. The signaling to ingress can be done with a control packet.

The choice of paths that are considered while distributing the load also plays a major role in the efficiency of this method. We want to select the best possible disjoint paths in the multipath. The options of paths become exponential. Therefore, in order to identify the optimal paths we explored various options of using combinatorial optimization methods. We proposed a path selection paradigm for the multipath based routing strategy. We consider this to be a significant advancement upon the existing body of knowledge, with novel ILP formulations, novel algorithms and complexity calculations of the considered routing problem.

The contributions of this research can be spelled out as follows.

Complexity of the resilient network routing problem

First of all, we have looked at the theoretical complexity of the problem. We started with the multipath routing problem, where each demand has to have a set of disjoint multipaths. By reduction from the Satisfiability problem, we have proved that the problem is NP-complete even for a single commodity case. For the modified multipath routing problem, we introduced additional scenario-specific constraints to restrict the load re-distribution only in the scenarios where a demand is affected. This introduction of new constraints makes the problem computationally harder to solve, and it also changes the structure of the problem. Therefore, we cannot apply the same logic to infer that the modified problem is also NP-complete. Determination of the complexity of

this modified multipath routing problem is still an open question.

Novel ILP formulations for the modified resilient network routing problem

In contrast with the current practice and use of multipath strategies where a pre-determined disjoint multipath is used for load distribution, we have investigated the benefit of using the path selection as part of the optimization process. We have looked at the possibility of merging the two steps, i.e., computation of disjoint paths and optimization of the network utilization. We examined the cases of using path sets with disjoint-only paths and non-disjoint paths. Our results corroborate the hypothesis that we set out initially for the research – i.e., that if the path selection is made a part of the optimization process, the load distribution will be such that not only the maximum network utilization is minimized, but the capacity utilization of individual links will also be optimized, thus balancing out the load on the whole network. We also observed that usually 3 paths are sufficient to capture the benefit of multipaths. This observation corroborates the results of Menth et al. [88], where they have also reached the same conclusion.

We have also presented two novel ILP formulations (E1 and E2) for the modified resilient routing problem. Using the test cases where we used three different sizes of networks, which can be viewed as representative cases for small, medium and large-sized networks, we have shown the performance of our methods empirically. We have shown that the E2 formulation is more compact than the E1 formulation and can be used to solve small to medium-sized routing problems within a reasonable time limit (2 hours time limit was used in our experimentation).

Heuristic algorithm for resilient multipath routing

An alternative approach worth exploring, in view of the complexity of the problem, was that of the development of an appropriate heuristic algorithm. In this thesis, we have presented a hybrid algorithm that combines a local search meta-heuristic with an exact neighborhood search and tackles the larger problems within reasonable computational time. The empirical examination on several test problems demonstrated that, with the new hybrid approach, the problem can be solved much faster. The heuristic works well and, in several cases, yields better solutions than ILP in a given time limit, or provides solutions for problems where ILP could not even find one valid solution in the given time limit. In cases where ILP is able to find the optimal solution, the heuristic algorithm obtains the same solution usually in less time.

A survey of the heuristic and hybrid methods to provide resilience in networks

We have also provided a comprehensive survey of the heuristic and hybrid methods used for solving the network routing problems, especially where network protection, reliability and survivability are among the routing considerations. Our survey has led to the finding that, apart from simple search heuristics, the mixing of exact and meta-heuristic methods in the area of network routing is a recent trend, where most of the work has been done in the past 10 years. There is a vast potential of research in the hybridization of exact and meta-heuristic methods in the area of survivable multi-path network routing. Our own research is an incremental step in that direction.

Column generation for resilient multipath network routing problem

Finally, we have also provided a column generation based formulation to solve the resilient multipath network routing problem. The problem is not straightforward for an easy pricing of the missing paths.

The pricing requires the dual values corresponding to the paths that are not in the primal problem. To overcome this, we have found the lower bounds for these dual values, which are not dependent on the paths and have reformulated the reduced costs for the missing variables. However, in order to calculate the reduced costs, we need to take into consideration all edges on a path in no-failure and failure scenarios. Thus, the calculation of each edge's weight is dependent on the fact whether or not it is part of a path. Hence, independent edge weights cannot be calculated, which means that we cannot apply a shortest path algorithm to generate the paths using the dual values as the edge weights.

We have, therefore, presented a novel ILP formulation for the pricing problem, where quadratic variables are linearized by linear coupling variables. The correctness of the proposed pricing strategy is verified on an example network. However, this approach needs to be taken further and is indeed a worthwhile direction for future research.

8.1 Future Research Directions

Optimal path selection in network routing has been the central concern of this research project. We adopted different approaches and methodologies for multipath selection for load distribution to forestall link failures. Exact and heuristic methods or combinations thereof, have been identified. Yet, this

quest for better solutions to the problem is far from complete. We suggest below some directions for future research. These pointers come out of the work presented in the preceding chapters and appear to pose an interesting agenda that could be pursued.

Experimentation with column generation

The extension of the application of column generation on the routing problem is also an interesting area of future research. There are different possible off-shoots. The most obvious one is the experimentation with the proposed framework and then to make an attempt to incorporate the constraints we excluded from the proposed column generation framework.

Another interesting area is to apply the simultaneous Column-Dependent Row (CDR) generation framework proposed by Muter et al. [98]. In their recent work the authors have presented the case where the problem at hand has more than one type of variables, which are inter-related. Hence pricing and adding variables is not straightforward. They have argued that, using the traditional column generation method, the missing dual values corresponding to the missing constraints may result in an incorrect pricing. Hence, a simultaneous column dependent row generation framework is more appropriate for such type of problems.

They have delineated three assumptions to determine if the problem is amenable to their proposed framework. The features of our problem match those required for CDR algorithms. We describe their assumptions using our problem formulation 7.2 presented in Chapter 7 as an example.

We generate paths through pricing χ variables and we have constraints (7.2d -7.2e) and (7.2g -7.2i) that link χ and ℓ variables.

The generation of the ℓ variables depends on the generation of χ variables. Furthermore, each ℓ variable is associated with only one set of linking constraints. This is in accordance with their first assumption.

The second assumption requires the definition of a minimal variable set. It is the minimal set of one type of variable that results in the generation of associated variables and linking constraints. In general, a set of linking constraints may be associated with several minimal sets. In our case, however, there is a one to one correspondence between minimal variable sets and linking constraints. A new $\{\chi_k\}$ variable will generate a new set of constraints of type (7.2d -7.2e) and (7.2g -7.2h) along with associated $|S|$ ℓ_k^s variables for each scenario. The generation of type constraint (7.2i) may be linked with more than one $\{\chi_k\}$ variables.

Our problem satisfies the second assumption, which requires that a linking constraint is redundant unless any of the minimal variable set is added in the restricted master problem. In our case, it is obviously the case as there is a one to one correspondence between minimal variable sets and linking constraints.

The third assumption requires that if χ_k is zero, ℓ_k^s cannot take positive values.

Since the problem satisfies the assumptions, it is amenable to the simultaneous column and row generation algorithm. It requires three types of pricing algorithms – two with respect to the χ and ℓ variables with the known dual values, and the third one is the row generating pricing. It is a two-stage optimization problem taking into account the unknown duals corresponding to the missing constraints.

Furthermore, our problem can be categorized as that of a mixed-CDR,

meaning that some minimal variable sets are of cardinality one as in constraints (7.2d -7.2e,7.2g -7.2h), while others are composed of two or more χ variables as in constraint (7.2i). The generation of the (7.2i) may complicate the column generation but it definitely would pose an original contribution.

Simulation of the proposed methodology

We have proposed a path selection method for multipath approach for routing traffic over MPLS networks. The effect of various path selection strategies with respect to other MPLS based protection switching mechanisms (such as FRR) needs to be ascertained. One would need to develop a simulation where different experiments can be conducted to see how much improvement in capacity utilization and recovery from failure is achievable by splitting the traffic on disjoint paths. By using simulation we will be able to see the performance of our approach with respect to delay and congestion.

In order to develop a simulation, one would have to develop a new routing protocol capable of the multipath routing feature on top of an MPLS network. The simulation could be developed using one of the discrete-event network simulators. For example, OPNET [3] is a commercially available simulator that has an MPLS module to simulate FRR and other traffic engineering features of the protocol. Network Simulator 3 (NS-3) [2] is a freely available discrete-event network simulator and an MPLS module can be added on top of the core NS-3 software to develop path based simulations.

Demand fluctuation

Possible issues with this multipath approach are that it relies heavily on the quality of the demand matrix. If the demand matrix is not a good represen-

tative of the actual demand, the load distribution and path selection will be suboptimal and we cannot achieve the desired effect of load balancing. Also, in case of special events, which might cause a large variation from the regular demand pattern, the load distribution may not be a good one. In order to tackle these issues, one option could be to have different load distributions for different types of demand patterns. Another possibility could be to re-calculate/adapt load distribution frequently.

The solution obtained through ILP/heuristic can also be tested to see how it will perform in case of demand fluctuation. One way would be to obtain the solution for one demand matrix and then use the obtained routing to experiment with various demand matrices and to determine the efficiency of routing in each scenario. If one were able to identify demands causing congestion, one could try to re-adjust the load of these demands onto other paths. As the overflow/congestion on a link is a cumulative effect of demands using that particular link, it might not be very straightforward to identify the demands causing overflow.

In a core network, there are demands between node-pairs, which might increase and decrease over time. But no additional demands are generated. So, there should be routing available to fulfill all demand values. In practice, while planning the routing for the network, average (over 5 min periods) demand value at 95% of the time is generally used. Another practice is to multiply the mean demand value by 3 and then use that as the demand value. Two possible ways to adapt to the demand variation may be to make certain rules that specify which route to use in a particular traffic situation. Alternatively, the routing can be aimed for the least congested path among a set of paths (say, five paths). However, both of these strategies would require a constant

monitoring of the network status. When there is a large fluctuation in demand values, it is unrealistic to use one single value for routing/planning. But the question is when to change the load distribution. If we change it in real time, it would resemble the way OSPF acts in real time. However, one should aim to improve upon restoration performance of OSPF. One possibility is to use the past data to predict the next hour traffic.

Another interesting aspect would be to look at the correlation between different demand variations and to find a routing that satisfies the most likely maximum demands, as planning for the worst case is not what practitioners are generally aiming at. Their interest would generally lie in a certain percentage of time for which the traffic can move without congestion.

Stochastic link failure

The availability analysis of the network is conducted to evaluate the probability that a connection is not there. Availability analysis uses probability of availability of each network element. The probability that a path is available is a product of availabilities of all links and nodes on that path. Link availability can be calculated as a ratio between mean up-time for a link to the time that the link should be up. There does not seem to be any work where availability of paths is made part of optimization problem (i.e one of the constraints) and it would be of interest to explore the literature on availability analysis from this point of view. One possibility would be to have minimum availability of all paths as high as possible.

To wrap up: we have used the self-protecting multipath paradigm to address the central question (survivability) of network management. We have been eclectic in our choice of methodologies, crossing the boundaries of exact

and heuristic (including metaheuristic) techniques with insouciance – indeed, synthesizing them to produce better results. In the process we have made the problem of network routing more tractable. The models that we developed have demonstrated a viability. We have shown the effectiveness of the disjoint multipath approach, and proven further that, if the selection of disjoint paths is made an endogenous part of optimization problem, then the link capacities can be utilized better. This is a marked improvement upon previous work in which shortest k-paths were pre-selected. The results bear out our hypothesis that, in our refined model, the load distribution on the overall network is more balanced. Our algorithm results also in minimizing the maximum link utilization (an objective function) and thereby provides a margin for growth in network traffic without causing congestion.

Our survey of the heuristic applications for network routing can be viewed not only as a contribution to the extant literature in its own right, but also as furnishing the context within which our hybrid algorithm constitutes an incremental advance in cross-fertilized operational research.

References

[1] Multipath TCP. http:/datatracker.ictf.org/wg/mptcp/charter/. accessed on 28/01/2013.

[2] The Network Simulator 3. http://www.nsnam.org.

[3] OPNET MPLS Module. http://www.opnet.com/solutions/network_rd/simulation_model_library/mpls.html.

[4] ITU-T Recommendation Y.1720: Protection switching for MPLS networks. Technical report, Dec 2006.

[5] ITU-T Recommendation G.808.1: Generic protection switching Linear trail and subnetwork protection. Technical report, February 2010.

[6] E.H.L. Aarts and J.K. Lenstra. *Local Search in Combinatorial Optimization*. Wiley, NY, 1997.

[7] S. K. Ahuja and S. Ramasubramanian. All-to-all disjoint multipath routing using cycle embedding. *Computer Networks*, 52:1506–1517, 2008.

[8] S. Alouneh, A. Agarwal, and A. En-Nouaary. A novel path protection

scheme for MPLS networks using multi-path routing. *Computer Networks*, 53(9):1530 – 1545, 2009.

[9] J. Armitage, O. Crochat, and Y.J.L Boudec. Design of a survivable wdm photonic network. In *IEEE INFOCOM 97*, 1997.

[10] G. Baier, E. Köhler, and M. Skutella. The k-splittable flow problem. *Algorithmica*, 42:231–248, 2005.

[11] S. Balon and G. Leduc. Combined intra- and inter-domain traffic engineering using hot-potato aware link weights optimization. *CoRR*, abs/0803.2824, 2008.

[12] R.S. Barpanda, A.K. Turuk, B. Sahoo, and B. Majhi. Genetic algorithm techniques to solve routing and wavelength assignment problem in wavelength division multiplexing all-optical networks. pages 1 – 8, Jan. 2011.

[13] R. Bellman. On a Routing Problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

[14] A. Bley, M. Grötschel, and R. Wessäly. Design of broadband virtual private networks: Model and heuristics for the B-WiN. In N. Dean, D. F. Hsu, and R. Ravi, editors, *Robust Communication Networks: Interconnection and Survivability*, volume 53 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 1–16. AMS, 1998.

[15] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: From natural to artificial systems.* Santa Fe Institute Studies in the Sci-

ences of Complexity. Oxford University Press, Oxford, 1999. ISBN 9780195131598.

[16] Q. Botton, B. Fortz, L. Gouveia, and M. Poss. Benders decomposition for the hop-constrained survivable network design problem. *INFORMS Journal on Computing*, 25(1):13–26, 2013.

[17] J. A. Boyan and M. L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in Neural Information Processing Systems*, pages 671–678. Morgan Kaufmann, 1994.

[18] L. S. Buriol, M. G. C. Resende, C. C. Ribeiro, and M. Thorup. A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. *Journal of Combinatorial Optimization*, 6:299–333, 2003.

[19] T. Campanale, G. Carello, L. De Giovanni, F. Della Croce, and R. Tadei. Optimising telecommunication network topology with reliability and routing constraints. In *Technical Report 3/03, Operations Research Series*, DAUIN - Politecnico di Torino, 2003.

[20] S.A. Canuto, M.G.C. Resende, and C.C. Ribeiro. Local search with perturbations for the prize-collecting steiner tree problem in graphs. *Networks*, 38:50–58, 2001.

[21] S. K. Cerav-Erbas. *Trac Engineering in MPLS Networks with Multiple Objectives: Modeling and Optimization.* PhD in mathematik, informatik und naturwissenschaften, Rheinisch-Westfl̈ischen Technischen Hochschule Aachen, Aachen, Germany, 2004.

[22] A. Chabrier. Heuristic branch-and-price-and-cut to solve a network design problem. In *Proceedings of CPAIOR03*, 2003.

[23] S. Chamberland. Overall design of reliable ip networks with performance guarantees. *Le Cahiers du GERAD*, 2000.

[24] J. Chen, P. Druschel, and D. Subramanian. An efficient multipath forwarding method. In *Proc. IEEE INFOCOM 98*, pages 1418–1425, 1998.

[25] D. Cherubini, A. Fanni, A. Mereu, A. Frangioni, C. Murgia, M.G. Scutell, and P. Zuddas. Linear programming models for traffic engineering in 100% survivable networks under combined IS-IS/OSPF and MPLS-TE. *Computers and Operations Research*, 38(12):1805 – 1815, 2011.

[26] V. Chvátal. *Linear programming*. W. H. Freeman and Company, 1983.

[27] I. Cidon, R. Rom, and Y. Shavitt. Analysis of multi-path routing. *IEEE/ACM Transactions on Networking*, 7:885 – 896, 1999.

[28] D. D. Clark. The design philosophy of the darpa internet protocol. In *SIGCOMM '88*, volume 18, pages 106–114, 1988.

[29] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151 – 158, 1971.

[30] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2001.

[31] L. A. Cox-Jr and J. R. Sanchez. Designing least-cost survivable wireless backhaul networks. *Journal of Heuristics*, 6:525–540, 2000. ISSN 1381-1231.

[32] O. Crochat and J.Y.L. Boudec. Design protection for wdm optical networks. *IEEE Journal of Selected Areas in Communication*, 16:1158–1165, 1998.

[33] G. Dahl and M. Stoer. A cutting plane algorithm for multicommodity survivable network design problems. *INFORMS Journal on Computing*, 10(1):1–11, 1998.

[34] L. De Giovanni and R. Tadei. Tailoring neighborhood search for the internet protocol network design problem with reliability and routing constraints. *Networks*, 49(1):65–79, 2007.

[35] A. de Sousa, D. Santos, P. Matos, and J. Madeira. Load balancing optimization of capacitated networks with path protection. *Electronic Notes in Discrete Mathematics*, 36:1249–1256, 2010.

[36] G. A. Di Caro. Ant colony optimization and its application to adaptive routing in telecommunication networks. Phd thesis, 2004.

[37] E. W. Dijkstra. *Numerische Mathematik 1*, pages 269 – 271, 1959.

[38] M. Dorigo and T. Stützle. *Ant Colony Optimization*. The MIT Press, 2004. ISBN 978-0-262-04219-2.

[39] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 26(1):29–41, 1996.

[40] F. Ducatelle, L. M. Gambardella, M. Kurant, H. X. Nguyen, and P. Thiran. Algorithms for failure protection in large IP–over–fiber and wireless ad hoc networks. In *Research Results of the DICS Program*, pages 231–259, 2006.

[41] F. Ducatelle, G. Di Caro, and L. Gambardella. Principles and applications of swarm intelligence for adaptive routing in telecommunications networks. *Swarm Intelligence*, 4:173–198, 2010.

[42] I. Dumitrescu and T. Stützle. Combinations of local search and exact algorithms. *Applications of Evolutionary Computation*, 2611:211– 223, 2003.

[43] E. M. El-Alfy, S. Z. Selim, and S. N. Mujahid. Solving the minimum-cost constrained multipath routing with load balancing in MPLS networks using an evolutionary method. In *IEEE Congress on Evolutionary Computation*, pages 4433–4438, 2007.

[44] ENISA. Measurement frameworks and matrices for resilient networks and services - technical report (discussion draft). Feb 2011. URL www.ensina.europa.eu.

[45] F. C. Ergin, E. Kaldirim, A. Yayimli, and S. Uyar. Performance analysis of nature inspired heuristics for survivable virtual topology mapping. In *Proceedings of the 28th IEEE conference on Global telecommunications*, GLOBECOM'09, pages 997–1002, Piscataway, NJ, USA, 2009. IEEE Press.

[46] M. Ericsson, M. Resende, and P. Pardalos. A genetic algorithm for

the weight setting problem in OSPF routing. *Journal of Combinatorial Optimization*, 6:299–333, 2002.

[47] K Farkas. IP traffic engineering using OMP technique. In *12th IASTED PDCS 2000*, Las Vegas, USA, 2000.

[48] S. Fernandes and H.R. Loureno. Hybrids combining local search heuristics with exact algorithms. In F. Rodriguez, B. M̈lian, J.A. Moreno, and J.M. Moreno, editors, *V Congreso Español sobre Metaheursticas, Algoritmos Evolutivos y Bioinspirados, MAEB'2007*, pages 269–274, 2007. ISBN 978-84-690-3470-5.

[49] R. W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6), 1962.

[50] L. R. Ford Jr. *Network Flow Theory. Paper P-923*, 1956.

[51] B. Fortz. Internet traffic engineering by optimizing OSPF weights. In *Proceedings of IEEE INFOCOM*, pages 519–528, 2000.

[52] B. Fortz and M. Poss. An improved benders decomposition applied to a multi-layer network design problem. *Oper. Res. Lett.*, 37(5):359–364, 2009.

[53] B. Fortz and M. Thorup. Increasing internet capacity using local search. *Computational Optimization and Applications*, 29:13–48, 2004.

[54] A. Fumagalli and L. Valcarenghi. IP restoration vs. WDM protection: Is there an optimal choice? *IEEE Network*, 2000.

[55] V. Gabrel, A. Knippel, and M. Minoux. A comparison of heuristics for the discrete cost multicommodity network optimization problem. *Journal of Heuristics*, 9(5):429–445, November 2003. ISSN 1381-1231.

[56] B. Gendron, J.-Y. Potvin, and P. Soriano. Diversification strategies in local search for a nonbifurcated network loading problem. *European Journal of Operational Research*, 142(2):231 – 241, 2002. ISSN 0377-2217.

[57] L. D. Giovanni, F. D. Croce, and R. Tadei. On the impact of the solution representation for the internet protocol network design problem with max-hop constraints. *Networks*, 44(2):73–83, 2004.

[58] A. Girard, B. Sanso, and L. Dadjo. A tabu search algorithm for access network design. *Annals of Operations Research*, 106(1-4):229–262, 2001.

[59] F. Giroire, A. Nucci, N. Taft, and C. Diot. Increasing the robustness of IP backbones in the absence of optical level protection. In *INFOCOM 2003*, 2003.

[60] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & OR*, 13(5):533–549, 1986.

[61] F. Glover and M. Laguna. *TABU search*. Kluwer, 1999. ISBN 978-0-7923-9965-0.

[62] D. E. Goldberg and J. H. Holland. Genetic algorithms and machine learning. *Machine Learning*, 3:95–99, 1988.

[63] H. Gredler and W. Goraiski. *The complete IS-IS routing protocol*. Springer, 2005. ISBN 1-85233-822-9.

[64] W. Grover. *Mesh Based Survivable Networks: Design, Operation and Evolution*. Prentice Hall, 2003.

[65] C. G. Gruber. *Design and Optimization of Resilient Multipath Networks*. PhD thesis, Technical University Munich, 2006.

[66] R. Hao, J.and Dorne and P. Galinier. Tabu search for frequency assignment in mobile radio networks. *Journal of Heuristics*, 4(1):47–62, June 1998. ISSN 1381-1231.

[67] J. Harmatos. A heuristic algorithm for solving the static weight optimisation problem in OSPF. In *IEEE Globecom*, 2001.

[68] L. He and N. Mort. Hybrid genetic algorithms for telecommunications network back-up rerouting. *BT Technology Journal*, 18:42–50, 2000.

[69] M. Heusse, D. Snyers, S. Guërin, and P. Kuntz. Adaptive agent-driven routing and load balancing in communication networks. *Advances in Complex Systems*, 1:237–254, 1998.

[70] D. Hock, M. Hartmann, M. Menth, and C. Schwartz. Optimizing unique shortest paths for resilient routing and fast reroute in IP-based networks. In *12th IEEE/IFIP Network Operations and Management Symposium (NOMS 2010)*, 2010.

[71] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[72] C. Hopps. Analysis of an equal-cost multi-path algorithm. *IETF, RFC 2992*, Nov 2000.

[73] G. Iannaccone, C. Chuah, S. Bhattacharyya, and C. Diot. Feasibility of IP restoration in a tier 1 backbone. *IEEE Network*, (04):13–19, 2004.

[74] IBM. IBM ILOG CPLEXversion 12.1, 2009. URL http://www.ilog.com/products/cplex.

[75] IBM. IBM ILOG CPLEXversion 12.3, 2009. URL http://www.ilog.com/products/cplex.

[76] A. Jabbar. *A Framework to Quantify Network Resilience and Survivability*. PhD thesis, University of Kansas School of Engineering, 2010.

[77] D. B. Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM*, 24(1):1–13, 1977.

[78] S. H Kim, K.-N. Chang, and S. Kim. A channel allocation for cellular mobile radio systems using simulated annealing. *Telecommunication Systems*, 14:95–106, 2000. ISSN 1018-4864.

[79] S. Kirkpatrick, D. Gelatt Jr., and M. P. Vecchi. Optimization by simmulated annealing. *Science*, 220(4598):671–680, 1983.

[80] J. M. Kleinberg. PhD thesis, Massachusetts Institute of Technology, 1996.

[81] T. Koch. ZIMPL version 3.1, 2010. URL http://zimpl.zib.de/.

[82] A. M. C. A. Koster, A. Zymolka, M. Jäger, and R. Hüelsermann. Demand-wise shared protection for meshed optical networks. *Journal of Network and Systems Management*, 13(1):35–55, 2005.

[83] M. Latva-aho. Networking Challenges for Future Services. http://www.comsoc.org/webcasts/view/networking-challenges-future-services. IEEE COMSOC webcast accessed on 28/04/2013.

[84] S. Liang, A. N. Zincir-Heywood, and M. I. Heywood. Intelligent packets for dynamic network routing using distributed genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '02, pages 88–96, 2002. ISBN 1-55860-878-8.

[85] Y. Lin and D. Meng. A genetic optimization algorithm to solve the problem of the load-balancing of network load. *International Journal of Computer Science and Network Security*, 6:63–68, 2006.

[86] R. Martin, M. Menth, and U. Spörlein. Integer spm: Intelligent path selection for resilient networks. In *6th IFIP-TC6 Networking Conference (Networking)*, Atlanta, GA, USA, 2007.

[87] J-F. Maurras and S. Vanier. Network synthesis under survivability constraints. *4OR*, 2(1):53–67, 2004.

[88] M. Menth, A. Reifert, and J. Milbrandt. Self-Protecting Multipaths - A Simple and Resource-Efficient Protection Switching Mechanism for MPLS Networks. In *Proceedings of the 3rd IFIP-TC6 Networking Conference, Lecture Notes in Computer Science (LNCS) 3042*, pages 526–537, May 2004.

[89] M. Menth, R. Martin, A. M. C. A. Koster, and S. Orlowski. Overview of resilience mechanisms based on multipath structures. In *Proceedings of*

International Workshop on Design of Reliable Communication Networks (DRCN), 2007.

[90] M. Menth, R. Martin, and U. Spörlein. Optimization of the self-protecting multipath for deployment in legacy networks. In *ICC*, pages 421–427, 2007.

[91] N. Mladenovič and P. Hansen. Variable neighborhood search. *Computers; Operations Research*, 24(11):1097 – 1100, 1997.

[92] E. Modiano and A. Narula-Tam. Survivable lightpath routing: a new approach to the design of WDM-based networks. *IEEE Journal on Selected Areas in Communications*, 20(4):800–809, September 2006. ISSN 0733-8716.

[93] E. Mulyana and U. Killat. An alternative genetic algorithm to optimize ospf weights. In *15th ITC Specialist Seminar*, 2002.

[94] E. Mulyana and U. Killat. A hybrid genetic algorithm approach for OSPf weight setting problem. In *2ndPolish-German Teletraffic Symposium (PGTS)*, Sep. 2002.

[95] M. Munetomo, Y. Takai, and Y. Sato. An adaptive network routing algorithm employing path genetic operators. In *ICGA*, pages 643–649, 1997.

[96] M. Munetomo, Y. Takai, and Y. Sato. A migration scheme for the genetic adaptive routing algorithm. In *Proceedings of IEEE Conference on Systems and Cybernetics*, pages 2774–2779, 1998.

[97] S. Murthy and J. J. Garcia-Luna-Aceves. Congestion-oriented shortest multipath routing. In *Proc. of IEEE INFOCOM 96*, pages 1028 – 1036, 1996.

[98] S. I. Muter, I. Birbil, and K. Bulbul. Simultaneous column and row generation for large scale linear programs with column dependent rows. *Algorithms & Optimization*, pages 1–31, 2012.

[99] A. Nucci, B. Sansuo, T. Crainic, E. Leonardi, and M.A. Marsan. Design of fault-tolerant logical topologies in wavelength-routed optical IP networks. In *IEEE Globecom 2001*.

[100] A. Nucci, B. Schroeder, S. Bhattacharyya, N. Taft, and C. Diot. IGP link weight assignment for transient link failures. In *18th International Teletraffic Congress (ITC)*, 2003.

[101] R. Ogier, V. Rutemburg, and N. Shacham. Distributed algorithms for computing shortest pairs of disjoint paths. *IEEE Trans. Inform. Theory*, 39:443–455, 1993.

[102] S. Orlowski. Local and global restoration of node and link failures in telecommunication networks. Master's thesis, Technische Universit, Berlin, 2003. URL http://www.zib.de/orlowski/.

[103] S. Orlowski and Micha Pióro. Complexity of column generation in network design with path-based survivability mechanisms. *Networks*, 59(1): 132–147, 2012. ISSN 1097-0037.

[104] S. Orlowski, R. Wessäly, and A. Tomaszewski. SNDlib 1.0 - survivable network design library, 2007. URL http://sndlib.zib.de.

[105] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004. ISBN 0125571895.

[106] M. Prais and C. C. Ribeiro. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12:164–176, 2000.

[107] J. Puchinger and G. R. Raidl. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *IWINAC (2)*, pages 41–53, 2005.

[108] M. Randall, G. Mcmahon, and S. Sugden. A simulated annealing approach to communication network design. *Journal of Combinatorial Optimization - JCO*, 6:55–65, 2002.

[109] C. Reichert and T. Magedanz. A fast heuristic for genetic algorithms in link weight optimization. In *5th International Workshop on Quality of future Internet Services (QofIS)*, pages 144 –153, Sep. 2004.

[110] M.G.C. Resende and C.C. Ribeiro. Greedy randomized adaptive search procedures. In *F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics*, pages 219–249, 2003.

[111] A. Riedl. A hybrid genetic algorithm for routing optimization in IP networks utilizing bandwidth and delay metrics. In *IEEE Workshop on IP Operations and Management (IPOM)*, 2002.

[112] A. Riedl. Optimized routing adaptation in IP networks utilizing OSPF

and MPLS. In *IEEE International Conference on Communications (ICC)*, 2003.

[113] D. Santos, A. de Sousa, F. Alvelos, and M. Pióro. Optimizing network load balancing: an hybridization approach of metaheuristics with column generation. *Telecommunication Systems*, pages 1–10, 2011.

[114] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5:169 – 207, 1996.

[115] A. Sen, B. Hao, and B. Shen. Survivable routing in wdm networks. In *7th International Symposium on Computers and Communications (ISCC)*, 2002.

[116] X. Shao, L. Zhou, X. Cheng, W. Zheng, and Y. Wang. Best effort shared risk link group (srlg) failure protection in wdm networks. In *ICC*, pages 5150–5154, 2008.

[117] D. Sidhu, R. Nair, and S. Abdallah. Finding disjoint paths in networks. In *Proc. ACM SIGCOMM 91*, pages 43–51, 1991.

[118] K. M. Sim and W. H. Sun. Multiple ant colony optimization for load balancing. In *IDEAL*, pages 467–471, 2003.

[119] K. M. Sim and W. H. Sun. Ant colony optimization for routing and load-balancing: survey and new directions. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 33:560–572, Sept. 2003.

[120] M. C. Sinclair. Evolutionary telecommunications: A summary. In *GECCO Workshop on Evolutionary Telecommunications: Past, Present and Future*, Orlando, Florida, 1999.

[121] D. Subramanian and J. Druschel, P.and Chen. Ants and reinforcement learning: a case study in routing in dynamic networks. In *Proceedings of the international joint conference on artificial intelligence(IJCAI)*, pages 832–838. Morgan Kaufmann, 1997.

[122] J. W. Suurballe. Disjoint paths in a network. *Networks*, 14:125–145, 1974.

[123] N. Taft-Plotkin, B. Bellur, and R. Ogier. Quality-of-service routing using maximally disjoint paths. In *Proc. 7th Int. Workshop Quality of Service (IWQoS99)*, pages 119–128, 1999.

[124] G. Theraulaz and E. Bonabeau. A brief history of stigmergy. *Artificial Life*, 5(2):97–116, 1999.

[125] A. Tomaszewski. The final answer to the complexity of a basic problem in network resilient network design. In *INOC 2013*, 2013. To appear in Electronic Notes in Discrete Mathematics, 2013.

[126] A. Tsirigos and Z. J. Haas. Analysis of multipath routing-part i: the effect on the packet delivery ratio. *Trans. Wireless. Comm.*, 3(1):138–146, January 2004. ISSN 1536-1276.

[127] Resilinets. Kensas University. 2012. URL www.ittc.ku.edu.

[128] T. Van Do, D. Papp, R. Chakka, and X.M.T. Truong. Analysis of mpls multipath routing with the failure and repair of lsps. 2008.

[129] S. P. M. van Hoesel, A. M. C. A. Koster, R. L. M. J. van de Leensel, and M. W. P. Savelsbergh. Bidirected and unidirected capacity installation in telecommunication networks. *Discrete Applied Mathematics*, 133(1-3): 103–121, 2003.

[130] C. Villamizar. OSPF optimized multipath (OSPF-OMP). *44th IETF Meeting, draft-ietf-ospf-omp-02*, 1999.

[131] C. Villamizar. OMP simulations. *Available online*, 1999. URL http://www.faster-light.net/omp/.

[132] S. Vutukury and J. J. Garcia-Luna-Aceves. An algorithm for multipath computation using distance vectors with predecessor information. In *Proc. IEEE ICCCN 99*, pages 534–539, 1999.

[133] S. Vutukury and J. J. Garcia-Luna-Aceves. A practical framework for minimum-delay routing in computer networks. *Journal of High Speed Networks*, 8(4):241–263, 1999.

[134] S. Vutukury and J. J. Garcia-Luna-Aceves. Mdva: A distance-vector multipath routing protocol. In *INFOCOM*, pages 557–564, 2001.

[135] N. Wang, K.-H. Ho, and G. Pavlou. Adaptive multi-topology IGP based traffic engineering with near-optimal network performance. In *IFIP-TC6 Networking Conference (Networking)*, May 2008.

[136] H. F. Wedde and M. Farooq. A comprehensive review of nature inspired routing algorithms for fixed telecommunication networks. *J. Syst. Archit.*, 52(8):461–484, August 2006.

[137] T. White. *SynthECA: A Synthetic Ecology of Chemical Agents*. PhD thesis, Carleton University, 2000.

[138] T. White and B. Pagurek. Towards multi-swarm problem solving in networks. *In 3rd International Conference on Multi-Agent Systems (ICMAS 1998)*, pages 333–340, 1998.

[139] L. W. Wolsey. *Integer programming*. Wiley, 1998. ISBN 978-0-471-28366-9.

[140] J. Xu, S. Y. Chiu, and F. Glover. Tabu search for dynamic routing communications network design. *Telecommunications Systems*, 8:55–77, 1997.

[141] M. Yabandeh, S. Zarifzafeh, and N. Yazdani. Improving performance of transport protocols in multipath transferring schemes. *Computer Communications*, 2007.

[142] W. T. Zaumen and J. J. Garcia-Luna-Aceves. Loop-free multipath routing using generalized diffusing computations. In *Proc. IEEE INFOCOM 98*, pages 1408–1417, 1998.

[143] A. Zolfaghari and F.J. Kaudel. Framework for network survivability performance. *IEEE Journal on Selected Areas in Communications (JSAC)*, pages 46–51, Jan 1994.