Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

Path Based *p*-Cycle for Resilient MPLS Network Design

A thesis presented in partial fulfilment of the requirements

for the degree of

Master of Engineering

in

Telecommunications and Networking

Wellington,

New Zealand.

Jing Zhang

2010

To my husband

Zheng Liu

Abstract

Resilient networks are those that are capable of continuously offering telecommunication services in the presence of network failures. New paradigms for reliable network design have been emerging and constantly improving network survivability. Failure-Independent Path-Protecting (FIPP) *p*-cycles are a path based extension of the well-known *p*-cycles and inherit the attractive properties of ring-like recovery speed and mesh-like capacity efficiency. They are suitable for application to the MultiProtocol Label Switching (MPLS) protocol that is widely used in Next Generation Networks (NGNs), in that it provides shared, failure independent, end-to-end protection to whole working paths.

This thesis contributes to advance in the state of the art for FIPP *p*-cycle based resilient networking. We firstly examine the two basic models: known as FIPP-SCP and FIPP-DRS. This is followed by an introduction to a Joint Capacity Allocation (JCA) design based on the FIPP-SCP model, which is more favourable to be used in MPLS networks.

The network design is referred to as the MFIPP-JCA model and involves three specific cases:

i) The BR model allows for bifurcated normal routing and imposes no restriction on the use of FIPP *p*-cycles;

ii) NBR or non-bifurcated routing, focuses on single path routing, while it retains the flexibility of a protection domain;

iii) An SNBR model where the main difference from the NBR model is that only a single FIPP *p*-cycle can be used to protect a working path.

Case studies investigated the performance of the MFIPP-JCA models and, for a comparison with the basic FIPP-SCP model. The main areas of interest include the total

cost of capacity, the number of FIPP *p*-cycles, and the solution time. The studies are also performed regarding changes in performance with regard to the number of eligible cycles. Those candidate cycles are the N-shortest cycles that are selected on either a circumference-based or hop-based criterion. The final contribution of this thesis is an in-depth discussion on the implementation issues for FIPP *p*-cycles in MPLS networks. We propose two operation modes both for bidirectional FIPP *p*-cycles, and make a judgment on the potential of unidirectional FIPP *p*-cycles.

Acknowledgements

I sincerely would like to thank my supervisor Prof. Richard Harris, for giving me the opportunity for being his Masters student, for trusting in my commitment, for his proficient guidance, and above all for his expert advice on many topics, which are not limited to the field of this research. I deeply appreciate his help and encouragement throughout my master studies.

I also would like to thank Senior Lecturer Dr. Edmund Lai for his patient explanations and support over the course of my master studies, and for his valuable suggestions on this thesis.

I would like to thank the staff of the School of Engineering & Advanced Technology of Massey University for providing me with technical support, along with a pleasant atmosphere for studying. Especially thanks to Mr. Poh Ng for helping me with the testing machine and the various types of software.

In addition, I would like to express my thanks to PhD student Le Thu Nguyen for her friendship and helpful advice on the research. Then, I wish to acknowledge Yun Liang and Chris Leather for their help on the writing of this thesis, as well as for their emotional support.

A very special thank you to my husband Zheng Liu, without your love, support and encouragement, I doubt that I would ever have begun my master studies. I cannot end without thanking my parents, who never hesitate to give me encouragement and their love.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AMPL	A Mathematical Programming Language
ATM	Asynchronous Transfer Mode
B&B	Branch and Bound
BR	Bifurcated (Normal) Routing
BIP	Binary Integer Programming
CAC	Call Admission Control
CSPF	Constrained Shortest Path First
DRS	Disjoint Route Set, or specific to the FIPP-DRS method
FEC	Forwarding Equivalency Class
FIPP	Failure-Independent Path-Protection
FIS	Fault Indication Signal
FRR	Fast Reroute
FTN	FEC-to-NHLFE Map
ILM	Incoming Label Map
ILP	Integer Linear Programming
IP	Internet Protocol
IS-IS	Intermediate System to Intermediate System
JCA	Joint Capacity Allocation
LDP	Label Distribution Protocol
LER	Label Edge Router
LIB	Label Information Base
LP	Linear Programming or Linear Program
LSP	Label Switched Path
LSR	Label Switched Router
MCMF	Multi-Commodity Maximum-Flow

MIP	Mixed Integer Programming
MPLS	Multi-Protocol Label Switching
NBR	Non-Bifurcated (Normal) Routing
NEPC	Node-Encircling *p*-Cycles
NHLFE	Next Hop Label Forwarding Entry
O-D	Origin-Destination
OSPF	Open Shortest Path First
PDH	Plesiochronous Digital Hierarchy
PHP	Penultimate Hop Popping
PLR	Point of Local Repair
PML	Path Merge LSR
PSL	Path Switch LSR
QoS	Quality of Service
RHS	Right Hand Side
RSVP-TE	Resource Reservation Protocol – Traffic Engineering
SCA	Spare Capacity Allocation
SCP	specific to the FIPP-SCP method
SDH	Synchronous Digital Hierarchy
SNBR	Non-Bifurcated Normal Routing with Single Backup Structure
SONET	Synchronous Optical Network
VPLS	Virtual Private LAN Service
VPN	Virtual Private Network
WDM	Wavelength Division Multiplexing

Chapter 1 Introduction

1.1 Network Recovery

By entering the phrase *"news on Network failure"* into the Google search engine and observing the results, you would be astonished at the "endless" number of pages of information. You can even count months of events in a long numerical sequence. Network failures occur in all types of networks. Fixed line telephony networks, broadband networks, wireless communication systems; no network can be so perfect that it can provide services without any interruptions. People would never know what to do if their telephones, cell phones and laptops were suddenly out of service. This sort of thing can happen in many situations, including natural disasters such as hurricane Katrina in the USA or the recent earthquakes in Haiti and Christchurch. It turns out to be a nightmare for network operators, who must busy themselves by getting a good grasp of the event. There is the potential for them to suffer great financial loss, as well as exposing themselves to the risk of losing customers.

Network recovery or resilient schemes are designed to offer a rapid response to network failures. Alternative paths are used to reroute the affected traffic as soon as a failure in the network is detected. In order to distinguish those paths from working paths, they are called recovery paths or backup paths. Recovery mechanisms can be classified into two general categories: *Protection* and *Restoration*. In the case of protection, the recovery paths are pre-planned and fully signalled in advance. No additional signalling is needed to establish the protection paths after failure occurs. Therefore, the major advantage of protection is that it provides fast and assured failure recovery. Restoration schemes require additional signalling to establish the restoration paths upon the occurrence of failure, regardless of whether the paths are pre-computed or dynamically determined. It can be more flexible and capacity efficient due to the possibility that they can take the

exact nature of the failure into consideration. But restoration times are usually longer, and traffic recovery cannot be 100% guaranteed, because sufficient spare capacity may not be available at the time of the failure.

Most studies in the field of resilient network design are dedicated to protection rather than restoration. There are several protection variants based on different approaches to resource reservation. Basically, they can be categorized into two levels: dedicated and shared as shown in Figure 1.

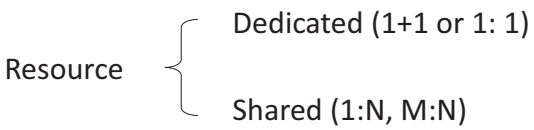


Figure 1 Resource allocation classification

For 1+1 dedicated protection, a recovery path protects exactly one working entity (e.g, a link or a working path). Normal traffic is permanently duplicated and carried on both the working and backup paths. The only thing that needs to be done at the time of failure is protection switching. Thus, this strategy is very fast, but at the expense of bandwidth. In order to enhance capacity utilization, extra traffic can be allowed to travel along the protection path in failure-free conditions. This gives the case of 1:1 dedicated protection. The extra traffic will be pre-empted whenever recovery paths are needed. By contrast, the 1:N protection scheme devotes one recovery path to the protection of up to N working entities which are explicitly identified. In other words, the resources on the recovery path are shared by N working entities. In fact, M:N protection is the general case of shared protection, where M specifies the number of recovery paths in a set to protect a group of working entities. Shared protection is more efficient than dedicated protection in terms of capacity utilization. Note that no reservation (i.e., best effort) can also be regarded as another category of protection, in which case, the failure is recovered only if the resources are available.

1.2 Research Introduction

p-Cycle[1] is a new strategy of M:N shared protection schemes and has evolved continuously since it was first proposed in 1998. The property of ring-like recovery speed and mesh-like capacity efficiency are becoming a surrogate for *p*-cycles. It is an attractive and promising mechanism, applicable to almost any type of network. This thesis aims to advance the state of the art in the use of *p*-cycles, especially in Multi Protocol Label Switching (MPLS) [2] networks. Among the *p*-cycle and its extensions, a Failure-Independent Path-Protecting (FIPP) *p*-cycle is considered to be the strongest contender for MPLS-based recovery, due to its path-oriented nature.

Meanwhile, as a new switching mechanism, MPLS provides a solution to the convergence of all services on to the same multi-service core. Recovery for this layer is essential. Traditional IP routing provides rerouting functions for recovery purposes, but this can only effect after the sub-second scale convergence. Increasingly, it is found that the overall recovery time cannot be tolerated. By contrast, 50ms restoration times can often be achieved in the physical layer. However, the investment in the physical layer for recovery can often be too expensive and may involve other non-technical issues. Node failures in MPLS networks can only be dealt with by the action of peer-level network elements. This is out of reach of a recovery mechanism, which works on the physical layer. It is worthwhile to note that for those small network operators, who rely on others for network infrastructure, physical layer restoration is not technically or economically an option. Lastly, the recovery schemes, either for physical or for the IP layer, cannot offer bandwidth protection to each traffic flow. By contrast, MPLS-based recovery is capable of doing so, because of the underlying label switching mechanism.

1.3 Layout of the Thesis

This thesis sets out to explore the use of *p*-cycle extensions for improving the reliability of MPLS networks and focuses on a recently introduced end-to-end path protecting scheme, known as FIPP *p*-cycles. Our investigation was carried out on the basis of bidirectional FIPP *p*-cycles.

This thesis contributes to the use of path protection in order to achieve a desired recovery mechanism for MPLS networks. It is divided into 7 chapters and organized as follows.

In Chapter 1, a brief introduction to the concept of network recovery is presented, as well as a classification of the many different recovery schemes. The scene of our research is set with respect to MPLS networks and the protection scheme.

Chapter 2 introduces the basics of MPLS, along with a detailed discussion of both the forwarding and control mechanisms. Important metrics for recovery schemes are also introduced. This is followed by a thorough review on the state-of-the-art for MPLS-based recovery schemes. *p*-Cycles and their various extensions are discussed before the conclusion of this chapter.

Chapter 3 introduces the foundations for linear programming, which aids in the understanding of the mathematics used in the rest of this thesis. The FIPP *p*-cycle concept is then discussed in detail, along with its attractive properties. The chapter also gives details about the two basic models for FIPP *p*-cycles, followed by a deeper analysis of them.

Chapter 4 is the core of this thesis. It begins with considerations for extending FIPP *p*-cycles to MPLS networks, followed by a general comparison with other recovery

mechanisms. Then, a design methodology, known as Joint Capacity Allocation (JCA) is introduced. Significantly, three JCA network designs for FIPP *p*-cycles are proposed. A detailed set of experimental results are described in Chapter 5. The solutions obtained from these new models are compared, as well as those for the basic model from which they originate. Results are summarized in Section 5.3.

Chapter 6 focuses on implementation related issues. It contains a review of the application for *p*-cycles to IP environments. Two modes of operation are proposed for the deployment of bidirectional FIPP *p*-cycles in MPLS networks. The pros and cons of unidirectional FIPP *p*-cycles are presented, and illustrated with examples. Overall conclusions and insights are offered in Chapter 7, as well as an outlook for future research.

1.4 Notations

We now state the full set of parameters and decision variables that will be used for the models presented throughout the thesis:

SETS	
$\boldsymbol{S}$	The set of links in the network, indexed by i for a failure link, and j for surviving link.
$\boldsymbol{P}$	The set of eligible (simple) cycles, indexed by k.
$\boldsymbol{P_c \in P}$	The subset of P, includes all eligible (simple) cycles which can protect DRS c, indexed by k as well.
$\boldsymbol{D}$	The set of demand relations, typically indexed by d.
$\boldsymbol{Q}$	The set of all eligible working paths (i.e., working LSPs) in the network, indexed by q.
$\boldsymbol{Q_d}$	The set of all pre-determined eligible paths available to carry the working demand d, also indexed by q.
$\boldsymbol{C}$	The set of all eligible DRSs, indexed by c.
$\boldsymbol{C_d \in C}$	The subset of C, includes all DRSs that contains the working path of demand relation d, also indexed by c.

PARAMETERS	
Δ	A large positive constant (10,000).
$\boldsymbol{\nabla}$	A small positive constant (0.0001).
$\boldsymbol{c_j}$	The cost of a unit of capacity placed on link j. A unit of capacity can be a single channel in a WDM network design or a unit of bandwidth (e.g. 1 Mbps) in an MPLS network design. This generally includes consideration of the technology employed and normally proportional to actual distance.
$\boldsymbol{h_d}$	The number of unit demands for demand relation d.

x_d^k	Encodes the protection relationship between cycle k and working path for demand d by the number of unit demands can be protected by a unit copy of cycle k. This parameter is generated by the Algorithm ***Protect_Cycle_Path_Ind*** in Section 3.2.2. $\begin{cases} 2, if\ working\ path\ for\ demand\ d\ is\ completely\ straddling\ cycle\ \ k \\ 1, if working\ path\ for\ demand\ d\ is\ on\ cycle\ k. \\ 0, otherwise. \end{cases}$
x_q^k	Encodes the protection relationship between cycle k and a candidate path q by the amount of bandwidth can be protected by a unit of bandwidth on cycle k. This parameter is generated by the Algorithm ***Protect_Cycle_Path_Ind*** in Section 3.2.2. $\begin{cases} 2, if\ \ path\ q\ is\ completely\ straddling\ cycle\ k. \\ 1, if\ path\ q\ is\ on\ cycle\ k. \\ 0, otherwise. \end{cases}$
y_q^k	Another form to encode the protection relationship between cycle k and a candidate path q, by the required bandwidth on cycle k to protect one unit of bandwidth on path q. In other words, this constant gives the fraction of the working flow from path q which is carried by cycle k during restoration under the assumption that there is only a single FIPP p-cycle to protect path q. $\begin{cases} 1, if\ \ path\ q\ is\ on\ cycle\ k. \\ 0.5, if\ path\ q\ is\ completely\ straddling\ cycle\ k. \\ 0, otherwise. \end{cases}$
π_j^q	Encodes whether link j is on path q. $\begin{cases} 1, if\ path\ q\ crosses\ link\ j. \\ 0, otherwise. \end{cases}$
π_j^k	Encodes whether link j is on cycle k. $\begin{cases} 1, if\ cycle\ k\ crosses\ link\ j. \\ 0, otherwise. \end{cases}$
∂_{mn}	Encodes the disjointness between two paths. The exact meaning of m and n is case sensitive. They can be used to encode the working path of a specified demand, or represent a candidate path. As discussed in

Section 3.2, mutual disjoint of two paths means there is no common links or transit nodes, and it can be relax to link disjoint only under single link failure assumption (node protection is not requited). Note that m, n is denote to

$$\begin{cases} 1, if \ \ the\ two\ paths\ are\ not\ mutually\ disjoint. \\ 0, if\ the\ two\ paths\ are\ mutually\ disjoint. \end{cases}$$

DECISION VARIABLES

$\boldsymbol{w_j}$	The working bandwidth of link j. (Continuous variable)
$\boldsymbol{s_j}$	The spare capacity (e.g., the reserved bandwidth in an MPLS network case) of link j. (Continuous variable)
$\boldsymbol{g_q}$	The amount of bandwidth on path q. (Continuous variable)
$\boldsymbol{\alpha_q}$	Encode the decision to use path q as one of the working paths. (Binary variable) $\begin{cases} 1, if\ paht\ q\ is\ used\ to\ route\ the\ corresponding\ demand. \\ 0, otherwise. \end{cases}$
$\boldsymbol{\alpha_d^k}$	Encodes the decision to use cycle k to protect working path of demand d. (Binary variable) $\begin{cases} 1, if\ cycle\ k\ is\ used\ to\ protect\ traffic\ of\ demand\ d\ . \\ 0, otherwise. \end{cases}$
$\boldsymbol{n^k}$	The number of unit-capacity of (or the amount of bandwidth on) cycle k used as an FIPP p-cycle for the protection. (Continuous variable)
$\boldsymbol{n_c^k}$	The number of unit-capacity of cycle k used as an FIPP p-cycle to protect DRS c. (Continuous variable)
$\boldsymbol{n_d^k}$	The number of unit-capacity of cycle k used to protect working path of demand d. (Continuous variable)
$\boldsymbol{n_q^k}$	The number of unit-bandwidth of cycle k used to protect path q. (Continuous variable)
$\boldsymbol{\gamma_q^k}$	Encodes the decision to use cycle k to protect path q. (Binary

variable)

$$\begin{cases} 1, if\ cycle\ k\ is\ used\ to\ protect\ traffic\ on\ path\ q. \\ 0, otherwise. \end{cases}$$

Chapter 2 Overview of MPLS Networks and *p*-Cycle Concept

2.1 MPLS Basics

Multi Protocol Label Switching (MPLS) [2] is a protocol tunnelling technique to provide controlled traffic engineering for the flow of packets inside a connectionless IP network. The main objective of this technique is to improve resource utilization and provide assured end-to-end quality of service (QoS) for differential services/users through flexible routing. MPLS aggregates switching and routing functionalities and hence it is often referred to as a layer 2.5 technology. The principal idea is to establish "virtual paths" (i.e., tunnels) within a core IP network and this idea has been adapted from similar concepts such as Virtual Paths/Virtual Circuits in Asynchronous Transfer Mode (ATM) technology. Nowadays, the most widely deployed MPLS service is the Layer 3 Virtual Private Network (VPN) service. Other applications, such as point-to-point Layer 2 transport and Virtual Private LAN Service (VPLS) are also available and growing rapidly [3]. This section provides background for the MPLS technology and extracts relevant details for this thesis.

2.1.1 How MPLS Works

Before discussing specific details relevant to this thesis, we give a schematic overview of MPLS operation. In an MPLS domain, the incoming packets are examined at the ingress router and encapsulated within an MPLS header whose main role is to assign a label to the packet. An important task for the ingress router is to match the packet to a Forwarding Equivalence Class (FEC), which is typically based on the destination address. This is followed by the identification of a specific label that is bound to the particular FEC and this is usually added to the packet in the form of a shim header label. All packets belonging to the same FEC are allocated with the same label, whereas more

granular assignment of packets to FECs can also be applied. Note that MPLS standards define the concept of a label stack to support a hierarchical network structure. To be more precise, the new label is pushed onto the top of the existing label stack, regardless of whether or not the label stack is empty. Then, the labelled packet is forwarded along a Label Switched Path (LSP), where each Label Switch Router (LSR) makes a switching decision according to the top label, until it reaches the egress MPLS node. The forwarding actions at the egress LSR depend on the inner header exposed after popping the outmost label. If the remaining packet has a label, i.e., it is still an MPLS packet, it will continue to do label swapping based on the new label obtained. Another possibility is that the packet turns out to be a pure IP packet. Rather than performing the label switching, the following action should be forwarding the packet according to the destination IP address.

The LSRs serving as ingress and egress nodes are also referred to as head-end routers and tail-end routers respectively and belong to the set of routers known as Label Edge Routers (LERs). It is the role of the ingress LSR to determine the appropriate egress LSR and LSP to that egress LSR associated with a particular FEC. Henceforth, prior to any packets being processed, related information needs to be disseminated from the ingress LSR to the egress LSR, including the binding between FECs and labels, as well as the route of LSPs. This requires a separate control component to implement the signalling and routing functions. In general, there are three related protocols: Label Distribution Protocol (LDP), Resource Reservation Protocol – Traffic Engineering (RSVP-TE), and Border Gateway Protocol (BGP).

Figure 2 shows the basic operations that take place in an MPLS domain.

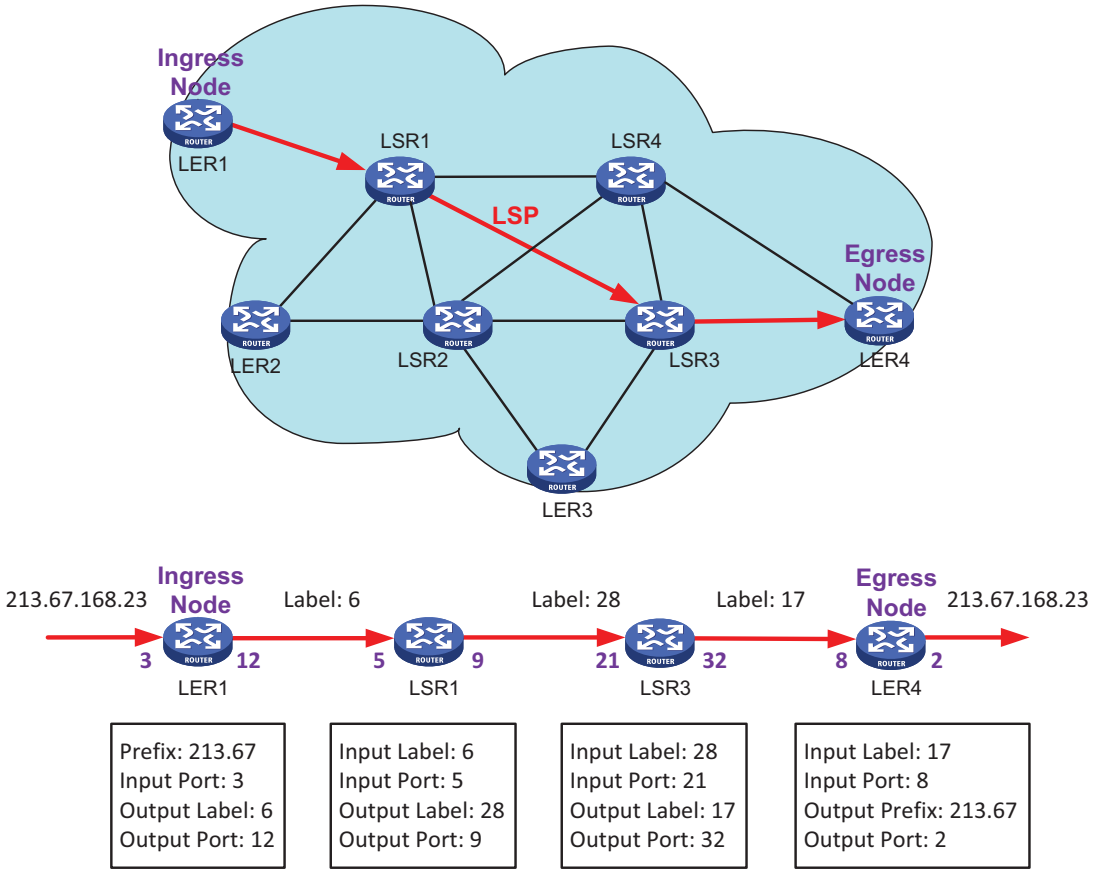


Figure 2 MPLS operation

The ingress node (LER1) takes on the responsibility for analysing the incoming IP packet. It assigns the packet with an appropriate outgoing label 6, according to the destination IP address. Then, the core label switch routers, such as LSR1 and LSR3 in Figure 2, only execute the forwarding process based on the incoming label and the input port. For instance, when a packet arrives with label 6 at LSR1 through port 5, it will be forwarded to the output port 9 with the new label 28. At the egress router LER4, the packet comes from port 8 with label 17 and reaches the end node of the LSP. The label will be popped and the IP packet will be handled in the network layer.

An alternative behaviour is to pop the label at the penultimate LSR of the LSP rather than at the tail-end LSR, which known as penultimate hop popping (PHP). This capability is appropriate due to the fact that the next hop is the actual egress node, and the label no longer needs to be carried. It benefits the implementation in that only a

single lookup is performed at the egress node. Typically, this feature is enabled as the default in relevant label distribution protocols. Due to the label switching technique in use, MPLS yield smaller label switch tables in the core of networks, rather than a large IP routing table. Additionally, the amount of label distribution control traffic is reduced as well, which improves scalability and shortens the processing time.

2.1.2 Forwarding Plane

MPLS provides desired fast forwarding and traffic engineering through interaction of forwarding and control components. To achieve fast forwarding MPLS introduces a label-switching mechanism rather than address matching to determine the next hop for a received packet. The core of its switching functionality is based on labels, the main value in an MPLS header. Figure 3 shows the MPLS shim header which contains four important fields. The MPLS label is a 20-bit, locally significant identifier which represents the packet's FEC. That is to say, there is a binding between the FEC and the label, and is local to the two adjacent LSRs. MPLS packets are forwarded on the basis of labels. MPLS supports a hierarchical label stack, organized in a last-in first-out manner. The S-bit is set on the header of the MPLS packet at the bottom of the stack.

Figure 3 MPLS shim header

A label stack enables a packet to carry information about more than one FEC, allowing it to traverse different MPLS domains or LSP segments within a domain using the corresponding LSPs along the end-to-end path. For example, traffic towards different destinations might traverse an MPLS domain through the same route, where label swapping is used only to get the traffic to an identical egress node. In this case, the particular route can be predefined as an LSP, and the corresponding label can be pushed onto the stack of those packets belonging to the transit flows. The introduction of the label stack reduces the number of labels needed to handle at an LSR. Note that label

stack processing is always based on the current top label. In practice, the 3-bit EXP (experimental field) is used to convey the class of service to be applied to the packet for QoS provisioning. The time-to-live or TTL field is 8 bits long and inherited from conventional IPv4 packets. In traditional IP networks, it is used as an upper bound on the time that an IP datagram can exist in the network. In practice, the TTL field is reduced by one per hop, thus it is also referred to as a hop limit. It serves one main purpose: it breaks routing loops. Routing loops may exist due to misconfigurations, or temporarily exist due to slow convergence of the supporting routing algorithms when a failure occurs or restoration is in progress. TTL is also used for other functions including multicast scoping, and route tracing. In MPLS, TTL processing depends significantly on the operating mode of the corresponding LSP. We shall refer to this again later.

An LSP can be regarded as a path consisting of a sequence of LSRs that traffic streams are switched along. Generally, it can be hop by hop routed or explicitly routed, which is referred to as route selection for an LSP. A hop by hop LSP is an LSP where each node chooses the next hop independently as performed in traditional IP networks, while in an explicitly routed LSP, each LSR has no authority to decide the next hop except, typically the LSP ingress router which takes charge of specifying LSRs in the LSP. The latter is beneficial to traffic engineering and supported with RSVP-TE, as it offers great control over the forwarding behaviour. It can be used to avoid congested links, provide better performance and guarantee resource availability. A tunnel usually describes the specific way from one router to another non-consecutive router, which may or may not be the ultimate destination of the packets. When label switching, rather than network layer encapsulation, is used to cause a packet to travel through a tunnel, it is said that the tunnel is implemented as an LSP, and this is commonly referred to as an LSP tunnel. In this thesis, LSP and tunnel are deemed to be equivalent terms and used interchangeably, all of which are considered to be explicitly routed.

Additionally, there are two basic models of tunnel operation in MPLS networks: the Pipe and the Uniform models[1] [4, 5]. They are conceptual models developed from the differentiated services perspective. Within the Pipe model, an MPLS network acts like a circuit when packets traverse the network, such that only the LSP ingress and egress routers are visible to nodes that are outside the tunnel. Packets transiting the LSP see the tunnel as a single hop regardless of the number of intermediate LSRs. Thus, the TTL field can be independent from that in the IP header leaving the egress router to decrement the original datagram's IP hop limit by one. On the other hand, the Uniform model makes all the nodes that an LSP traverses visible to the outside. When an ingress router first attaches a label to a datagram, a TTL field should be initially loaded from the network layer header TTL field. As the packet traverses its LSP, its TTL value gets decremented by one at each LSR-hop. Once the value reaches zero before the packet has reached the destination, the LSR should pop the entire label stack, return an ICMP time exceeded message to the packet's source and finally discard the packet. Alternatively, the TTL value should be copied into the network layer header TTL field when the packet emerges from its LSP normally.

At each LSR, the label swapping is performed through a lookup of the incoming label. This yields a corresponding output label that must be used for the next leg of the LSP with an output interface on which the packet should be forwarded. The data for lookup is stored in a series of forwarding tables, also referred to in the literature as the Label Information Base (LIB). More specifically, there are three main tables according to Specification [2]. The Next Hop Label Forwarding Entry (NHLFE) defines the next hop and the appropriate operations to perform on the packet's label stack. The other two deal with mapping the incoming packet to a set of NHLFEs. The Incoming Label Map (ILM) handles labelled packets. It takes charge of mapping each incoming label to a set of NHLFEs, while the FEC-to-NHLFE Map (FTN) is responsible for packets that are

[1] A short variation of the Pipe model is defined as Short Pipe model.

without labels for mapping FECs instead. Take a labelled packet for example; the LSR examines the label at the top of the label stack. ILM is used in this case to map this label to an NHLFE which decides where to send the packet and performs any required operations on the label stack. Then, the packet is encapsulated with a new label stack and forwarded through the given port. Instead, if the incoming packet is unlabelled, the LSR uses the FTN to map the FEC of the packet, which is determined from the network layer header, to an NHLFE.

2.1.3 Control Plane

So far we have introduced the important concepts in the forwarding plane which is based on a label switching paradigm in MPLS networks. The signalling and routing functions are provided via a separate control plane composed of three main parts:

- Information distribution
- Path computation
- Label distribution and path setup

In general, there are two ways to compute a path along which labels are swapped. One is centralized or offline path computation, whereas the other is referred to as distributed or online path calculation. With regard to centralized path computation, all LSPs are simultaneously computed by an offline management tool and configured explicitly. In this way, more assured control can be placed over paths with a global view of reservations and bandwidth availability. Moreover, both normal and failure states can be taken into account to enhance the overall survivability of the network. By contrast, in the distributed path computation, every LSR dynamically computes paths that satisfy a set of constraints according to the current network state. Specifically, it requires the network topology and resource information to be distributed on the fly. Typically, the network information is carried by link state routing protocols: Open Shortest Path First (OSPF) or Intermediate System to Intermediate System (IS-IS). Both of these protocols

are extended to advertise link characteristics, reservation states and even policy information for the purpose of advanced traffic engineering. Various types of Constrained Shortest Path First (CSPF) algorithms, as an extension of the shortest path algorithm, can be used to calculate paths taking different factors into consideration. It is possible to exclude or prefer paths while pursuing a best match for the constraints based on used and available bandwidth, priorities, link colouring and other administrative attributes etc.

As mentioned in Section 2.1.1, the label has a local meaning just between two adjacent LSRs. These two routers must agree on a label to identify a particular LSP. In other words, it requires coordination for binding a label to a FEC between each pair of LSRs. Label distribution protocols are introduced for this purpose. A dedicated protocol known as the Label Distribution Protocol (LDP) was once invented specifically for MPLS. It has a similar function to discover neighbours and learn network topology as for OSPF, but with no routing functionality embedded. The extension of LDP – CR-LDP is defined to support constraint-based routing, and supplements traffic engineering.

Another pre-existing protocol – BGP supports multiple address families per se and hence it is quite straightforward to add and advertise labels as a new address family. The main benefit of using BGP to distribute label bindings is its ability to establish LSPs that cross Autonomous System boundaries.

Likewise, RSVP was also devised before MPLS came into being. It has been developed to establish and maintain LSPs with or without associated bandwidth reservation for MPLS networks. The extensions to RSVP for LSP tunnels are referred to as RSVP-TE, where the key application is traffic engineering [6]. Since the abandonment of CR-LDP in 2003, RSVP-TE has become the dominant protocol for MPLS traffic engineering. One major property of an RSVP-TE signalled LSP is the capability to deviate from the

path that would be prescribed by the IGP. More generally, RSVP-TE supports the instantiation of explicitly routed LSPs, which allows the implementation of a variety of network performance optimizations. Not only traffic engineering, but also enhanced survivability can be expected. Hence, all LSPs in this thesis are assumed to be RSVP-TE signalled LSP tunnels that are explicitly routed.

To establish an LSP tunnel, the ingress LER needs to create and send an RSVP PATH message, which is addressed to the same destination as the flow itself, i.e., the egress LER. A LABEL_REQUEST object is inserted into the PATH message to request a label binding for this particular path. Furthermore, another EXPLICIT_ROUTE object (ERO) can be added to specify the route as a sequence of LSRs. The explicitly routed path can be either strict or loose, specifying all or several of the LSRs in the LSP respectively. Typically, the pre-selected route has a high possibility of meeting the tunnel's QoS requirements, or that it makes efficient use of network resources, or that it satisfies some policy criteria. Unlike the data flow, transit LSRs along the path must examine the contents of the message and make necessary modifications if they are allowed to do so. In response to the Path message, the egress LER sends an RSVP RESV message, having a RSVP_LABEL object inside. The RESV message is sent back to the upstream neighbour, rather than being addressed directly to the ingress LSR. The node receives an RESV message and uses the assigned label for outgoing traffic associated with this tunnel. In addition, it allocates a new label and sends it to the one-hop upstream LSR within a new RSVP message created by itself. After assigning the new label, there is sufficient information for the node to update its ILM, which is used to map incoming labelled packets to a NHLFE. In the same way, the RESV message is handled at each intermediate LSR and propagates until it reaches the ingress router. Hence, a label switched path is successfully established. Note that those RESV messages follow the exact reverse path of the PATH message. Both of these two types of messages travel on a hop-by-hop basis since label setup is required at each node that they cross, as well as

other information e.g. bandwidth reservation.

A key issue regarding RSVP is the manner in which it handles state management in the LSRs. It is a soft state protocol. It requires a periodic exchange of messages once an LSP is established, in order to maintain its state. The information exchanged with RSVP is assumed not to be permanent. In contrast, LDP is a hard state protocol, where LSRs assume the information will be retained indefinitely within the peer LSR, unless it hears otherwise. As a result, RSVP has to refresh states by periodically sending PATH and RESV messages for each active LSP. Although such an operation makes RSVP responsive to dynamic networks, the constant refresh of Path and RESV messages is unnecessary in the context of MPLS networks, on account of the fact that LSPs are relatively static and stable. It is clear that the processing overhead of such a scheme becomes a heavy burden for a router maintaining a large number of LSPs and this can incur scalability problems. Several 'Refresh Reduction Extensions' to basic RSVP were proposed to reduce the overhead, such as Bundles, message_ID extensions etc. Among those techniques, one promising method is to introduce a Summary Refresh message, which allows multiple RSVP sessions to have their state refreshed by a single message sent between RSVP neighbours within the refresh interval [7].

2.2 MPLS Failure Recovery Mechanisms

Based on the nature of recovery that is desired, we can identify two different forms of recovery respectively known as local and global recovery. Typically, local recovery consists of link recovery and node recovery, while global recovery is equivalent to path recovery. An MPLS protection domain is defined as the set of LSRs over which a working path and its corresponding protection path are routed. Among those LSRs, there are two special functional LSRs [8]:

- Path Switch LSR (PSL)

A common LSR is located on both the working path and its corresponding recovery path and responsible for switching or replicating the traffic from the working path to its backup path in the case of a failure.

- Path Merge LSR (PML)

  PML is an LSR that terminates the backup tunnel and is responsible for receiving the traffic that comes from the recovery path. It either merges the traffic back onto the working path, or, if it is itself the destination, passes the traffic to higher layer protocols.

In local recovery, only the affected network elements (e.g., links or nodes) are bypassed by the recovery paths. Indeed, the PSL and PML are chosen to be as close to the failed network element as possible. For instance, if a single link fails, a recovery path is set up between the nodes adjacent to the failure. Instead, the complete working path is bypassed in terms of global recovery. Actually, the PSL and PML will coincide with the source and destination of the working path, respectively. Since the PSL is much closer to the point of failure in a local recovery, it can be quickly informed of the fault occurrence either by failure detection or through notification. This leads to superiority in recovery time against global recovery. However, the resulting entire path is often longer than needed and may also lead to loo*p*-back or back-hauling. These suboptimum recovery paths require more capacity than that in global recovery which benefits from the nature of network-wide optimization. Moreover, global recovery has a higher probability to survive successive failures. In the next subsection, main MPLS-based failure recovery mechanisms are presented, as well as criteria for the evaluation of recovery mechanisms.

2.2.1 Evaluation Criteria

A variety of different recovery mechanisms have been proposed for MPLS networks and each of them has its own strengths and weaknesses. In the following, we list some

important criteria that allow us to weigh their performance and assess pros and cons of a given recovery mechanism. These criteria can also be regarded as objectives in designing and planning of telecommunication networks. Some of them are contradictory and trade-offs are inevitable to enhance the overall performance of a recovery mechanism.

Failure Coverage

Recovery schemes may offer various types of failure coverage, which is defined by several metrics. Firstly, recovery schemes may account for different fault types, for example, link failure, node failure, both of these, and even degraded services. Also, multiple concurrent faults, such as double link failure, can be taken into account during survivable network design in order to achieve very high availability. Recovery mechanisms may completely or partially cover the failure scenario. For instance, the recovery scheme may be able to recover only a percentage of the affected traffic which may be attached to a high priority. Instead of 100% coverage, a certain portion of nodes might be considered as within a node failure recovery scheme. In addition, some recovery schemes are inherently bandwidth guaranteed, whereas others can or cannot provide enough backup capacity to reroute all affected traffic depending on the situation. For a given fault, there may be one or more recovery paths.

Capacity Requirements

Recovery schemes may require different amounts of backup capacity in the event of a failure. Backup capacity is dependent on the traffic characteristics of the network, on the particular protection plan selection algorithms, as well as the signalling and re-routing methods [8]. Capital cost does correlate with capacity, especially in long-haul networks. As a result, we prefer to optimize resilient network design with the goal of reducing the cost of backup or total capacity. *Redundancy* is an important characteristic of networks, as well as a measure of architectural efficiency for survivable networks. It

indicates how efficient a design is in its use of backup capacity and thus, often used as a surrogate for cost in basic comparative studies in which exact cost data is not available. The logical redundancy of a network [9] is defined as the pure ratio of backup to working capacity, as shown in Equation (2.1). It does not take any distance effects into account.

$$\mathcal{R} = \sum_{j \in S} s_j \Big/ \sum_{j \in S} w_j \tag{2.1}$$

$S - the\ set\ of\ spans\ in\ the\ network$

$w_j -\ the\ number\ of\ units\ of\ working\ capacity\ on\ span\ j$

$s_j -\ the\ number\ of\ units\ of\ backup\ capacity\ on\ span\ j$

Recovery Time

Another important criterion for a recovery mechanism is the recovery time. It is defined as the time between a network fault and the point at which a recovery path is activated and the traffic starts flowing on it. It is the sum of the fault detection time, hold-off time, fault notification time, recovery operation time, and the traffic restoration time. These different phases are shown in Figure 4 [8].

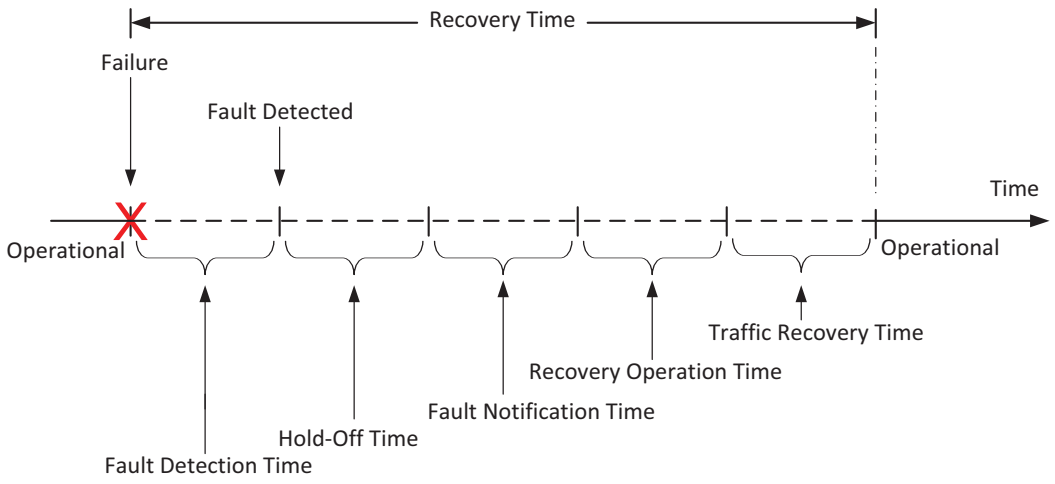


Figure 4 Recovery cycle [8]

Once a network failure occurs, it takes some time for a neighbour node to detect the fault by MPLS-based recovery mechanisms. This time interval is defined as the *Fault Detection Time*. It may depend on the lower layer protocols, on the frequency of signals sent, on the fault diagnosis process, and so on. After the failure is detected, the node that detected the fault may or may not wait some time before taking any MPLS-based recovery action, according to the configuration. Introducing such a hold-off time allows a lower layer recovery scheme to take effect and repair the fault. Then it starts sending notification messages (i.e., the Fault Indication Signal) towards other nodes to inform them for the upcoming recovery action. Dampening techniques can be used to help stabilize the network in case of a flapping resource. The time between the first and last recovery actions is referred to as the *Recovery Operation Time*. During this period, nodes exchange messages in order to coordinate the recovery process. After the last recovery action, the affected traffic begins to be carried on the recovery path. However, it could still take a while for the traffic to be completely restored. This interval is called the *Traffic Recovery Time* and depends on the location of the fault, the recovery scheme in use, and the propagation delay along the recovery path. Such a succession of phases begins with fault identification and ends with traffic recovery, and is referred to as the *Recovery Cycle*. Evidently, the smaller the recovery time, the less the services are impaired by the network failure.

Scalability

A recovery mechanism is said to be scalable and future proof if the performance does not depend too much on the size of the network and the traffic volume to be transported. In fact, as the network grows and the amount of traffic increases, the performance of a recovery scheme, such as the recovery time, the required backup capacity and state overhead, may change considerably. The state corresponds to the information for maintaining recovery paths that are stored in the individual network element and grows with the number of recovery paths. The exact state required by the recovery scheme

varies. For some schemes the state overhead may increase very quickly with growing network or traffic size, whereas other recovery schemes experience only a modest state overhead increase.

Signalling plays a crucial role in the recovery operation and signalling messages are required to be exchanged between nodes before or during the recovery process. Both of the fault detection and notification rely on signalling. The number of messages used to establish the recovery path is proportional to the number of recovery paths. A huge volume of signalling messages will incur large consumption of resources in terms of bandwidth, CPU usage, memory etc. Hence, it has a great impact on the scalability of a recovery scheme as well.

The performance of a recovery mechanism can also be compared according to other criteria, such as the packet loss, packet reordering and duplication, additive latency and jitter, network stability etc. Last, but not least, it is a useful feature to take appropriate recovery actions for traffic with different recovery classes. In other words, the capability to support Quality of Protection is another essential measure for evaluation of a recovery scheme.

2.2.2 Global Repair

As discussed, global repair devotes itself to protecting entire working paths, against any network link or node failure on the path. Note that the traffic cannot be restored in case of faults occurring at the ingress or egress LSR. Traffic loss is inevitable for such types of failure. In either global restoration or protection schemes, the ingress node always takes the role of PSL when the FIS arrives, irrespective of where the failure occurs along the working path. In the context of IP/MPLS network, the FIS can be either an IGP update or an RSVP PATH ERROR message. In both cases, this involves propagation of the failure information all the way back to the head-end LSR. Thus, the

FIS delivery is of the utmost importance, not only because the success of it triggers the following recovery actions, but also because the failure propagation time amounts to a large part of the overall recovery time, in particular, for global protection schemes.

2.2.2.1 Global Restoration

Global restoration is usually the default recovery technique for MPLS networks. Once the head-end LSR is notified about the failure occurrence by means of the FIS, it recomputes the path using the CSPF algorithm, taking into account the constraints and available resources in the network. Then it signals the LSP along the new alternative path. During the recovery path computation, relaxing constraints can be allowed as long as it conforms to service level agreements, and might be necessary in case that the failure causes the inability to find an alternative path. CSPF duration time is a function of the network size and the CSPF algorithm in use. Moreover, it is worth noting that one CSPF process must be triggered per affected LSP. If multiple LSPs starting on a single head-end LSR fail concurrently, the router has to compute a path for each of them. Nevertheless, an average CSPF computation time on a network with hundreds of nodes rarely exceeds a few milliseconds [10].

2.2.2.2 Global Protection

Global path protection is also referred to as end-to-end path protection. The main difference between global restoration and protection is the action taken by the ingress router when the LSR is informed about a fault. Since the backup path is pre-signalled, the only thing that is needed to be done is the activation of the protection switch, rather than performing a routing computation on the fly as in global restoration. In the case of global 1:1 protection, an alternative disjoint backup path is required for each active path.

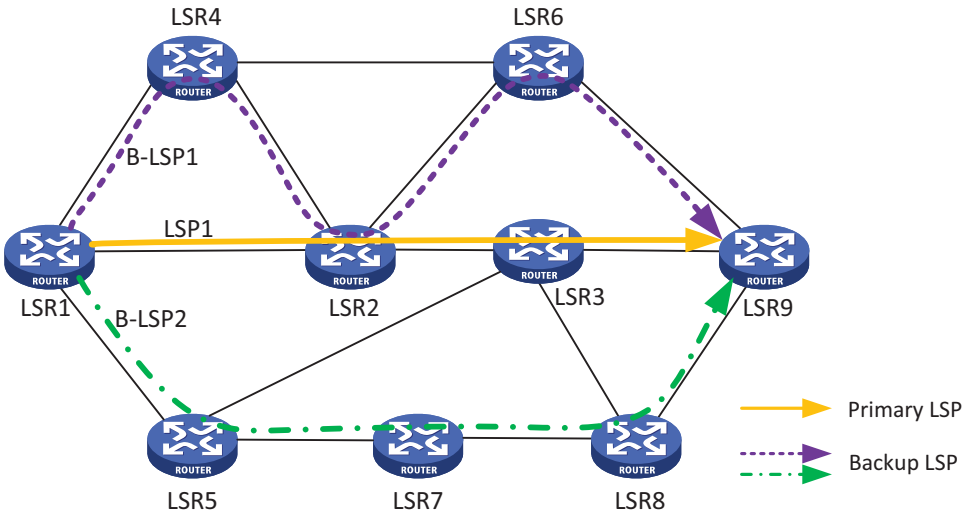


Figure 5 Global 1:1 path protection

The backup LSP, also known as the secondary LSP, must be link disjoint or node disjoint from the primary LSP, in order to cover the link failure only, or both link and node failure respectively. As depicted in Figure 5, B-LSP1 is a link-disjoint backup LSP for the primary LSP, while B-LSP2 is both link and node disjoint from LSP1. The global protection is usually less capacity efficient than the restoration counterpart, because the latter can take the exact failure into consideration and has the capability to reuse the surviving working capacity of a failed path[2]. However, it outperforms the global restoration in terms of recovery time, which attributes predominately to pre-signalling of the backup LSP.

2.2.3 Local Repair

Local repair, also known as local recovery, is intent on protection against a link or node failure. In this way, the failure propagation time can be reduced to a minimum, because the LSR immediately upstream of the fault functions as the PSL and initiates the recovery operation. That node is known as the point of local repair (PLR). Despite the fact that local repair can be divided into local restoration and local protection, only local protection has won wide attention in the research community. In local protection, the

[2] This feature is known as “stub release” or “stub reuse”.

traffic is quickly rerouted around the point of failure and backup LSPs are pre-signalled in advance of a fault. For this reason, the scheme is referred to as fast reroute (FRR), which enables the redirection of traffic onto backup LSP tunnels within tens of milliseconds [11]. It offers transparency to the ingress node and faster restoration times. As a consequence, traffic loss can be avoided as much as possible. Two different methods for fast reroute are described in the following subsection. Note that they are applicable only to explicitly routed LSPs and supported by extensions to RSVP-TE.

2.2.3.1 Facility Backup

As the term suggests, facility backup focuses on protection against network resource failure, e.g., a link or a node. The key aspect of this facility is the employment of bypass tunnels. A bypass tunnel is used to protect a set of primary LSPs that pass over a common facility. Take a link failure for instance, as shown in Figure 6.

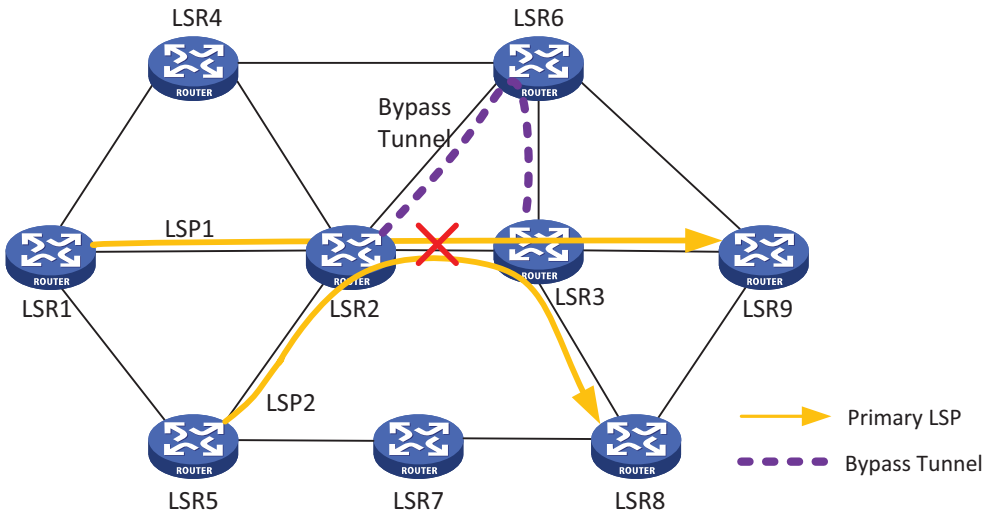


Figure 6 Facility backup operation in case of a link failure

The bypass tunnel is shared by two primary LSPs that use the link between LSR2 and LSR3. Undoubtedly, as the PLR in this case, LSR2 must be the head-end router of the bypass tunnel, and takes the responsibility of switching the traffic that comes from LSP1 and LSP2 to the bypass tunnel during the recovery process. On the other hand, the

bypass tunnel must intersect the path of the original LSPs somewhere downstream of the unreliable link, in order to redirect the traffic back. LSR3 is the only router in the topology that is guaranteed to lie along the path of all the LSPs being protected. The router where the backup tunnel terminates is also called the merge point. Similarly, the method offers protection against node failures by virtue of the next hop bypass tunnel which terminates on the neighbour of the failed node. A major drawback of this mechanism is that the resulting recovery path might be suboptimal due to the use of the bypass tunnel.

2.2.3.2 One-to-One Backup

Instead of creating a single bypass tunnel to protect a network entity, there is a separate detour path for each protected LSP at a PLR within one-to-one backup.

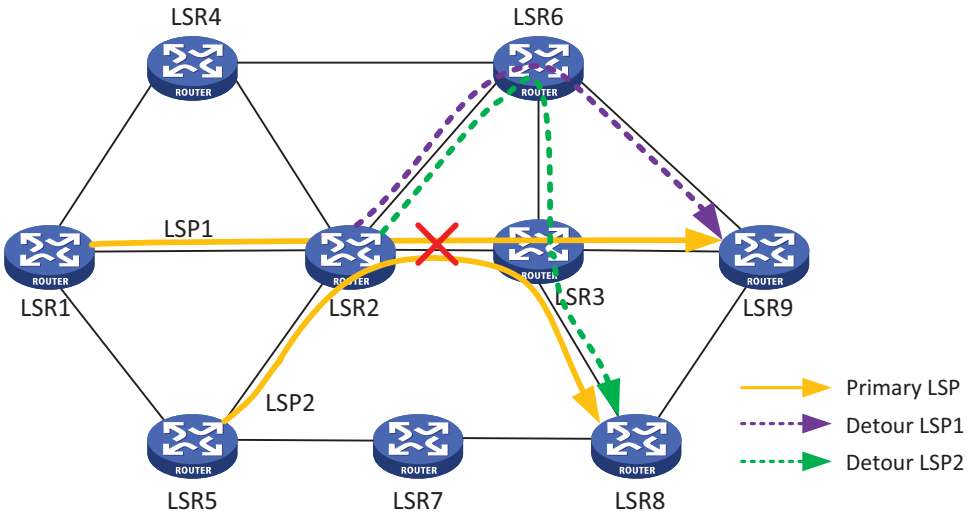


Figure 7 One-to-one backup operation in case of a failure

Figure 7 shows the detour paths created for LSP1 and LSP2 at LSR2 in case that the link between LSR2 and LSR3 fails. A detour path is dedicated to each LSP and no longer restricted to rejoin the original LSP at some given point. It can also be created to detour around a particular node for the sake of node protection. This mechanism provides the possibility of selecting optimal paths from the PLR to the corresponding egress LSR for each LSP under the protection. However, it results in many more backup

paths than for the facility backup case.

2.3 *p*-Cycles

p-Cycles is a well-known paradigm that has emerged for survivable networking, and extensively introduced in published books. They were a preconfigured protection cycle, proposed to serve in optical networks initially [12-14], although it is recognized as a universal protection scheme applicable for nearly all types of resilient networks, such as WDM, IP, MPLS, SDH, SONET. A *p*-cycle is only constructed from the available spare capacity and comprises of a single unit of spare capacity on each span that it crosses. The operation of a unit capacity *p*-cycle is illustrated in Figure 8. *p*-Cycles operate similarly to self-healing rings in that both use a cyclical structure and provide one recovery path for each on-cycle span failure, as shown in Figure 8b.

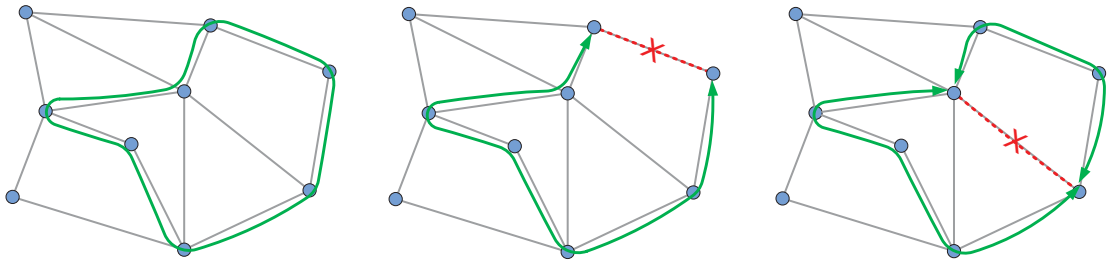


Figure 8 a) A *p*-cycle b) Failure of an on-cycle span c) Failure of straddling span

On the contrary, it can be accessed for the protection of straddling spans, which are spans that have two end nodes on the cycle but are not themselves as a part of the *p*-cycle. An example is presented in Figure 8c. The significance is that it provides two recovery paths for each straddling span failure due to the fact that the *p*-cycle itself remains intact. The ability to protect the straddle spans contributes to its high capacity efficiency. Also, the fact that it permits shortest-path routing of working paths adds further to the overall network capacity efficiency. Theoretical studies in [15] reveal that a fully pre-configured *p*-cycle network has the same lower bound on the redundancy as

a span-restorable mesh network. The well-known lower limit is formulated as $1/(\bar{d}-1)$, where $\bar{d}$ is the average nodal degree of the network. This is also confirmed further with some experimental observations based on the results from [1]. This mechanism offers fast restoration time (50-150ms) comparable to that of ring networks, because calculation and connection of recovery paths is carried out in advance of any failure. In fact, the only actions required are the failure identification and the protection switching at the two end nodes of the failed spans, in case of a failure. Therefore, p–cycles are no more complex than ring systems from the operational perspective. The elite way to conclude the property of *p*-cycles is "ring-like speed with mesh-like capacity". Note that *p*-cycles are designed to protect the network against a single failure. For the period of fault repair, a readjustment of the *p*-cycle scheme can be performed temporarily in either a global or an incremental manner, in order to protect the vulnerable network [16]. There have been various extensions to the basic concept of *p*-cycles, which are now referred to as span- (or link-) protecting *p*-cycles for clarity. A comprehensive survey of *p*-cycles can be found in [17, 18].

Node-Encircling p-Cycles

Node-encircling *p*-cycles[19] (NEPCs) are an exploration of node protection, especially against router failure in the MPLS layer. It is, by definition, composed of all nodes that were adjacent to the protected node, but not the protected node itself. This node is said to be "encircled" within the corresponding NEPC. The key to an NEPC is that it assuredly intersects any flow transiting the encircled node, and hence provides one or more detour paths for each of them readily. A simple cycle, however, cannot always be accessible for a node, especially in sparse networks. Albeit the fact that the non-simple cycles are also counted as candidates, it adds greatly to the algorithm complexity. Obviously, it is not capacity efficient to combine a set of NEPCs (for node recovery) with span-protecting *p*-cycles in a complete network design.

FIPP p-Cycles

Failure-Independent Path-Protecting (FIPP) *p*-cycles [20-22] are an attractive extension of *p*-cycles to path protection. They are capable of offering similar on-cycle and straddle protection to entire working paths, while holding the property of end-to-end, failure independent protection against either span or node failure. Since the scheme is the main focus of this thesis, the details regarding its principles and mathematical models are deferred to Chapter 3.

Flow p-Cycles

Another type of *p*-cycle is known as the flow-protecting *p*-cycle[23], which sets out to protect path segments for contiguous working flows. It renders a generalization of the *p*-cycle concept, in which pure span and pure path-protecting *p*-cycles can be viewed as two extremes. Rather than protection only for an individual span or an end-to-end path, flow *p*-cycles cope with a portion of a path with any size. In other words, arbitrary path segments are allowed to be protected elements. Results in [23] show that the spare capacity efficiency is significantly higher than that of span-protection *p*-cycles, which is attributed to a more flexible protection. Actually, the spare capacity efficiency of flow *p*-cycles lies between the span-restorable and path-restorable networks. As a path-oriented protection scheme, it can inherently overcome a node failure (as long as it is not one of the end nodes of the segment), or even have potentially higher possibility to survive in the case of multiple failures. A significant drawback is that flow *p*-cycles need to be employed in a failure-specific way and hence require a massive number of operations in real time. Selective use of flow *p*-cycles for a specific set of high-prioritized flows would be more manageable and practical from the operational point of view.

It is worth mentioning that a “two-hop strategy” was devised lately, and introduced a new approach to node protection with span protecting *p*-cycles in [24]. In brief, it

explores the intrinsic two-hop node protecting capability of *p*-cycles, which has been overlooked in the past. A 2-hop path segment is defined as two links that are adjacent to a protected node. Similar to ring-like behaviour, a span-protecting *p*-cycle can protect on-cycle node failure; whereas, in contrast, it provides two recovery paths for those off-cycle node failures. The prerequisite is that a 2-hop path segment related to the protected node intersects the corresponding *p*-cycle at the two end nodes of the path segment. This scheme can be viewed as a special case of flow *p*-cycles in the sense that it restricts the failed flows to be considered and restored strictly only as if they were two-hop segments. Indeed, it offers capacity efficiency comparable to that of flow *p*-cycles, while it simplifies fault identification and real-time operations. Results in [24] show a tendency that span protecting *p*-cycles (with the 2-hop node protection feature) are attractive to networks with a small number of candidate cycles, while FIPP *p*-cycles give rise to least costs for cases involving more candidate cycles.

2.4 Summary

In this chapter, we introduced some basic concepts of MPLS networks and briefly explained the label switching technique. With the favourable property of explicit routing, RSVP-TE lends itself to traffic engineering and fast rerouting capabilities that operate independently of traditional IGPs. Main MPLS-based recovery mechanisms were also reviewed, as well as some specific criteria that help to make a judicious comparison between these schemes. A detailed comparison on the scalability of these schemes is postponed to Chapter 4.

The *p*-cycles scheme is a fairly well-known network protection method that has desirable properties such as high capacity efficiency and fast restoration speed with no loss of simplicity. Extensive research has been carried out and broadened the theory of *p*-cycles, which has lead to the emergence of a number of different types of *p*-cycles.

The common ground for these *p*-cycle extensions is that they inherit the on-cycle and straddle protection for the relevant network elements and require protection switching at the two end nodes of the failed component in the event of failure.

Chapter 3 Studies of Failure-Independent Path-Protection (FIPP) *p*-Cycles

3.1 Linear Programming Fundamentals

Optimization, as a branch of Operational Research (OR), is an important technology in the support of resilient network planning. During the past few decades, a large number of approaches have been developed to solve general optimization problems. As the complexity of problems has increased and the existing solution techniques have been unable to solve the size of practical problems, it has been found necessary to develop more sophisticated heuristic approaches. Among the many new techniques we have Simulated Annealing, Generic Algorithms, Tabu Search[25] and more recently, Particle Swarm Optimization[26]. These techniques are being constantly improved in order to provide quicker and better solutions. Nevertheless, Linear Programming (LP) is still the dominant method for optimization problems in practice and the main approach used for obtaining resilient network designs in this thesis. Note that in this context, the term "programming" is not associated with the current common use of this word, viz: creating a sequence of instructions to enable some functionality of a computer. Instead, here the term "programming", relates to the concept of scheduling or planning.

This thesis focuses on the *application* of LP techniques to the solution of practical design problems in communication networking, not on the techniques themselves. This section is intended to provide an elementary, user level understanding of LP methods and terminology in order to aid the reader in understanding the models and solutions presented in this thesis. A broad and more elaborate description of optimization approaches in the networking field can be found in [25].

3.1.1 Linear and Integer Linear Programming

Optimization, or mathematical programming, refers to choosing the best element from some set of available alternatives. In the simplest case, this means solving problems in which the main goal is to minimize or maximize a real function (referred to as the *objective function*) by systematically choosing the values of real or integer variables from within an allowed set. The policy is imposed by making those decision variables subject to a series of equations, which are referred to as the *constraints* of the problem.

Linear Programming (LP), is a type of mathematical programming, that is concerned with the optimization (minimization or maximization) of a linear function while satisfying a set of linear equality and/or inequality constraints [27]. Generally, the decision variables may take on continuous non-negative real values. Notice that a linear program refers to an instance of a linear programming problem, whereas linear programming is relevant to the process of solving such an instance. Integer linear programming (ILP) and Mixed Integer Programming (MIP) are linear programs with all or part of the variables restricted to integer (or discrete) values respectively. A special case of integer programming is Binary Integer Programming (BIP), also called Zero-One or 0/1 Integer Programming, where variables are required to be 0 or 1, rather than arbitrary integers. Variables constrained in this way typically represent no/yes decision outcomes. Most of the mathematical models in this thesis are mixed binary integer programming or simply LPs.

More specifically, each LP/ILP formulation has the following main components: the objective function, the decision variables, the data set, the constraints and lastly the variable bound functions. The objective function describes the goal of the optimization problem as a linear weighted function of the unknown variables. The data set is classified into two separate, but related parts: sets and parameters. A set is a collection of objects that is to be considered in the optimization. Parameters, which may be

constants or characteristics of those elements in the sets, are input data. Bounded variables can be interpreted as a special set of constraints. These two together assert relationships between variables that must be adhered to.

The standard format of a linear program can be written using matrix notation as follows:

Minimize

$$\boldsymbol{c^T x} \tag{3.1}$$

Subject to

$$\boldsymbol{Ax = b} \tag{3.2}$$

$$\boldsymbol{x} \geq 0 \tag{3.3}$$

Where $\boldsymbol{c}$ is an *n*-dimensional column vector of costs or weights, $\boldsymbol{x}$ is an *n*-dimensional vector of decision variables, $\boldsymbol{A}$ is an *m x n* matrix of linear constraint coefficients and $\boldsymbol{b}$ is an *m*-dimensional vector of constraint resources. $\mathbf{0}$ represents the *n*-dimensional vector consisting of all zeros.

The problem can also be conveniently represented via the column of $\boldsymbol{A}$, e.g., let a_j be the *j*-th column of $\boldsymbol{A}$. The problem can be formulated alternatively as given below.

Minimize

$$\sum_{j=1}^{n} c_j \cdot x_j \tag{3.4}$$

Subject to

$$\sum_{j=1}^{n} a_j \cdot x_j = b \tag{3.5}$$

$$x_j \geq 0, \qquad j = 1,2, \dots, n. \tag{3.6}$$

3.1.2 Solution Approaches

This section deals with various approaches to solving an LP. To achieve better understanding, it starts with the modelling and graphical solution of a two variable problem which, though highly simplified, provides a concrete understanding of the basic concepts of LP. Then, it goes on to illustrate the general approach that underpins the method developed by G. Dantzig known as the Simplex Method. Finally, there is a brief discussion about standard approaches to Integer Programming.

3.1.2.1 Geometric Solution

The example is a two-variable LP model for a diet problem [28]. Consider a person who uses a special daily feed, which is a mixture of corn and soybean meal. He needs at least 800lb of such feed with dietary requirements of at least 30% protein and at most 5% fibre. The compositions in percentages of the two foods are given in Table 1.

Table 1 Compositions and cost of a special feed

Feedstuff	Protein (%)	Fibre (%)	Cost ($/lb)
Corn	9	2	.30
Soybean meal	60	6	.90

So how does he determine the daily minimum-cost feed mix? The objective of this problem is to minimize the total daily cost. The decision variables are the amount (in lb) of corn and soybean meal he should take each day, defined as x_1, x_2 respectively. Obviously, there are three constraints in total. The first one reflects the minimal daily amount needed, while the remaining two originate from his dietary requirements. Although it is a quite simple linear programming problem, the set concept is revealed as a set of two different foods: corn and soybean meal. Parameters include the proportion of ingredients (i.e., protein and fibre in this case), and the unit cost of these two foods etc. The only two variables in the problem are required to be non-negative real values,

and this is interpreted in value-bound functions. Therefore, after several simple operations, the LP model finally becomes:

Minimize

$$z = .3x_1 + .9x_2 \tag{3.7}$$

Subject to

$$x_1 + x_2 \geq 800 \tag{3.8}$$

$$7x_1 - 10x_2 \leq 0 \tag{3.9}$$

$$3x_1 - x_2 \geq 0 \tag{3.10}$$

$$x_1, x_2 \geq 0 \tag{3.11}$$

Figure 9 provides the geometric solution for this model. The feasible solution space is restricted by the three constraint equations resulting in the shadowed area. The dotted lines, working as isolines, represent objective function values for different cases. The points have smaller values on those dotted lines which are closer to the origin point. Thus, the optimal solution is the intersection of the two lines which conform to the Equation (3.8) and (3.9), as indicated with a red circle in Figure 9. The resulting minimum cost of the feed mix is 437.65 dollars per day with 470.6lb corn and 329.5lb soybean meal.

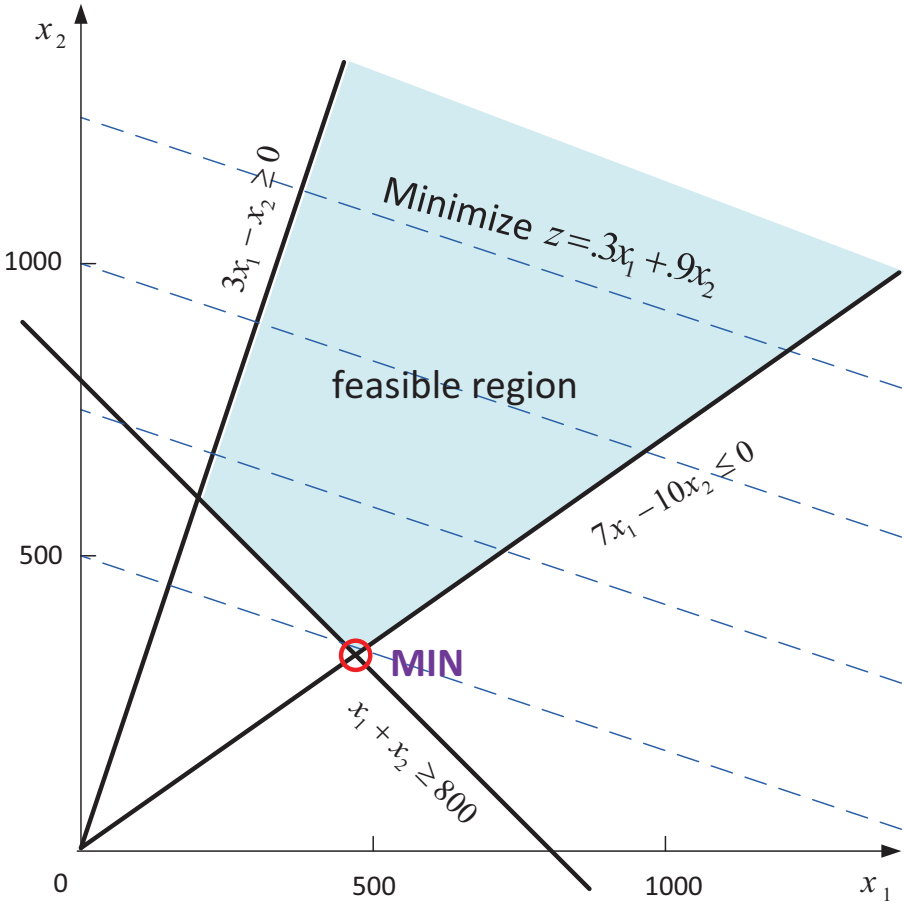


Figure 9 Geometric solution of the diet model [28]

3.1.2.2 The Simplex Method

As can be observed, the optimal solution exists at an extreme point, and potential optimum points lie along the borders of the feasible region. This is the basis for the original idea of the Simplex Method (developed by G. B. Dantzig in 1947), and which has become the standard technique for solving LP problems. In order to be solved using the Simplex method, LP problems are modelled and transformed into a standard form, as shown in Section 3.1.1, involving a nonnegative RHS and variables. An LP problem is given below to illustrate how the Simplex method works.

Maximize

$$z = 5x_1 + 4x_2 \quad \Rightarrow \quad z = 5x_1 + 4x_2 + 0s_1 + 0s_2 \qquad (3.12)$$

Subject to

$$6x_1 + 4x_2 \leq 24 \quad \Longrightarrow \quad 6x_1 + 4x_2 + s_1 \quad = 24 \qquad (3.13)$$

$$x_1 + 2x_2 \leq 6 \quad \Longrightarrow \quad x_1 + 2x_2 \quad + s_2 = 6 \qquad (3.14)$$

$$x_1, x_2 \geq 0 \quad \Longrightarrow \quad x_1, x_2, s_1, s_2 \geq 0 \qquad (3.15)$$

Before applying the Simplex method to solve such an LP, a preliminary step is to turn those inequality constraints into equality constraints by adding slack (or surplus) variables as shown above. The slack variables are used to represent the difference between the left and right hand sides of the inequality. Next, the objective function needs to be further rewritten in equation form as follows.

$$z - 5x_1 - 4x_2 - 0s_1 - 0s_2 = 0 \qquad (3.16)$$

Row	x_1	x_2	s_1	s_2	*RHS*
z	-5	-4	0	0	0
s_1	6	4	1	0	24
s_2	1	2	0	1	6

(3.17)

The starting Simplex Tableau (3.17) can be presented with coefficients and RHS of the LP problem specifying x_1 and x_2 as non-basic (zero) variables. The first row represents the objective function, while the next two rows are related to the two constraints for the problem. The basic variables are listed in the leftmost column with their values in the rightmost one. The objective function shows that the solution can be improved by increasing either x_1 or x_2. According to the Simplex method rules, x_1 has a higher negative coefficient in the modified objective function and thus it is the first entering variable to improve the objective value. Whereas, corresponding leaving variable can be determined by computing the positive ratio of *RHS* to row coefficients in the column of the entering variable. In this example, s_1 exhibits the smallest ratio and

is the leaving variable, whose row is referred to as the pivot row. Similarly, the column of the entering variable is called pivot column. The updated tableau (3.18) is presented below, with new basic variables x_1 and s_2.

Row	x_1	x_2	s_1	s_2	*RHS*
z	0	$-2/3$	$5/6$	0	20
x_1	1	$2/3$	$1/6$	0	4
s_2	0	$4/3$	$-1/6$	1	2

(3.18)

Now, the new objective value is 20, but not yet optimal, as the fact that there is still one variable (i.e., x_2) has a negative coefficient in the objective function. It makes x_2 the next entering variable because it has the possibility to further improve the objective function value. Then, repeat the above process to find corresponding leaving variable s_2. After the desired pivot operations, the final tableau is obtained as shown in (3.19). None of the coefficients associated with the non-basic variables in the objective row are negative. Thus, the result is deemed to be optimal with a value of 21.

Row	x_1	x_2	s_1	s_2	*RHS*
z	0	0	$3/4$	$1/2$	21
x_1	1	0	$1/4$	$-1/2$	3
x_2	0	1	$-1/8$	$3/4$	$3/2$

(3.19)

In practice, an LP problem may be much more complicated than the above case. Such a vertex-based approach gives way to several versions of the Simplex algorithm, including the Primal Simplex Algorithm, Dual Simplex Algorithm, and Primal-Dual Simplex Algorithm etc. Indeed, there may be several obstacles that need to be overcome

to achieve a solution. For instance, degeneracy, alternative optima, unbounded solutions or an infeasible solution [28]. In general, the Simplex method starts out at a basic feasible solution (extreme corner point), and makes an effort to move the objective value towards an optimal one by successively searching other feasible solutions in a systematic and iterative way.

3.1.2.3 Algorithms for Integer Linear Programming

ILPs are linear programs with some or all of the variables restricted to integer values, and even to only 0 or 1 in the case of BIPs. They are typically more difficult to solve than LPs due to the discrete characteristics of the values involved. The dominant method to solve integer (or binary) programs is known as the Branch and Bound Method (B&B), which is definitely efficient in terms of computation and a primary technique available in practice. An alternative approach is known as the Cutting Plane Algorithm, but this is not as popular as the B & B approach and may be slower to find the optimal as the size of the constraint set grows significantly as computations progress.

The first step of solving an ILP is to relax the solution space by removing the integrality constraints on variables and replacing any binary variable with the continuous range [0, 1]. This leads to a regular LP, whose solution provides an upper or lower bound on the objective value of the original ILP according to the optimization goal (i.e., maximization or minimization). If the optimal solution of an LP happens to satisfy all the integer or binary restrictions, it can be identified as the optimal solution of the ILP as well. Unfortunately, this is rarely the case. The typical strategy is to add special constraints to iteratively modify the continuous solution space in a manner that will eventually render an optimum integer solution. There are different ways to address the special constraints. To be specific, the B&B method is simply carried out through branching the solution space of variables and pruning the B&B tree based on the updated upper or lower bounds. Instead, in the Cutting Plane Method, it uses “cuts”

(actually inequalities) to reduce the solution space in such a way as to ensure that integer solutions are not excluded, this approach eventually produces an integer optimum result. The Branching and Cut method is a recent enhancement of the B&B method, which attempts to improve the quality of bounds during the computations. More details on the theory of linear and integer linear programming can be found in the references.

3.1.3 Quality of a Solution: Feasibility and Optimality

A solution of an LP is said to be *feasible* if it satisfies all the constraints, while the feasible solution that yields the best objective value is identified as *optimal*. Moreover, a significant superiority of the LP method over other heuristic approaches is that LP is capable of providing a lower bound on possible solutions as it progresses. This gives information about the quality of current solution obtained, which is defined as the difference between the current solution and the predicted optimal solution. Though LP models are designed to "optimize" a specific objective, they, especially those with large problem size, are not always solved to optimality due to various practical limitations that include CPU speed, memory size and a required solution time. The quality of the resulting solution depends on the completeness of the computation.

With respect to ILPs, a drawback of current ILP algorithms is their lack of consistency in solving integer problems. It is relatively easy to find a feasible solution for ILP problems but it is a time-consuming process to prove the current integer solution is indeed the optimal one. There is an important parameter called the *MIP gap* in integer programs. It is the minimum allowed difference between the best found solution and the optimal of the relaxed LP. The computation process will stop as soon as a feasible integer solution has been found and proven to be within the specified MIP gap. A solution within 1% of optimal is considered to be absolutely perfect; however, since most of real network design problems are quite large, solutions within a much bigger MIP gap, e.g., 50%, may still be acceptable. Other mechanisms to terminate ILP

optimization before reaching the optimal include setting limits on time, number of nodes, size of tree memory, and even the number of integer solutions [29].

3.2 Principles of Failure-Independent Path-Protecting *p*-Cycles

After the discovery of *p*-cycles, it was natural to ask if there was a counterpart of *p*-cycles that can provide end-to-end path protection. This question seems hard to be addressed especially with the expectation of maintaining the simple characteristics of *p*-cycles. As for path recovery, there are several issues that must be identified and explained before further discussing the path protection equivalent to *p*-cycles.

Mutual Capacity Problem

In span protection schemes, spans are regarded as a basic element to be protected. The objective is to find detour paths to replace the failed span, i.e., to find a set of paths between two adjacent nodes without crossing the failed span. It is described as a single commodity flow problem. In contrast, path recovery mechanisms deal directly with demand flows. They are inherently multi-commodity flow problems. Considering a span failure, it involves multiple demands that need to be rerouted simultaneously. Moreover, these affected demands compete for available link capacities and thus this leads to the so-called “mutual capacity” problem. Rerouting decisions for the affected demands are correlated and interfere with each other in a capacitated network. For instance, a span failure influences two demands. One O-D pair may have three different route choices with equal costs, while the other O-D pair only has one possible restoration route. There is a possibility that the route chosen by the first O-D pair uses up the capacity on spans which the only route of the second O-D pair must traverse. As a result, the demand between the second O-D pair cannot be restored. However, in fact choosing another route for the first O-D pair may not affect the feasibility of restoration of the second demands. From the capacity assignment perspective, it is equivalent to finding out

which pairs should be allocated the spare capacity of a span. In other words, how to decide the assignment of such "mutual capacity"? Thus, the coordination between routing choices is significant in path-oriented recovery. The mutual capacity issue must be addressed one way or another in acceptable and protection guaranteed path-protecting schemes.

Stub-Release Issue

In a path-oriented recovery scheme, another important concept is stub release. In a single span failure scenario, the affected working path does break, but the remaining upstream and downstream portions actually survive. An option in survivable networks, which is known as *stub release*, allows the working capacity on those surviving portions to be released and to be reused in restoration. It requires an acknowledgement of the exact failure location. Moreover, different failure scenarios produce different stub release capacities. As a result, stub release makes a recovery mechanism failure-specific. Note that the stub release issue does exist in span-oriented recovery schemes, as the capacity occupied by surviving portions is always reused from the end-to-end path view. Obviously, stub release brings in more available capacity for restoration purposes and produces higher capacity efficiency; however, this is at the expense of operational complexity. Two main considerations are pointed out in [9]: 1) the implementation of stub release requires the real-time dissemination of an accurate failure location, and 2) the reversion process after fault repair becomes complex. It makes sense to avoid stub release in protection schemes, in order to achieve failure independency, which is an important requirement for pre-connected recovery paths.

In conclusion, a *p*-cycle based path protection mechanism is expected to have the following desirable properties[21]:

- A single failure should be protected with a 100% guarantee for all demands, including link and node failures.

- The mechanism requires failure detection at the end nodes only, without specifying the exact failure location.
- Backup paths are fully pre-planned and pre-connected, leads to simple protection switching actions on the two end nodes of the path only.
- The network redundancy is far less than 100%, approaching the theoretical lower bound which comes from the optimal multi-commodity maximum-flow (MCMF) solution.

It is worth noting that the flow *p*-cycle, as the first path-oriented corresponding to span-protecting *p*-cycles, is not a protection scheme. Recall from Section 2.3 that it aims to protect the continuous flows of demand over path segments, which do not have to be the entire end-to-end paths. During the restoration of a particular segment, other surviving segments along the same path will be reused as before. Although its failure-specific characteristic does not require additional real time recovery actions, the implicit stub reuse does make the creation of end-to-end recovery paths impossible. In other words, backup paths cannot be end-to-end pre-connected prior to the failure.

3.2.1 The FIPP *p*-Cycles Concept

In 2005, a new *p*-cycle concept called Failure-Independent Path-Protecting (FIPP) *p*-cycles [21] was developed for WDM networks. It successfully extends the concept of *p*-cycles to the protection of end-to-end paths. The issues of mutual capacity and failure-specificity are addressed by introducing the idea of Shared Backup Path Protection (SBPP) into span-protecting *p*-cycles. FIPP *p*-cycles incorporate the properties of failure independency and path protection. Thus, the backup paths provided by FIPP *p*-cycles can be fully pre-connected without considering the exact failure location. As a protection scheme, FIPP *p*-cycles are much faster than those pre-planned but not pre-connected path-protection methods such as SBPP. This extension of *p*-cycles also maintains the characteristics of high efficiency and fast restoration speed. The FIPP *p*-cycle scheme has the same restoration speed as span-protecting *p*-cycles due to their

similar operational mechanisms. Still, only the two end nodes of the failed working path are responsible for failure detection and traffic switching. To sum up, FIPP *p*-cycles possess all the desired properties simultaneously that were mentioned before, which makes it expedient to serve as a path protection scheme based on *p*-cycles.

The key principle behind an FIPP *p*-cycle is summarised in [21]:

"Let the cycles act as p-cycles for end-to-end paths between nodes on the cycle, but only allow each cycle to provide protection relationships to a group of paths whose routes are all mutually disjoint."

To be specific, *p*-cycles can be extended to protect a set of end-to-end primary paths. A requirement is that the primary paths sharing a *p*-cycle should be disjoint from each other. Requiring resources to be disjoint is usually mentioned in the diverse routing or path protection context. It includes link disjoint and node disjoint properties. Considering link disjointness, at most one of the working paths will be out of service at a time in single link failure scenarios. The FIPP *p*-cycle will provide protection paths for that particular working path in this case. Mutual capacity here is referred to as the reserved capacity for FIPP *p*-cycles. No capacity competition will occur due to the fact that each FIPP *p*-cycle is used to protect just one working path under any circumstances. If node-protection is desired, the group of working paths is required to be fully link and node disjoint. Otherwise, the working paths that traverse the same nodes cannot be assuredly restored, because there will be a possibility that they could fail at the same time and compete for the capacity reserved on the FIPP *p*-cycle. Note that affected traffic cannot be restored when the end node of working paths fails. None of the network recovery schemes can avoid such traffic losses since it is the source or destination nodes of the traffic that fails. The capability of node protection in FIPP *p*-cycles depends on whether or not the working paths are node disjoint.

Figure 10 illustrates a set of working paths protected by an FIPP *p*-cycle. These working

paths are arranged to share the single FIPP *p*-cycle for their end-to-end protection.

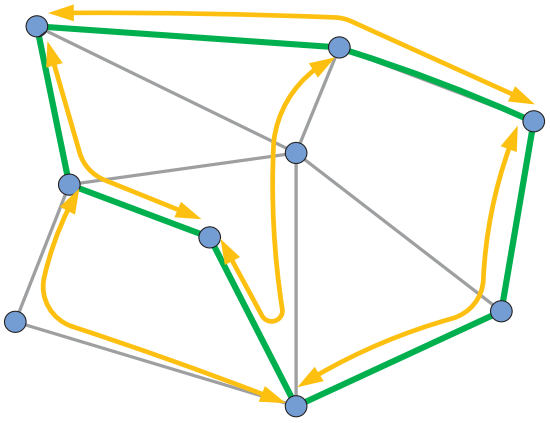


Figure 10 A set of working paths protected by an FIPP *p*-cycle

The five working paths are mutually disjoint from both a link and a node disjoint point of view. The FIPP *p*-cycle will provide protection for any single link failure, as well as for any single intermediate node (on path) failure. The mutual capacity problem is addressed by the restriction that working paths under a common *p*-cycle protection will be disjoint, thus they will not ask for restoration capacity simultaneously. Further, the mutual disjointness requirement leads to failure-independent end-to-end path protection. No failure localization is needed, and only the end nodes of the affected working paths do protection switching, which is pre-determined.

3.2.2 Relationships between Cycle and Working Path

A typical FIPP *p*-cycle network design involves a number of cycles. Each FIPP *p*-cycle has a protection relationship with one or more working paths. The cycle and corresponding path set form a configuration. More specifically, there are two main relationships between a working path and the cycle which can protect it. Note that the relationship between a cycle and a working path only exists when the two end nodes of the working path are both on the cycle. In other words, only on this premise, the cycle is capable of protecting the working path.

On-Cycle Relationship

On-cycle relationship is specified when a working path has at least one common link with the cycle. In this way, the cycle can provide at most one protection path for this working path. This relationship can be classified further into two cases. Imagine that a working path is completely along a cycle, that is to say, all links of the working path belong to the cycle simultaneously. The protection path is definitely the remaining part of the cycle in such a case. It is called a fully on-cycle relationship. The opposite case to this, the partially on-cycle relationship, occurs when at least one of the links on the working path is not shared by the cycle. The protection that the cycle can provide all depends on the specific position of the common links.

To illustrate, Figure 11a shows a fully on-cycle working path and its protecting cycle, while in Figure 11b illustrates how the cycle protects the affected working path in the

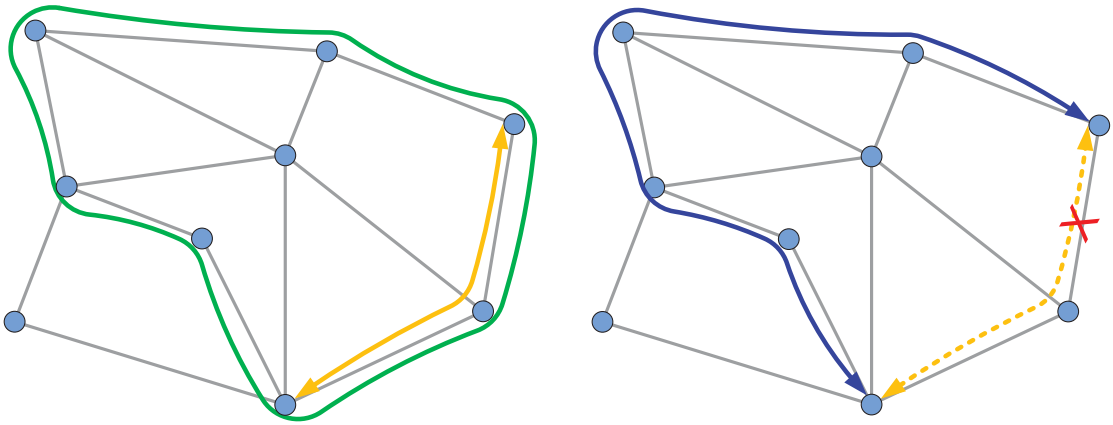


Figure 11 a) Fully on-cycle relationship b) The protection path provided by the cycle

case of a link failure. The involved traffic will be diverted along the cycle in the opposite direction. The arrowed solid curve represents the protection path supplied by the cycle in case the failure occurs along the dashed working path. Again, the traffic demand on the working path can be restored not only under link failure but also under intermediate node failure.

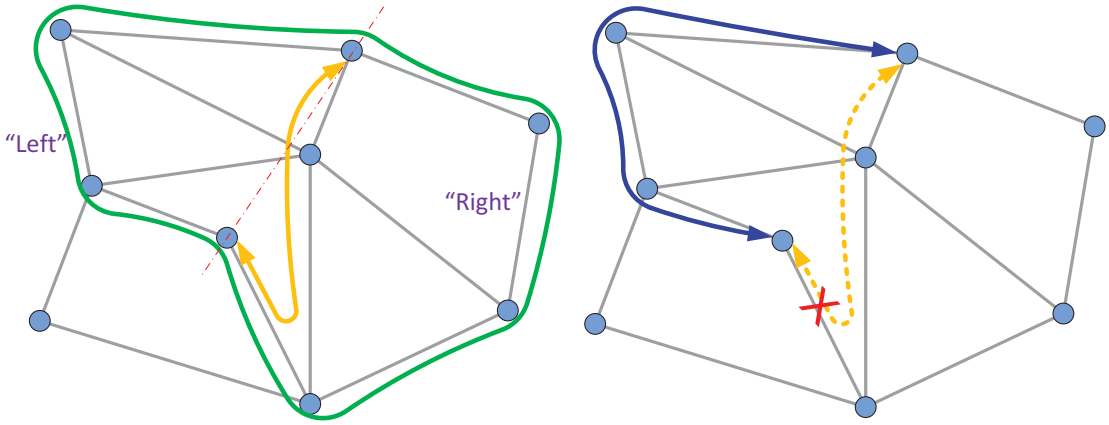


Figure 12 a) Partially on-cycle relationship b) The protection path provided by the left segment

Figure 12 is an example of a partial on-cycle case. With respect to the cycle, the two end nodes of the working path break the path into two segments, defined as left and right segments respectively. Common links are all distributed on one of the segments (right segment in this case) and leaves the other segment unoccupied. Thus, no matter which link or transit node fails, the left segment will always survive and become the protection path.

There is also another case that common links are distributed on both segments. The whole working path shapes like a letter "Z" on the cycle. Under a single element failure assumption, just one of the segments will be affected and the other one will remain intact and be utilized as in the fully on-cycle cases. The problem is, there is no way to know in advance which one will survive until an actual failure occurs. Attempts to include such cases will sacrifice the failure independency of the protection scheme. This situation depicted in Figure 13 has a vivid description in the literature: "z-case" or "z-form". The arrowed curve has common links with both the left and right segment. Thus, which segment can be the protection path depends on the specific failure location. In other words, the cycle cannot provide link-disjoint protection path to a "z-form" working path. Tradeoffs exist between additional capacity efficiency and complete failure independency.

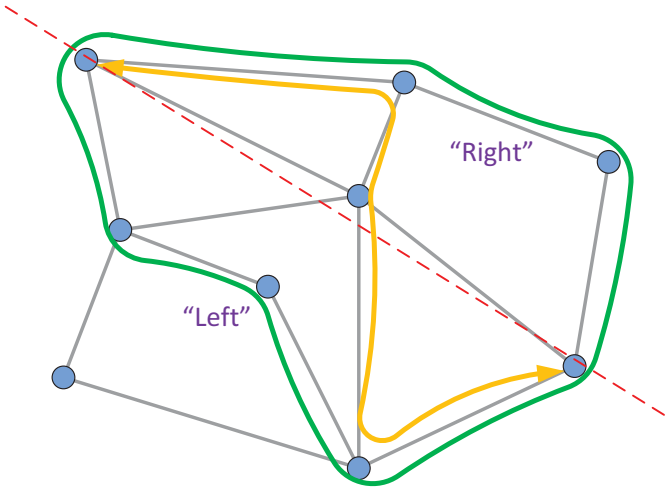


Figure 13 z-Case

Straddling Relationship

It becomes more straightforward with regard to straddling relationships. The working path has no common elements with the cycle except that its two end nodes are on-cycle. So the cycle and the working path are mutually disjoint (except for the two end nodes). It means that any failure on the working path cannot cause damage on the cycle. The two segments of the cycle both will survive and can be assigned to route the affected traffic. Therefore, the cycle can provide two protection paths simultaneously. Or from the capacity point of view, a single unit cycle can afford two unit capacities for traffic restoration. Figure 14a shows a full straddling working path and its corresponding protecting cycle. Figure 14b demonstrates how the two segments of the cycle turn out to be protection paths for the working path by simple protection switching at the two end nodes. The straddling relationship is one of the main advantages and contributes to the high capacity efficiency of *p*-cycles.

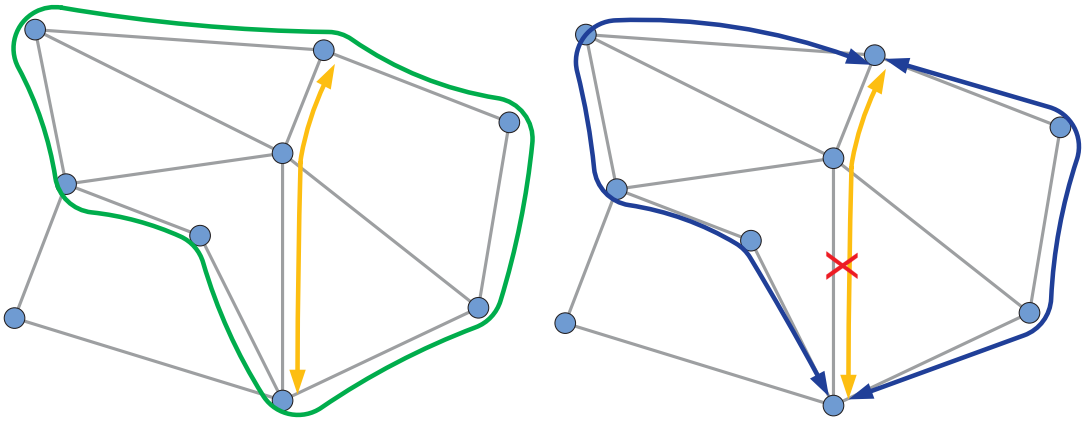


Figure 14 a) Fully straddling b) Two protection path provided by the cycle

The basic algorithm we used to determine the protection relationship between a cycle and a working path is as follows:

```
Protect_Cycle_Path_Ind(Cycle k, Path q) {
    If not both end nodes of path q on cycle k
        Return 0;
    If path q is disjoint from cycle k
        Return 2;
    If z-case should be avoided
        Return ((is_z_case(Cycle k, Path q)) ? 0 :
1);
    Else
        Return 1;
    }
```

```
is_z_case(Cycle k, Path q) {
    break cycle k into two segment L and R
according to the two end nodes of path q;
    For each link i that belongs to path q{
        If link i belongs to L
            Mark Left;
        Elseif link i belongs to R
            Mark Right;
        }
    If Left and Right are both marked
        Return true;
    Return false;
    }
```

Figure 15 Algorithms for Cycle-Path protection relationship

3.3 FIPP *p*-Cycle Network Design

3.3.1 Introduction

In the last section, we elaborated upon the concepts and properties of FIPP *p*-cycles. Since it is a fairly new recovery scheme, there are quite a few publications related to it, especially the papers that propose new models for the network design problem. In the FIPP *p*-cycles context, there are two main objects: *cycles* and *working paths*. Recall that the FIPP *p*-cycle mechanism requires working paths, which belong to a path set and are protected with a common FIPP *p*-cycle, and they must be mutually disjoint. The key issue for relevant network design is how to encode the mutual disjointness requirement. There are two main orientations to address this issue, first proposed in [21], which deal with the two objects (i.e., cycles and working paths) in the opposite sequence.

One principle considers the problem from an FIPP *p*-cycle point of view. In general, candidate cycles need to be clearly specified first, followed by the identification of a subset of working paths, which conform to the mutual disjointness requirement and can be protected with one of those predefined cycles in an efficient manner. The corresponding model is named the FIPP-SCP model in [21]. Instead, the other method originates from a different object. More specifically, it begins with constructing mutually disjoint path sets, which are referred to as Disjoint Route Sets (DRS) later in [22]. The next step is to specify a (simple) cycle which is across all end nodes of the working paths belonging to a particular DRS. The reason is that such a cycle is capable of protecting those working paths with certainty, regardless of the exact protection relationship. The earliest ILP model developed according to this idea was named FIPP-DRS and detailed in [22]. Several attempts based on the FIPP-DRS model can be found in literature, including a column generation based design [30] and joint capacity placement design [31]. The following sections present a mathematically detailed interpretation, as well as an understanding of the two basic models for FIPP *p*-cycle network design.

3.3.2 FIPP-SCP Model

In [21], an ILP model, known as FIPP-SCP, was proposed just following the invention of FIPP *p*-cycles. It is the first approach to address the FIPP *p*-cycle network design problem. The SCP notation was given since this model deals with spare capacity placement only. The principle is stated below:

"Given a cycle considered as a candidate FIPP *p-cycle, identify a subset of routes between end nodes that are on the cycle x that never contend at the same time for the restoration by the associated p-cycle."[21]*

Similar to the classic span-protecting *p*-cycle design, it requires enumerating either all cycles of a graph or a reduced number of only high-quality candidate cycles. Additionally, one working path should be specified to carry the traffic for each demand. Finally, a group of these pre-defined working paths will be identified under the protection of a common cycle, with the restriction of disjointness. The associated ILP model is given below.

FIPP-SCP Model:

Minimize:

$$\sum_{\forall j \in S} c_j \cdot s_j \tag{3.20}$$

Subject to:

$$\sum_{\forall k \in P} x_d^k \cdot n_d^k \geq h_d \qquad \forall d \in D \tag{3.21}$$

$$n^k \geq n_d^k \qquad \forall d \in D \tag{3.22}$$

$$s_j \geq \sum_{\forall k \in P} \pi_j^k \cdot n^k \qquad \forall j \in S \tag{3.23}$$

$$\alpha_d^k \geq \nabla \cdot n_d^k \qquad \forall d \in D, \forall k \in P \tag{3.24}$$

$$\alpha_d^k \leq \Delta \cdot n_d^k \qquad \forall d \in D, \forall k \in P \tag{3.25}$$

$$\partial_{mn} + \alpha_m^k + \alpha_n^k \leq 2 \qquad \forall m, n \in D | m \neq n, \forall k \in P \tag{3.26}$$

The objective is to minimize the total cost of spare capacity in the recovery scheme. Constraint (3.21) guarantees sufficient protection (100% or more) for the working path of a demand. By "sufficient protection", we mean the amount of traffic between this O-D pair is guaranteed to be restored by the current solution. Note that there is no restriction on the number of *p*-cycles used to protect a single demand. Constraint (3.22) ensures that the backup capacity assigned to cycle k is equal to or larger than that required by each path protection using cycle k. For each cycle, it may protect several working paths served for different demands. Due to the disjointness requirement, these working paths will never fail at the same time under the single failure assumption. Thus, the reserved capacity of a *p*-cycle will be allocated to protect a working path at a given time, and is equal to the maximum required spare capacity among those O-D pairs. Constraint (3.23) ensures that adequate backup capacity is reserved on each link. In other words, it guarantees that there is enough spare capacity to support the set of *p*-cycles used for restoration. Constraint (3.24) and (3.25) are typical equations to decide a binary variable by its continuous counterpart. To be specific, the binary choice α_d^k is decided by whether the cycle k is used to protect the working path of demand d. Constraint (3.24) requires α_d^k to be 1 if n_d^k is greater than 0 (i.e., cycle k is selected to protect the working path of demand d). The positive constant ∇ here must be small enough to make the right hand side of the equation remain in the interval [0,1]. Otherwise, if the right hand side becomes greater than 1, such a constraint will cause the solution to be infeasible. The philosophy of constraint (3.25) is similar. It forces α_d^k to be 0 when n_d^k is 0 (i.e., this cycle k will not be used to protect the working path of demand d regardless of its capability). Accordingly, the constant Δ must be large enough to guarantee the feasible solution space. The run-time decision of such a binary variable should be treated carefully. Constraint (3.26) explicitly imposes the set of working paths protected by the same *p*-cycle to be mutually disjoint. ∂_{mn}, α_n^k, and α_n^k cannot be 1 simultaneously to avoid non-disjoint working paths being protected by a common cycle k and inducing spare capacity contention subsequently. Two situations are allowed

by (3.26). If the working paths of demand m and n are mutually disjoint, the cycle k is permitted to provide protection to either of them or both; otherwise the cycle k can protect at most one of them.

3.3.3 FIPP-DRS Model

The FIPP-DRS model is the other basic method for FIPP p-cycle network design problems. As introduced, it sets out by constructing disjoint route sets. The only working path for every demand flow should be involved within the set of candidate DRS. Also, a number of eligible cycles needs to be figured out for the protection of a DRS. Typically, the simple cycle which unifies all the end nodes of those paths within the particular DRS is entitled to be a candidate cycle. Note that the on-cycle nodes are not restricted to be the end nodes of those paths only. But it is possible that no simple cycle exists in the network that covers all of the concerned nodes. In order to achieve full protection, sufficient spare capacity should be assigned for each FIPP p-cycle in the solution. This model copes with spare capacity placement as well. The objective to minimize the total cost of spare capacity is identical to that of the FIPP-SCP model. Constraints are given as follows:

Subject to:

$$\sum_{\forall c \in C_d} \sum_{\forall k \in P_c} x_d^k \cdot n_c^k \geq h_d \qquad \forall d \in D \tag{3.27}$$

$$n^k = \sum_{\forall c \in C} n_c^k \qquad \forall k \in P \tag{3.28}$$

$$s_j \geq \sum_{\forall k \in P} \pi_j^k \cdot n_k \qquad \forall j \in S \tag{3.29}$$

Constraint (3.27) guarantees sufficient protection for working demands between each O-D pair. It includes all eligible cycles that can protect a DRS which contains the working path of demand d. Constraint (3.28) ensures that the reserved capacity on each

p-cycle is equal to the total backup capacity that this *p*-cycle is required to provide for selected DRSs. It also can be explained from the demand's viewpoint. No matter how many DRSs the working path of demand *d* belongs to, each *p*-cycle can afford sufficient capacity to protect such a sub-set of DRSs. Equation (3.29) acts as a typical constraint in all types of *p*-cycle design. It ensures that adequate backup capacity is reserved on each link for the path protecting *p*-cycles in the final solution.

3.4 Further Studies on the Two Basic Models

So far, we have presented the mathematical ILP of FIPP-SCP and FIPP-DRS methods. This section provides a detailed illustration of these two models, without consideration of a performance comparison. The underlying ideas of these two strategies respectively reveal the two possible orientations of FIPP *p*-cycle network design. Indeed, though not identified clearly by the title as an FIPP-SCP model, FIPP-DRS is a true spare capacity placement method. Both the two models require only one working path to serve each demand.

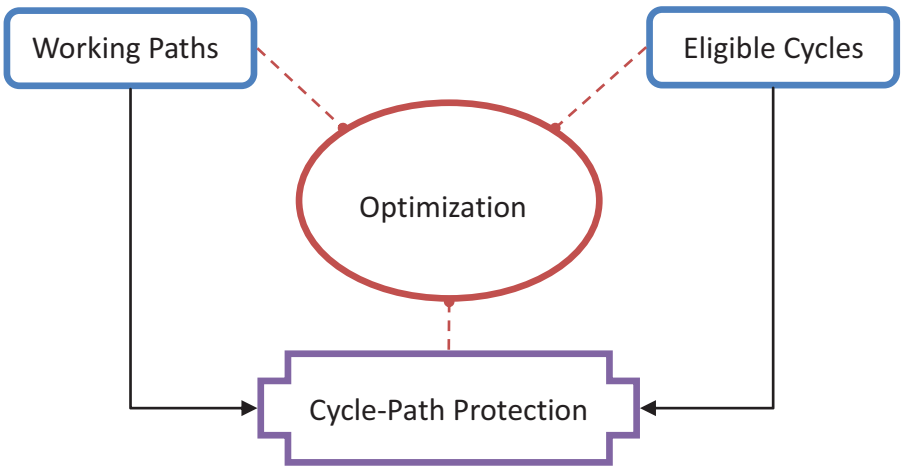


Figure 16 Process for FIPP-SCP model

The FIPP-SCP model merely focuses on two basic objects, i.e., cycles and working paths. Candidate cycles are required to be identified in advance through a cycle finding

algorithm. Also, the working path is pre-selected for each demand, by the shortest path first algorithm in most studies. As shown in Figure 16, not only the sets of candidate cycles and working paths, but also the protection relationship between them is the basic factor in the optimization process. The selection of a cycle to be an FIPP p-cycle is carried out on a per path basis. On the other hand, the FIPP-DRS method is more complicated due to the introduction of a new concept of disjoint route set, which leads to a total of three objects to be considered. The relevant objects and relations are summarized in Figure 17.

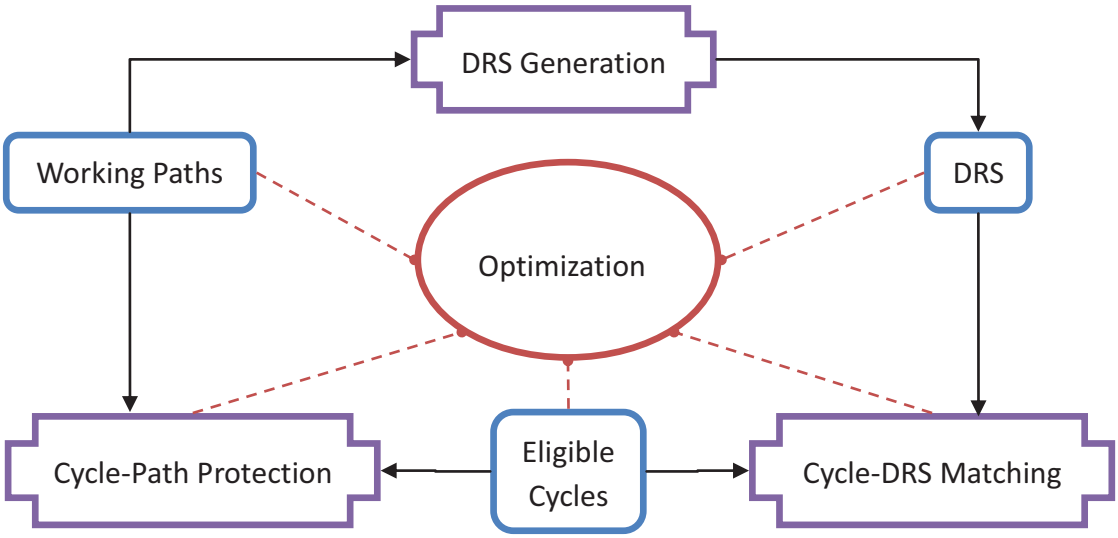


Figure 17 Process for FIPP-DRS model

This model works by generating working paths for every demand in the network and then selectively pooling these working paths into a number of DRSs. Obviously, enumerating all possible route sets will be exhausting and intractable, because the computation complexity approaches $O(n^2!)$ [22]. As a matter of fact, heuristic algorithms must be used to generate a set of eligible DRSs. Particular care needs to be taken to make sure that every working path appears in at least one of the resulting DRSs. In addition, a separate algorithm, denoted as "Cycle-DRS Matching" in the figure, is required to specify a number of cycles to protect each DRS. A typical way is to pick up eligible cycles from the pre-defined cycle set. A cycle is said to be eligible for a

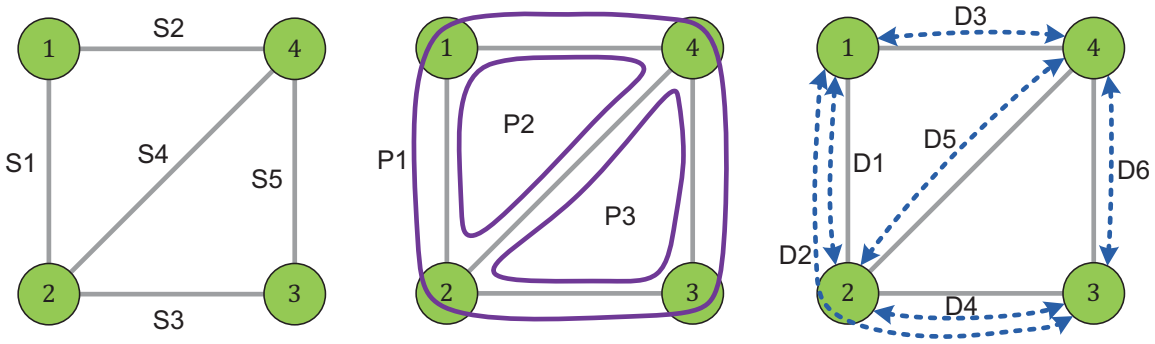
particular DRS, if it crosses every node that is an end-node to the paths within this concerned DRS. In brief, the FIPP-DRS method requires a more complex data collecting procedure that involves two more algorithms, one of which implements the generation of DRSs and has to be a heuristic method.

Strictly speaking, for FIPP-DRS, the decision of a cycle to be an FIPP *p*-cycle is based on the pre-generated path set, rather than the individual path. In other words, the combination of pre-defined DRSs, which is protected with a cycle, is explored during the optimization. On the contrary, such a set of working paths under the same cycle protection is completely formed by individual paths through the process of optimization for the FIPP-SCP method. Moreover, FIPP-SCP shows direct results on the number of unit-capacities for a cycle used to restore a particular working path, whereas FIPP-DRS yields those solutions for each DRS. Additional effort is required for each working path to figure out the exact amount of backup capacity available per cycle. The pre-defined DRSs, along with those additional but pre-solved algorithms related to it, contribute to the simpler ILP model relative to that of FIPP-SCP. Henceforth, with less constraints and variables, it takes less time for the optimization of FIPP-DRS to reach the optimal. This is in accordance with the discovery in [22] that the FIPP-DRS design were solved in seconds or minutes, whereas the runtime of the FIPP-SCP method could be a couple of hours, or even up to several days.

FIPP-SCP and FIPP-DRS are also different from the way they capacitate the cycles in the process. For FIPP-SCP, the path-based nature of modelling allows itself to adhere strictly to the mutual disjoint requirement. Consequently, every cycle is equipped with the maximum amount of spare capacity among those required for the working paths it protects. By contrast, there is no constraint imposed to restrict the working paths belonging to the group of DRSs protected by a common cycle, to be mutually disjoint, in the context of FIPP-DRS. In fact, both of the models were originally devised for WDM networks, where a unit capacity was identical to a telecommunication channel

over the optical fibre. Hence, the spare capacities on spans, as well as the number of unit-capacity of *p*-cycles were required to be integer-valued. Such a requirement can be relaxed when the MPLS environment is considered. In the context of optical networks, an FIPP *p*-cycle is defined to be a spare unit-capacity cycle. The term emphasises the number of units of capacity, rather than the cycle structure. Accordingly, the view is generally accepted that an FIPP *p*-cycle only provides protection to mutually disjoint working paths. On the other hand, when considering from the standpoint of the cycle structure, the behaviour of FIPP-DRS is no longer equivalent to that of FIPP-SCP. In the FIPP-SCP method, the set of working paths protected with the same cyclic pattern is mutually disjoint from each other due to the last constraint of the corresponding ILP model. However, path disjointness only counts with the algorithm that generates DRSs within the FIPP-DRS method. The resulting path set under the protection of a cyclical pattern, actually, is not limited to having only disjoint paths.

To better illustrate, let us consider a very small network with 4 nodes and 5 links.



a) Topology b) Cycles c) Working paths

Figure 18 An example network

Figure 18 depicts the network topology, all simple cycles and the working path set for a fully-meshed demand pattern. The amount of traffic for each demand is assumed to be 2 units of capacity. The cost of links is set to be the same. It turns out that with FIPP-SCP all of the three cycles are selected to be FIPP *p*-cycles. Two FIPP *p*-cycles for each cycle pattern are required to provide full protection, resulting in a total spare capacity of 20

units. The Table 2 shows the number of units of capacity for each demand that can be restored by the corresponding cycle pattern. Note that the working path for D5 is a complete straddling path over P1. So the two FIPP *p*-cycles using P1 structure can provide 4 units of capacity to restore the traffic of D5.

Table 2 Solution for the FIPP-SCP model

	D1	D2	D3	D4	D5	D6
P1		2	2		4	2
P2	2		2		2	
P3				2	2	2

By contrast, only P1 turns out to be the final FIPP *p*-cycle structure, when the problem is solved with FIPP-DRS. The candidate set of DRSs is presented in the Table 3.

Table 3 The pre-selected DRSs for the problem

DRS	Demands	DRS	Demands
1	D1 D3 D4 D5 D6	9	D3 D6
2	D1 D4 D5	10	D3
3	D1	11	D1 D4 D6
4	D2 D3 D5 D6	12	D4
5	D2 D5 D6	13	D5
6	D2 D3	14	D3 D4 D6
7	D2	15	D2 D6
8	D3 D4 D5	16	D6

There are four FIPP *p*-cycles in the solution, serving four DRSs (shaded in Table 3) respectively. All of them conform to the P1 structure. Again, here every FIPP *p*-cycle can protect 2 units of capacity for D5, rather than one for the others. Details are given in

Table 4.

Table 4 Solution for the FIPP-DRS model

P1	DRS	D1	D2	D3	D4	D5	D6
1	1	1		1	1	2	1
2	2	1			1	2	
3	6		1	1			
4	15		1				1

The total spare capacity is only 16 units, less than that of FIPP-SCP. An observation of results with FIPP-DRS shows that an FIPP *p*-cycle is really used to protect mutually disjoint working paths that belong to a pre-defined DRS. Henceforth; there is no contention for the capacity available on an FIPP *p*-cycle, irrespective of the exact failure scenario. However, it should be noticed that the same cyclic structure P1 is shared by D1 and D2, the working path of which cross the same link between node1 and node2. The solution of FIPP-DRS proves that this method allows non-disjoint working paths to be protected by an identical cyclic structure. The theoretical underpinning for this statement lies in the way of spare capacity assignment for potential FIPP *p*-cycles. The total number of FIPP *p*-cycles (with the same cyclic structure) is the sum of the amount required for each selected DRS individually. Again, for a group of DRSs sharing the protection of a common cyclical pattern, there is no constraint to guarantee that their members, i.e., the working paths that belong to them, are mutually disjoint.

3.5 Summary

At the beginning of this chapter, the foundation for the linear programming was introduced, along with a brief description of the various solution approaches, including those for integer linear programs. Next, the concept and characteristics of FIPP *p*-cycles

were detailed in Section 3.2. The two basic relationships between a cycle and a path were defined and discussed according to several different situations. Section 3.3 introduced the two main methods for FIPP *p*-cycle based network design problems: known as FIPP-SCP and FIPP-DRS, with the corresponding ILPs.

A comprehensive comparison of the two basic models was carried out in the theoretical sense, accompanied by a small but representative example. For FIPP-DRS, in general, there are special steps required to generate a disjoint path set in advance and paths are somewhat arranged into groups before entering into the optimization process. Contrary to the case for FIPP-DRS, the FIPP-SCP method regards paths as individual components. Consequently, large problems for FIPP-SCP are unlikely to be solved to optimality within the same time period as for FIPP-DRS. On the other hand, the fact that FIPP-DRS outperforms FIPP-SCP in terms of capacity efficiency can be attributed to the relaxed disjointness requirement that allows non-disjoint paths to be protect by the same cyclical structure.

Chapter 4 Optimal Design for MPLS Network Capacity

4.1 Introduction

The motivations to introduce recovery mechanisms into the logical layer, such as IP, ATM, MPLS etc., can be attributed primarily to the limitations and drawbacks of recovery on the physical layer, as discussed at the beginning of this thesis. It is generally agreed that the earliest span-protecting *p*-cycle is, theoretically, a recovery mechanism which can be extended to all types of networks. One of the promising types of *p*-cycles, known as FIPP *p*-cycles was also discussed in the previous chapter. The present chapter provides an attempt to extend FIPP *p*-cycles concept from optical networks to MPLS networks through a jointly optimized network design. This chapter is organized as follows: First, the necessary adaptations, as well as some specific design considerations are introduced in detail. Next, a brief overview of the advantages of FIPP *p*-cycles over other MPLS failure recovery techniques [3] is presented, including a scalability comparison. This is followed by a detailed introduction of the joint capacity design. The main contribution of this chapter is the extension of the basic FIPP-SCP model from a spare capacity placement paradigm, to a jointly optimized network design.

4.2 Extend FIPP *p*-Cycles for Resilient MPLS Network Design

4.2.1 *p*-Cycle Based Recovery for MPLS Layer

Though *p*-cycles were devised as a general concept and can be utilized for other layers in a similar logical way as they would be originally used in optical networks, adaptations are still inevitably required to fit them into IP or MPLS environments. In

[3] See Section 2.2.

this section, we concentrate on the corresponding impact on network design and postpone the deployment related issues to Chapter 6. Since the FIPP *p*-cycle is a fairly new protection mechanism, yet to date, nearly all of the research for FIPP *p*-cycles were based on optical networks. Fortunately, there are still some guidelines to follow, which are borrowed from corresponding span-protecting *p*-cycle network designs.

In optical networks, the installed capacity is configured with individual optical channels and divided into working capacity and spare capacity respectively. An optical channel is allocated to carry demand volumes in the normal or a failure state. In other words, a unit of capacity is classified with certainty as either working capacity or spare capacity. As a result, the variables related to the amount of capacity, including the ones representing the number of units of capacity for each *p*-cycle, are forced to be integer in the optical network context. On the other hand, demand volumes are carried through end-to-end paths/tunnels and typically given in Megabits per seconds (Mbps) in MPLS networks. Hence, the tunnel capacity does not need to be necessarily in integral units of Mbps and we can model the capacity related variables as continuous.

A new concept of virtual *p*-cycles is introduced within an IP environment, for the reason that there is no identifiable spare capacity which can be designated and reserved for restoration [19, 32]. A virtual *p*-cycle is defined as a pre-configured circuit-like cyclic structure, and bandwidth is reserved for such a logical construct to protect corresponding resources. Remember that it tends to regard a *p*-cycle as a single unit of capacity (i.e., an optical channel) in optical network designs.

The first attempt to extend span-protecting *p*-cycles to logical layers is carried out in a pure IP network [19, 32], yet it is more desirable to employ *p*-cycles in an IP network with tunnelling techniques. Since there is no distinction between working and spare capacity, it does not make sense to address the resilient network design from an explicit

cost of spare capacity point of view. Instead, a design that aims to minimize the congestion level during restoration is more acceptable under these circumstances. When applied to MPLS, the situation becomes quite different due to the application of the tunnelling mechanism. MPLS networks provide transport services through end-to-end tunnels for each demand node pair (O-D pairs for short). Note that these tunnels are also referred to as primary tunnels and set up as label switched paths. Meanwhile, backup tunnels coexist for survivability consideration. Unlike in pure IP networks where there is no means to separate the working and backup capacity, MPLS does have the capability to distinguish them by defining working and backup tunnels and designating corresponding capacity for each of them. The exploitation of the exact cost of spare or total capacity is allowed to be carried out. It is more beneficial to apply *p*-cycle based protection schemes for MPLS networks than for optical networks to some extent. MPLS functions over both the forwarding and control plane. It fundamentally is a Call Admission Control (CAC) mechanism because reservations are carried out on the control plane and take effect on the data plane only when traffic occurs. Therefore, it is instinctive to utilize the reserved but idle capacity, whereas in SONET or WDM networks, special operations are required in order to make use of idle spare capacity.

In addition, as already remarked in [19, 32], MPLS is a favourable technical means to realize *p*-cycles in an IP environment. Virtual *p*-cycles have a similar characteristic to LSPs in that allocated capacity will not be occupied until traffic is required to be carried. Traffic will be taken by LSPs when it truly happens or go through the *p*-cycles in the failure state. Studies of [33] and [34] are both involved with bandwidth guaranteed protection in MPLS networks with *p*-cycles and hence are based on the same model with the objective of minimizing the total cost of spare capacity. The model is a modification of the one in [19, 32] which aims to minimize the congestion level over links during restoration. Redundancy (also called Backup Capacity Ratio in some cases) is well below 100% for most of the test networks and goes as low as 70% for particular

networks in [33]. Note that the candidate cycle size was restricted from 3 to 9 in the test. Again, those studies demonstrate the high capacity efficiency of *p*-cycles.

FIPP *p*-cycles is a failure independent path protection recovery scheme and extends the classic span-protecting *p*-cycles to the scope of path protection. Two types of virtual *p*-cycles are combined to provide protection to the IP layer in [19, 32]. Span-protecting *p*-cycles are adapted to protect links, while node-encircling *p*-cycles are employed to cope with node failures. By contrast, the FIPP *p*-cycles themselves are sufficient to provide equivalent link and node protection simultaneously, due to the nature of path-orientation. Moreover, all the LSPs across a link are protected as a single entity by the bypass tunnel (i.e., the corresponding span protecting *p*-cycle) in the first combined *p*-cycle mechanism. This leads to a relatively coarse protection granularity, especially in terms of guaranteed QoS. That is to say, it is hard to implement LS*P*-based recovery. Conversely, the FIPP *p*-cycles mechanism exhibits a much finer protection granularity, and has the potential to offer different classes of recovery on an LSP basis, according to the type or priority of the relevant traffic. Last, but not least, path protection schemes are more cost efficient than link protection schemes. Results in [30] indicate that FIPP *p*-cycles yield less extra cost than link-protecting *p*-cycles for all test cases, with the biggest gap being 17.8%. The most efficient FIPP *p*-cycle design produces only an extra 49.8% cost of protection. The path protection characteristic of FIPP *p*-cycles results in higher capacity efficiency and makes the protection QoS-enabled. There are adequate grounds for the expectation of FIPP *p*-cycles to be a well performed protection scheme for MPLS networks.

4.2.2 The Selection of Basic ILP Models

Given the conclusion of our studies on the two basic models of FIPP *p*-cycles in the previous chapter, the FIPP DRS method shows higher capacity efficiency for the reason that it breaks the strict disjointness requirement. In fact, it allows a cycle (here we

concentrate on the structure per se) to protect non-disjoint working paths as long as there are enough copies of FIPP *p*-cycles that conform to this cycle structure. Note that in optical networks, the capacity is related to the number of optical channels and an FIPP *p*-cycle is constructed with a single spare channel on the underlying spans. Thus, it is possible to implement the FIPP *p*-cycles in such a manner. For non-disjoint path protection by a common cycle structure, it is straightforward to reserve a sufficient number of *p*-cycles which have an identical cyclic route.

On the other hand, the concept of virtual *p*-cycles is added to represent the cyclic structure in an MPLS environment, with the aid of which the prior statement can be updated. For MPLS networks, the capacity is described in terms of bandwidth. If non-disjoint working paths are under the protection of a common virtual *p*-cycle, the spare capacity designated to the particular virtual *p*-cycle will be the sum of backup bandwidth required by each of them.

As will be discussed in Chapter 6, each virtual FIPP *p*-cycle is pre-configured as two unidirectional cycle LSPs, one for each direction. The required backup bandwidth will be reserved for each cycle LSP. However, relaxing the disjointness constraints as the way in the FIPP DRS method leads to the fact that a virtual FIPP *p*-cycle protects several non-disjoint working paths. Suppose a virtual FIPP *p*-cycle is still implemented in the desired way with two unidirectional cycle LSPs, the contention for the same backup bandwidth on the corresponding cycle LSP, becomes unavoidable between those non-disjoint working paths. Of course, this problem can be addressed through creating multiple cycle LSPs per cyclic structure, instead of a single cycle LSP for each direction. More specifically, multiple cycle LSPs is used to protect the non-disjoint LSPs that are under the protection of the same cyclical pattern, respectively. In other words, a cycle LSP has to be duplicated and assigned with appropriate bandwidth for the purpose of protecting a particular working LSP. However, the total number of cycle LSPs (i.e.,

backup tunnels) will increase and cannot be determined only by the number of distinct cycles that appeared in the optimal solution anymore. Instead, the decisions concerning which cycle is chosen to protect which working paths should be examined carefully to gather the total number of cycle LSPs required, as well as the exact amount of backup capacity needed per cycle LSP. To sum up, the application of FIPP-DRS makes the implementation more complicated in MPLS networks. Instead, the FIPP SCP model which has a strict requirement on disjointness of working paths is more preferable to employ FIPP *p*-cycles in MPLS environment.

4.2.3 z-Case

The use of z-case[4] eliminates the desired property of failure independence for FIPP *p*-cycles. The reason for this is that when z-cases are taken into account, a decision between a pair of cycle LSPs has to be made after failure occurs, before which nothing can be done to restore the traffic. Henceforth, the z-case must be avoided absolutely in our design, in order to keep the failure independency of FIPP *p*-cycles. This can be achieved through a function called ***Protect_Cycle_Path_Ind*** in our C++ project, which is capable of avoiding z-case as indicated.

4.3 FIPP *p*-Cycles as an MPLS-Based Recovery Method

4.3.1 Comparison with Other MPLS Failure Recovery Mechanisms

The outstanding properties of FIPP *p*-cycles, especially their end-to-end and path-oriented nature, makes it possible for them to be a very attractive recovery scheme for MPLS networks. Moreover, node protection is an intrinsic property of FIPP *p*-cycles and bandwidth protection per demand can be achieved with certainty.

Contrary to local protection techniques (e.g., facility backup and one-to-one backup),

[4] A special protection relationship between a path and a cycle, see Section 3.2.2.

FIPP *p*-cycles is an end-to-end path protection scheme. No effort is needed to build backup LSPs separately for link protection and node protection. It is inherently capable of protecting node-failures. On the other hand, finer protection granularity can be obtained through FIPP *p*-cycles. It is more straightforward to maintain the QoS per LSP than in local recovery cases. For example, a quandary will arise for the facility backup scheme, when deciding the amount of bandwidth that must be reserved for the bypass tunnels. It is apparently not cost efficient to allocate the sum of the bandwidth required by each LSP to the corresponding bypass tunnel. Also, it is computationally complex to assign a reasonable bandwidth that can guarantee full protection for all LSPs in cases of failure. As for one-to-one backup, although the reserved bandwidth on backup tunnels can be determined directly by the bandwidth of primary tunnels, it is difficult to perform bandwidth sharing. The overall performance of capacity efficiency is low, even if taking specific treatments such as merging operations. The FIPP *p*-cycles scheme gets around such a dilemma by regarding a mutually disjoint path set as the protection element and allowing the backup bandwidth to be shared among those path members within the set. Therefore, as a path protection scheme, an FIPP *p*-cycle is more suitable for node-protection and bandwidth-protection desired resilient networks.

FIPP *p*-cycles can be classified as a global path protection method. The classic 1:1 path recovery method is often criticized owing to the double-booking of resources and the nondeterministic switchover delay as well as unnecessary link protections[10]. The reason for low resource utilization is that no backup bandwidth sharing is employed in the scheme. By contrast, in FIPP *p*-cycles, backup paths are explored in a way to share the entire cyclically pre-connected protection structure. The necessary backup capacity can be reduced accordingly. The 1+1 path protection method is extraordinarily redundant with a redundancy of more than 140% for all test networks, and it even exceeds 200% in some sparse networks [35], whereas results in [30] reveal that networks based on FIPP *p*-cycles have a large potential to achieve redundancies of less

than 100%. It was our original intention to explore the possibility of employing FIPP *p*-cycles in MPLS networks. It is a capacity-efficient failure-independent end-to-end path protection scheme and all backup paths can be pre-established.

4.3.2 Scalability Analysis

So far we have been focusing on the advantages of FIPP *p*-cycles in terms of its intrinsic properties. A general discussion about scalability issues could help to understand the results of the following tests. Scalability is an undoubtedly major issue in all types of networks regardless of which layer it serves. With respect to MPLS networks, it deserves particular consideration. There are two main reasons for this. One is due to the capability of an LSR. The router itself has a limit on the number of LSPs it supports. The other, from the operational aspect, is that a larger number of LSPs in the network is cumbersome to manage and renders the trouble shooting task much more complicated.

Regardless of whether the backup paths are pre-established or dynamically signalled, they do contribute to the overall number of LSPs that are created and maintained in the network. A larger amount of LSPs definitely places a heavy burden on the RSVP signalling. The impact can be characterized in three aspects [10]. First, RSVP is a soft state protocol. That means when two LSRs exchange information with RSVP, they do not assume that the information is stored permanently. Rather, the information is required to be refreshed periodically by resending. Mechanisms, such as decreasing the refresh interval or refreshing a larger number of LSPs by a common message at a time, will assist to reduce such an impact. Secondly, memory is consumed to deal with these RSVP messages and hold the RSVP state for each LSP on a router. Finally there is an issue relating to the recovery time in local protection schemes. It is affected by the number of LSPs needed to reroute the traffic on a particular mid-point LSR. Working LSPs are assumed to be the same in different recovery schemes. Hence, scalability is considered in terms of the number of backup tunnels required.

The following notations and discussions are based on the research published in [10]. Observe that the notation in this section differs from that used elsewhere in the thesis, and does not match that of Section 1.4.

Notation:

N Total number of nodes (LSRs)

L Total number of links (unidirectional)

D Network diameter. To be specific, it is the average number of hops required for primary LSPs.

C Nodal degree, represents the average number of neighbours. It is an important metric to reflect connectivity in all types of networks.

$$C = L/N$$

W Total number of primary LSPs in the MPLS network

B Number of backup tunnels required.

P Total number of distinct bi-directional *p*-cycles used in the recovery scheme.

M The number of meshes in the network. There may be multiple meshes of working LSPs in a network serving different traffic types. For example, one mesh for the voice traffic and one mesh for the data traffic. As a consequence, each of them requires corresponding backup tunnels.

S Average number of splits for protection purposes. In cases where bandwidth is a scarce resource, more than one backup tunnel may be required to guarantee the protection bandwidth. That is to say, in 1+1 path protection, each primary LSP has *S* end-to-end backup paths while with respect to FIPP *p*-cycles, the same number of *p*-cycles is allowed to protect a primary LSP[5]. Similarly, in a local protection context, it can be interpreted as there are *S* bypass tunnels for each protected resource, i.e., links and nodes.

[5] Note that there is no restriction on the number of p-cycles used to protect a single path in the original FIPP p-cycle models and our work, in order to enhance the capacity efficiency.

The following assumptions are made to simplify the discussion.

- It is assumed that primary LSPs are deployed with a full mesh connection.
- No differential classes of recovery are employed.
- All flows are non-bifurcated under the normal operating state.

Consider the number of backup tunnels in different cases:

1. **Global path protection (1:1)**

   The number of backup paths is given as:

$$B_G = S \times W = S \times M \times N \times (N-1)$$

2. **Local protection:**

   - ***Facility backup***

     The calculation includes backup tunnels for both link and node protection.

$$B_F = M \times S \times L + M \times S \times N \times C \times (C-1)$$

     The first part of the right hand side represents the number of backup tunnels need for link protection, while the second part stands for node protection. Substituting $L = N \times C$ into the equation produces a simplified form.

$$B_F = S \times M \times N \times C^2$$

   - ***One-to-one backup***

     The number of backup paths is proportional to the number of primary LSPs and can be described as follows by the absence of merging. It proves to be D times that in the global 1:1 path protection case.

$$B_O = S \times W \times D = S \times M \times N \times (N-1) \times D = D \times B_G$$

3. FIPP ***p*-cycles:**

   There are two scenarios to deploy FIPP *p*-cycles in an MPLS network, depending on what is regarded as a backup path. Detailed discussions about these two scenarios can be found in Chapter 6. Here, we just focus on scalability related issues.

- ***Scenario 1 using actual protection paths as backup tunnels***

  It is intuitive to consider actual protection paths to be backup tunnels, just like the way other path protection methods do. By a protection path, we mean the surviving portion of a protecting *p*-cycle to where the affected traffic will be rerouted. Thus, the number of backup paths is determined by the number of primary LSPs and the number of *p*-cycles used to protect each of them. Another factor that should be taken into consideration is the protection relationship between a working LSP and its protecting *p*-cycle. The on-cycle relationship produces at most one protection path while the straddling relationship affords two protection paths. Since the straddling relationship brings higher capacity efficiency, models that aim to minimize the total cost of spare capacity will prefer to explore the straddling relationship. It makes sense to expect as many straddling relationships as possible in the optimal solution. Hence, in the worst case, but for the sake of reducing spare capacity, the number of backup paths can be formulated as

  $$B_{p1} = 2 \times S \times W = 2 \times S \times M \times N \times (N-1) = 2 \times B_G$$

  Recall that S is referred to as the number of *p*-cycles used to protect one primary LSP in an FIPP *p*-cycle context. The number of backup tunnels will be double that in the global 1:1 path protection method under the same split degree.

- ***Scenario 2 using p-cycles as backup tunnels***

  Instead, constructing *p*-cycles as backup tunnels is another choice. Then, the number of backup tunnels is independent of the number of primary LSPs, and can be decided by the number of distinct *p*-cycles in the optimal solution. According to the unidirectional property of LSP, a bidirectional *p*-cycle should be composed of two unidirectional LSPs, one for each direction. Thus, the number of backup tunnels turns out to be

  $$B_{p2} = 2 \times P$$

  It is hard to predict and compare the number of backup tunnels with other cases

above, since it is computed from the cycle point of view. Still, there is evidence to show the superiority of this scenario. Recent work in [36] shows that the number of selected *p*-cycles never exceeds 45 for a 15-node network. Alternatively, a transformed way can be obtained by introducing a sharing factor ϑ which encodes the average number of primary LSPs shares protection of an FIPP *p*-cycle. We arrive at another formulation:

$$B_{p2} = 2 \times \frac{W}{\vartheta} = 2 \times M \times N \times (N-1) \times \frac{1}{\vartheta}$$

For comparison purposes, a set of parameters is given in advance and listed within the graph. At first, suppose:

$$S = 2, \qquad M = 2, \qquad D = 5, \qquad C = 4, \qquad \vartheta = 3$$

Figure 19 clearly shows the number of backup tunnels over the number of nodes in the network for the above cases. One-to-one backup, global 1:1 path protection and FIPP *p*-cycles scenario 1 show poor scaling capability. The results in the facility case and FIPP *p*-cycles scenario 2 are quite similar. Consequently, scenario 2 is more suitable to be carried out with respect to FIPP *p*-cycles implementation in an MPLS environment.

To better illustrate the impact of the sharing factor in the FIPP *p*-cycles scenario 2, the number of backup tunnels is presented with different ϑ, as depicted in Figure 20. Local facility protection with the presumed parameters is used as a benchmark. Obviously, the restriction that no sharing within a *p*-cycle (i.e., $\vartheta = 1$) gives rise to the largest number of backup tunnels. As a matter of fact, if an FIPP *p*-cycle is only allowed to protect at most one LSP, the number of backup tunnels will be the same as that of global 1:1 path protection under our assumptions. Indeed, models aiming to minimize the spare capacity will force the sharing factor to be larger than 1, if possible. In a network with nodes less than 50, FIPP *p*-cycles scenario 2 exhibits attractive capability in terms of scalability, especially in cases when the sharing factor is over 3.

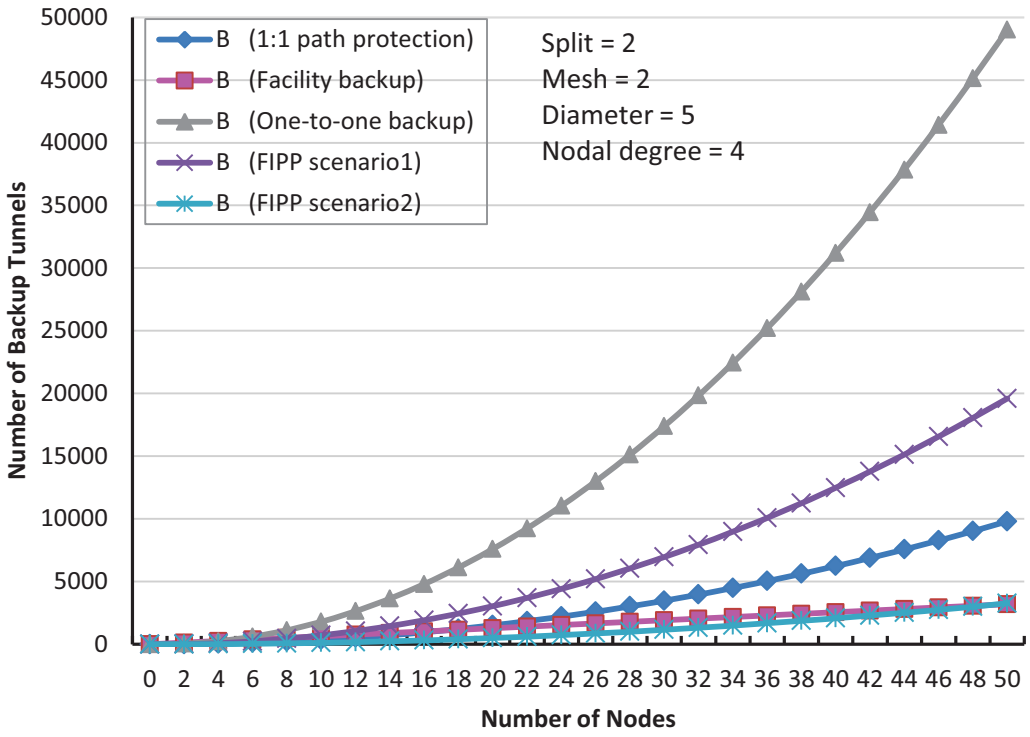


Figure 19 Comparison of required backup tunnels for global 1:1 path protection, facility backup, one-to-one backup and FIPP *p*-cycles

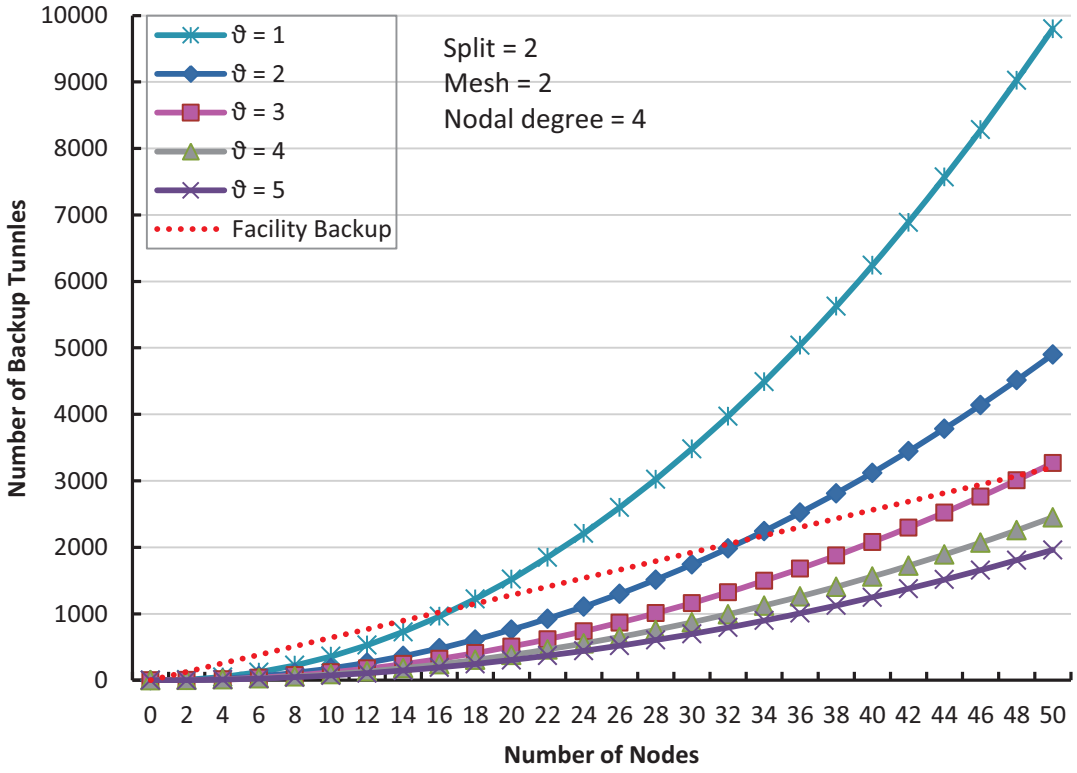


Figure 20 The number of backup paths over different sharing factors in FIPP *p*-cycles scenario 2 (Using facility backup case as benchmark)

In conclusion, it is more beneficial to realize FIPP *p*-cycles as backup tunnels for protecting several primary LSPs from the scalability aspect. Additionally, FIPP *p*-cycles are more scalable than current global and local protection mechanisms in networks of reasonable size.

4.4 Joint Optimization

Both the initial designs for FIPP *p*-cycles, i.e., FIPP-SCP and FIPP-DRS models are based on the same approach, namely Spare Capacity Allocation (SCA). In SCA, working paths are resolved in advance and without regard to the survivability design problem. Typically, the working capacity assigned to each link would arise from the shortest path routing or minimum hop algorithm. Then, spare capacity is placed with the goal of minimizing its total cost while still guaranteeing 100% restorability against a single failure. On the other hand, a joint capacity allocation (JCA) design involves concurrent decisions of both the working capacity and spare capacity, and thus typically aims to minimize the total cost of required capacity. To be specific, the main information required includes the network topology and the demand matrix in the case of a JCA design. These are inputs to the whole problem and responsible for determining the working paths, selected protection structures as well as providing elaborate capacity placement, for working and spare capacity respectively. During the optimization, working paths can be picked up from a pre-generated set of path alternatives for different demand pairs, or even shaped from scratch (e.g., by introducing flow reservation constraints). The resulting routing and corresponding working capacity placement have an effect upon the subsequent survivability design, whereas in SCA the routing of working demands is pre-determined and independent of the succeeding reliability design.

Indeed, JCA design gives the solver wider opportunities to further reduce the total

capacity required by adapting working paths. It allows the working paths to be deviate from those strict shortest paths used in parallel SCA problems. Of course, as a result, the investment on working capacity is somewhat more costly. Nevertheless, the total cost responds to both working and spare capacity and will not be worse than that of the SCA method, provided that the problem is optimally solved. It is justified to expect a greater reduction in the cost of spare capacity and further a resilient network design with a lower total cost. There is evidence of the significant benefits brought in by joint optimization. Joint design in span-protecting *p*-cycle networks [12], was found to yield a reduction of up to 25% in the total capacity relative to the optimal non-joint counterpart. Essentially, observations show that the consequent working paths do not dramatically deviate from the shortest one, in terms of path length. It is nearly equal shortest paths, rather than significantly longer paths, that lead to the improved efficiency. An explanation, given in [9] (pp. 314-316), is that joint optimization takes advantage of the load-level effects over almost equal shortest paths and is unlikely to involve paths with a wild departure from the shortest ones. JCA methods exploit how the choices of working paths interact co-ordinately with the spare capacity allocation to maximize the overall capacity efficiency. The above considerations motivate the adaptation of FIPP-SCP model into MPLS networks and an appropriate extension of the model to JCA network design.

4.5 MFIPP-JCA Models

4.5.1 Principles and Assumptions

According to the considerations in the above sections, we propose a joint optimization network design extended from the classic FIPP-SCP method and adapted into the MPLS environment. During our first attempt to formulate such a problem, two general questions perplex us. Is there only one primary LSP for a particular O-D pair? There may be multiple meshes of LSPs in a network serving different purposes at the same

time, usually one mesh for real-time applications, e.g., video, voice traffic, and one mesh for ordinary data traffic. In our studies, we intend to investigate networks without differential services, which is equal to dealing with target networks that only support a single service type. Still, there is a technique that can incur multiple LSPs between a pair of nodes. The characteristic of load balancing is precisely to forward the traffic from a source to a destination across multiple paths. As an extra capability, conventional protocols, like OSPF or IS-IS, support equal load balancing. Load balancing over equal cost paths is comprehensible for the reason that it can achieve less link load without extra cost. With regard to MPLS, both equal and unequal load balancing can be performed, no matter whether the multiple paths have identical cost or not, especially for traffic engineering purposes. It can be concluded that the ability to route a demand over several paths simultaneously, is desired and might be required necessarily in MPLS-TE (MPLS Traffic Engineering) networks. Even so, if the network is provisioned with far more than enough capacity, a single working LSP for each demand would be more beneficial from an operational prospect. Such a contradiction exists in all types of networks, and lasts for a long time. It is also described as the choice between bifurcated or non-bifurcated (working) flows. We found it hard to decide intuitively which would outperform the other.

In fact, we are dealing with a joint network design problem for a shared protection scheme. Both the working paths and protection cycles will be determined in process. Just as there is confusion about whether we should restrict the number of working LSPs per demand, a similar argument lies in the protection as well. That is the second question. What happens if we allow only one FIPP *p*-cycle to protect a working LSP? Degraded performance on capacity efficiency is what can be foreseen. Also, it is not certain whether by introducing this constraint it will generate fewer cycles in the final solution. The fact that a working LSP can only be protected by an FIPP *p*-cycle does not mean the outcome of a cycle total will equal the number of working paths. Remember

that FIPP cycles are shared by these LSPs, so the number of cycles that appear in the solution is still nondeterministic. However, adding such a restriction is not like the ones which simply introduce one more constraint to fix the total number of relevant variables to one. On the contrary, it simplifies our mathematical model. The major benefit is, with the help of this assumption, we can omit one family of continuous variables and remove all the numerically sensitive parameters. Therefore, we explore the MFIPP-JCA problem in 3 different cases with a common objective which is to minimize the cost of total capacity.

Additionally, several statements are needed to be emphasised prior to the presentation of our mathematical models. As discussed, the variables that relate to capacity will be modelled as continuous valued due to the characteristics of capacity in an MPLS context. We only consider single failure scenarios, either a link failure or a node failure, because the probability of multiple failures is generally assumed to be rather small in practice. The actual unidirectional traffic demands taken into account are assumed to be symmetrical. Next, in our network design, the demand from one node to another is merged with the one from the opposite direction. Thus, it can be taken for granted that the demand is bi-directional, and given as “between” rather than “from-to”. Accordingly, virtual *p*-cycles are assumed to be bi-directional as well. We represent networks with undirected graphs where links are undirected. Lastly, there is only one type of traffic demand between node pairs. No differential services are taken into account. A detailed description is given in the following sections.

4.5.2 Case 1: Bifurcated Normal Routing (BR)

The first case to be introduced involves bifurcated normal routing with no restriction on the protection structure. The demand volume can be split among several paths in any desired way under a normal situation. Put differently, unit demands belonging to the same demand bundle do not have to be routed over the same path. Any combination of

candidate paths that serve the demand between a node pair may be used as long as the sum of the allocated capacity over these paths satisfies the demand requirement for that particular node pair. In the failure state, an affected LSP is allowed to be protected by a number of FIPP *p*-cycles. This case gives the theoretical upper bound on the capacity efficiency among our MFIPP-JCA models due to the maximum relaxation on both the routing and protection. The formulations are given as follows.

Case 1 MFIPP-JCA-BR Model

(Splitting of flows is allowed under both the normal and the protection state.)

Minimize:

$$\sum_{\forall j \in S} c_j \cdot (w_j + s_j) \tag{4.1}$$

Subject to:

$$\sum_{\forall q \in Q_d} g_q \geq h_d \qquad \forall d \in D \tag{4.2}$$

$$\sum_{\forall q \in Q} \pi_j^q \cdot g_q \leq w_j \qquad \forall j \in S \tag{4.3}$$

$$\sum_{\forall k \in P} x_q^k \cdot n_q^k \geq g_q \qquad \forall q \in Q \tag{4.4}$$

$$n^k \geq n_q^k \qquad \forall q \in Q, \forall k \in P \tag{4.5}$$

$$s_j \geq \sum_{\forall k \in P} \pi_j^k \cdot n^k \qquad \forall j \in S \tag{4.6}$$

$$\gamma_q^k \geq \nabla \cdot n_q^k \qquad \forall q \in Q, \forall k \in P \tag{4.7}$$

$$\gamma_q^k \leq \Delta \cdot n_q^k \qquad \forall q \in Q, \forall k \in P \tag{4.8}$$

$$\partial_{mn} + \gamma_m^k + \gamma_n^k \leq 2 \qquad \forall (m,n) \in Q | m \neq n, \forall k \in P \tag{4.9}$$

All models for the three cases aim to minimize the cost of total capacity which includes the cost for both working and reserved capacity, as required in Equation (4.1). There are eight constraints in total, which can be related to the seven conformance requirements.

The seven requirements originate from consideration of the following aspects.

1. Full demand routing
2. Working capacity placement
3. Enough protection for each path
4. Total capacity of an FIPP *p*-cycle
5. Spare capacity placement
6. Runtime binary decision (for choosing a cycle as an FIPP *p*-cycle)
7. Disjointness requirement

Constraint (4.2) ensures that all of the unit demands for every demand relation are routed along one or more candidate working paths belonging to that demand relation. Constraint (4.3) makes sure that the working route decisions made for every demand are adequately capacitated. Constraint (4.4) places enough capacity on FIPP *p*-cycles to protect the working paths selected to be part of the solution. It guarantees at least 100% protection for the chosen working paths. Based on the prior constraint (4.5), it calculates the total capacity required for each cycle. It updates the variable n^k to keep track of exactly which cycles protect which paths and their quantities. To be specific, it makes sure that the backup capacity assigned to cycle k is equal to or larger than that required by each path protection using cycle k. According to constraint (4.6), adequate spare capacity is allocated on each link for the FIPP *p*-cycles selected. It guarantees that there is enough spare capacity to support the set of FIPP *p*-cycles used for restoration. Constraint (4.7) and (4.8) makes runtime binary decision. It resorts to the reserved capacity for a cycle to decide whether this cycle is exactly chosen to use in the ultimate solution. An elaborate discussion on the function of this pair of constraint is already given in Section 3.3.2. Finally, constraint (4.9) explicitly forces the set of working paths protected by the same cycle to be mutually disjoint.

4.5.3 Case 2: Non-Bifurcated Normal Routing (NBR)

Non-bifurcated flows are also called single-path flows or unsplittable flows. The NBR case is designed on the premise that there must be only one working LSP for each demand. Still, there is not any restriction on the protection structure. It is acceptable to protect an LSP by multiple FIPP *p*-cycles. This case is much closer to realistic situations. It is implicitly assumed that installed capacity is enough to realize all desired traffic by single-path routing under the normal state, while in the failure situation the capacity might be insufficient to carry the affected demand by a single protection structure. The objective function is the same as in Case 1 and is not repeated any more. The corresponding constraints are presented below.

Case 2 MFIPP-JCA-NBR Model

(Splitting is only allowed under the failure state. This means only one primary LSP is established per demand.)

Subject to:

$$\sum_{\forall q \in Q_d} \alpha_q = 1 \qquad \forall d \in D \qquad (4.10)$$

$$\sum_{\forall d \in D} \sum_{\forall q \in Q_d} \pi_j^q \cdot \alpha_q \cdot h_d \leq w_j \qquad \forall j \in S \qquad (4.11)$$

$$\sum_{\forall k \in P} x_q^k \cdot n_q^k \geq \alpha_q \cdot h_d \qquad \forall d \in D, \forall q \in Q_d \qquad (4.12)$$

$$n^k \geq n_q^k \qquad \forall q \in Q, \forall k \in P \qquad (4.13)$$

$$s_j \geq \sum_{\forall k \in P} \pi_j^k \cdot n^k \qquad \forall j \in S \qquad (4.14)$$

$$\gamma_q^k \geq \nabla \cdot n_q^k \qquad \forall q \in Q, \forall k \in P \qquad (4.15)$$

$$\gamma_q^k \leq \Delta \cdot n_q^k \qquad \forall q \in Q, \forall k \in P \qquad (4.16)$$

$$\partial_{mn} + \gamma_m^k + \gamma_n^k \leq 2 \qquad \forall (m,n) \in Q | m \neq n, \forall k \in P \qquad (4.17)$$

The main difference between this model and the last one is that it involves a binary

variable to encode the choice of the working path per demand relation rather than the exact amount of traffic routed on the paths. The mathematical formulations to express the top three requirements (i.e., fully demand routing, working capacity placement and enough protection for each path) are changed appropriately. Constraint (4.10) explicitly calls for single path routing during the normal situation. Since there is only one working LSP for each demand pair, all demand volume will be carried on this single working path. There is no need to introduce an extra variable to represent the required capacity of the paths since it can be replaced by the amount of the corresponding traffic volume. Constraint (4.11) guarantees that there are sufficient capacities on the links to support the single path routing of traffic demands. Next, constraint (4.12) is modified from Equation (4.4), ensuring sufficient protection (100% or more) for a path selected to be a working LSP in the final solution. By "sufficient protection", we mean the amount of traffic over this path is assured to be restored by the current solution. This equation substitutes the amount of traffic flow over path q with the demand volume required between the relevant demand pair. This replacement can be taken due to the single path routing as well. The remaining constraints are identical to those in the MFIPP-JCA BR model as a result of the same assumptions being made about the protection structure.

4.5.4 Case 3: Non-Bifurcated Normal Routing with Single Backup Structure (SNBR)

The third case places tight requirements on both normal routing and the protection structure. It restricts the working demand such that it must be routed on a single LSP in the normal state and it further limits the number of FIPP p-cycles to protect an LSP. To make it an extreme case, we assume that only one FIPP p-cycle can be used to protect a path. This confirms the title assigned to this single backup structure. It can be expressed as no split routing during restoration with one exception. The exclusion occurs when a path is completely straddling an FIPP p-cycle. The restored flow has to be split and routed over the two anticlockwise protection paths which are provided by the FIPP

p-cycle. Such a split for the traffic demand is unavoidable and even desired, because it is the straddling relationship that contributes to the high capacity efficiency of *p*-cycle based schemes. Note that sharing protection of a common FIPP *p*-cycle among multiple paths is still retained. As introduced in Section 4.5.1, there are both strengths and weaknesses to this approach. Though it might not be as capacity effective as the other two cases, it is still attractive due to the simplified mathematical formulation (see below). There are fewer continuous variables and constraints than the above two cases. The most desired reward of the strict restrictions is that it removes all numerical sensitive factors in the model. Here are the resulting ILP formulations.

Case 3 MFIPP-JCA-SNBR Model

(Splitting is not allowed on any condition but for only one exception[6].)

Subject to:

$$\sum_{\forall q \in Q_d} \alpha_q = 1 \qquad \forall d \in D \tag{4.18}$$

$$\sum_{\forall d \in D} \sum_{\forall q \in Q_d} \pi_j^q \cdot \alpha_q \cdot h_d \leq w_j \qquad \forall j \in S \tag{4.19}$$

$$\sum_{\forall k \in P} \gamma_q^k = \alpha_q \qquad \forall q \in Q \tag{4.20}$$

$$\gamma_q^k \leq 2 \cdot y_q^k \qquad \forall q \in Q, \forall k \in P \tag{4.21}$$

$$n^k \geq \sum_{\forall q \in Q_d} \gamma_q^k \cdot y_q^k \cdot h_d \qquad \forall d \in D, \forall k \in P \tag{4.22}$$

$$s_j \geq \sum_{\forall k \in P} \pi_j^k \cdot n^k \qquad \forall j \in S \tag{4.23}$$

$$\partial_{mn} + \gamma_m^k + \gamma_n^k \leq 2 \qquad \forall (m,n) \in Q | m \neq n, \forall k \in P \tag{4.24}$$

The MIP formulations for this model dramatically diverge from that of the other two

[6] As presented above, the exception is when there is a straddling relationship between a path and its protection FIPP p-cycle.

models, especially the MFIPP-JCA-BR model. This can be attributed to the fact that an additional restriction is imposed on the protection scheme, which fixes the number of cycles to protect a working path to be just one. The objective is still unchanged and formulated as Equation (4.1). Constraints (4.18) and (4.19) are the same as the first two constraints in the MFIPP-JCA-NBR model, which ensure that full demand routing over a single working path and that there are sufficient capacitated links respectively. Constraint (4.20) makes sure that each selected path is protected by only one cycle. The left hand side of the equation is the total number of cycles used to protect path q, while the right hand side is the binary decision of whether path q is used to finally route the corresponding demand. When path q is chosen to be the single working path, the right hand side turns out to be one. Henceforth, the number of cycles to protect it is forced to be one as well. On the other hand, if path q is not selected to be the primary path, the binary decision will become zero. This further implies that no cycle is required to protect such a path. Constraint (4.21) imposes the control over the selection of FIPP p-cycles. It guarantees that cycle k is taken to protect path q only if it is capable of doing so. Otherwise, it is not allowed to be the protection cycle for path q. To be specific, suppose that there is no protection relationship between cycle k and path q (i.e., this cycle cannot protect the path). This is conveyed by letting y_q^k to be zero, which further makes γ_q^k zero as well. Whereas in the case of cycle k it can protect path q, regardless of the exact protection relationship, the right hand side of the inequality turns out to be equal to or larger than one. This allows the cycle to be the potential protecting cycle for path q. Constraint (4.22) makes sure that cycles are sufficiently capacitated. Such a condition should be checked on a cycle basis for every demand relation. Due to the assumption of single-path flows, the summation on the right hand side is actually the required capacity on cycle k to protect the chosen path which carries the traffic of demand d. Constraint (4.23) guarantees that adequate backup capacity is reserved on links complying with the selected FIPP p-cycles for restoration. Finally, constraint (4.24) explicitly enforces the disjointness requirement on the set of working

paths protected by the same cycle, as stressed in the principle of FIPP *p*-cycles.

The three cases are summarized in Table 5.

Table 5 Summary of MFIPP-JCA models

Cases	BR	NBR	SNBR
Working LSPs per demand	no restriction	1	1
FIPP *p*-cycles per LSPs	no restriction	no restriction	1
Objective	$\sum_{\forall j \in S} c_j \cdot (w_j + s_j)$		
Fully demand routing	$\sum_{\forall q \in Q_d} g_q \geq h_d \quad \forall d \in D$	$\sum_{\forall q \in Q_d} \alpha_q = 1 \quad \forall d \in D$	
Working capacity placement	$\sum_{\forall q \in Q} \pi_j^q \cdot g_q \leq w_j \ \ \forall j \in S$	$\sum_{\forall d \in D} \sum_{\forall q \in Q_d} \pi_j^q \cdot \alpha_q \cdot h_d \leq w_j \quad \forall j \in S$	
Enough protection for each path	$\sum_{\forall k \in P} x_q^k \cdot n_q^k \geq g_q \ \ \forall q \in Q$	$\sum_{\forall k \in P} x_q^k \cdot n_q^k \geq \alpha_q \cdot h_d \quad \forall d \in D, \forall q \in Q_d$	$n^k \geq \sum_{\forall q \in Q_d} \gamma_q^k \cdot y_q^k \cdot h_d \quad \forall d \in D, \forall k \in P$
Total capacity of an FIPP *p*-cycle	$n^k \geq n_q^k \quad \forall q \in Q, \forall k \in P$		
Runtime binary	$\gamma_q^k \geq \nabla \cdot n_q^k \quad \forall q \in Q, \forall k \in P$		$\sum_{\forall k \in P} \gamma_q^k = \alpha_q \quad \forall q \in Q$

decision	$\gamma_q^k \le \Delta \cdot n_q^k \quad \forall q \in Q, \forall k \in P$		$\gamma_q^k \le 2 \cdot y_q^k \quad \forall q \in Q, \forall k \in P$
Disjointness Requirement	$\partial_{mn} + \gamma_m^k + \gamma_n^k \le 2 \quad \forall (m,n) \in Q \mid m \ne n, \forall k \in P$		
Spare Capacity Placement	$s_j \ge \sum_{\forall k \in P} \pi_j^k \cdot n^k \quad \forall j \in S$		
numVars	$2S + P + Q + PQ$	$2S + P + PQ$	$2S + P$
numBins	PQ	$PQ + Q$	$PQ + Q$
numCons	$2S + D + Q + 3PQ + PQ^2$		$2S + D + Q + PQ + PQ^2 + DP$

Note:

S – Number of links P – Number of eligible cycles

D – Number of demand relations Q – Number of eligible paths

numVars – Number of continuous variables

numBins – Number of binary variables

numCons – Number of constraints

Constraints are classified according to the seven aspects listed in Section 4.5.2. The MIP of MFIPP-JCA-SNBR is quite different from the other two. There is no need to make a binary choice that indicates whether to use a cycle to protect a particular path according to the reserved capacity for the cycle, at runtime. So in the SNBR case, we fill Equation (4.20) and (4.21) in the corresponding rows to simplify the table. In the last three rows, the problem size is analyzed in terms of the number of continuous variables, binary variables and constraints respectively.

4.5.5 A Case Study on the Problem Size

It can be concluded from Table 5 that the SNBR model requires the smallest number of variables and constraints. Consider the Atlanta network[37] as shown in Figure 22,

which we have used as one of our test networks. This network consists of 15 nodes and 22 links. The number of fully meshed demands amount to 105 and there are 80 simple cycles in total. Suppose we use 2 shortest paths as candidate paths for demand routing between a node pair, the parameters and resulting numbers of variables and constraints are listed in Table 6.

Table 6 Problem size for Atlanta network

S	22	P	80
D	105	Q	210
Cases	BR	NBR	SNBR
numVars	17134	16924	124
numBins	16800	17010	17010
numCons	3578759	3578759	3553559

The table shows that in the SNBR case, the number of continuous variables is amazingly smaller in the SNBR case than for the other two cases. Although the number of constraints in these three cases is quite similar, the reduction in the case of SNBR is still quite favorable. With respect to the binary variables, there is the same number of these in the case of both NBR and SNBR, which is slightly more than for the instance of bifurcated normal routing. The problem size interprets the computational effort needed to solve the corresponding MIP model. The application of the MFIPP-JCA-SNBR model is much more beneficial with regard to shortening the solution time.

4.6 Summary

As the core of our research, we have attempted to adapt the FIPP *p*-cycle scheme into MPLS networks and this has been introduced and discussed in this chapter. Firstly, the impact of the MPLS environment on designs was discussed. This was followed by the

reasons for choosing FIPP-SCP as the basic model, along with some special issues regarding required adjustments. Also, there was a general comparison between the FIPP *p*-cycle mechanism and some classical recovery schemes used in MPLS networks, with an emphasis on the scalability issue.

Three models for jointly optimized FIPP *p*-cycle networks were proposed, and these were denoted as BR, NBR and SNBR respectively, all of which belong to the set of MFIPP-JCA network designs. The intent of our joint optimized design was to seek more opportunities to enhance the quality of an FIPP *p*-cycle protected network, in terms of the overall capacity cost. The NBR method is an exactly parallel joint network design to the basic SCP model, whereas BR allows bifurcated traffic flows with an aim to further reduce the total cost of network capacity. By contrast, SNBR is quite different from the other two, owing to the restriction imposed on the protection domain that only a single cycle can be used to protect a working path.

Chapter 5 Experimental Results for MPLS Network Design

5.1 Experimental Setup

All three MFIPP-JCA models were coded using AMPL and then solved using the MIP solver of IBM CPLEX 12.1 on a Windows XP machine with a 2.13 GHz Intel® Core™2 Duo CPU P7450 and 2GB of RAM. The experiments were carried out in two phases.

Since there is no standard test suite for such mesh protection studies, we developed a 9-node network family all with 12 spans, in order to cover a range of network patterns. Their network topologies are shown in Figure 21, which are small and moderately connected with an average nodal degree of around 2.67. Although, they are not representative of real networks in size, they play a useful role in the research methodology, because this network family allows us to do a comprehensive comparison study of different network architectures on a test case where all eligible cycles and all possible demand pairs are concurrently represented in the optimization problems. It is worthy to mention that the New Zealand core network topology has 9 nodes with spans of 12 too, consisting of a ladder-like structure. Unfortunately, such a structure cannot support the use of FIPP *p*-cycles. So we also intend to investigate whether it can be applied to a network with the same connectivity through the first stage of testing. Link costs are all set to 1, in order to emphasise the pure topological aspects of the network. Demand relations are fully-meshed, the number of which is 36. We considered 6 different sets of demand values, where each was uniformly distributed within a specific range. The total volume of demands was identical to 180 in all test cases for this network family. The reason for such an experimental design is that a goal of this network family was to allow studies on the impact of demand distribution on network performance.

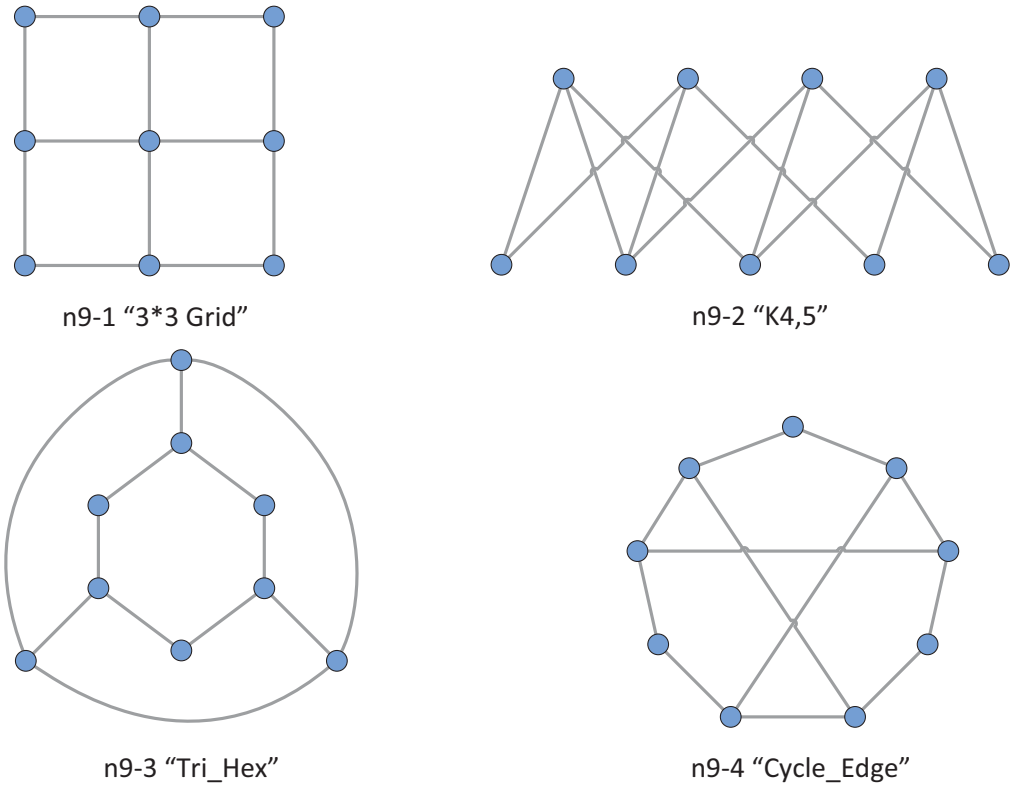


Figure 21 A 9-node network family

In the second place, three real networks are chosen to be test instances. They are the Atlanta network, German network and European COST 239 network respectively, as shown in Figure 22, Figure 23, and Figure 24. Table 7 gives the statistical information of the three test networks. Atlanta network contains 15 nodes and 22 links, while German network consists of 17 nodes with 26 spans. Both of them are dense with a nodal degree of around 3. The sixth column in the Table 7 represents the number of non-zero demands, which is identical to the number of active demand relations as the input to the optimization process. The total number of simple cycles in these two test cases lies in the range that can be handled with our machine, and henceforth no additional pre-selection over cycles needs to be taken. That is to say, all of the cycles are entitled to be candidate cycles in these two instances.

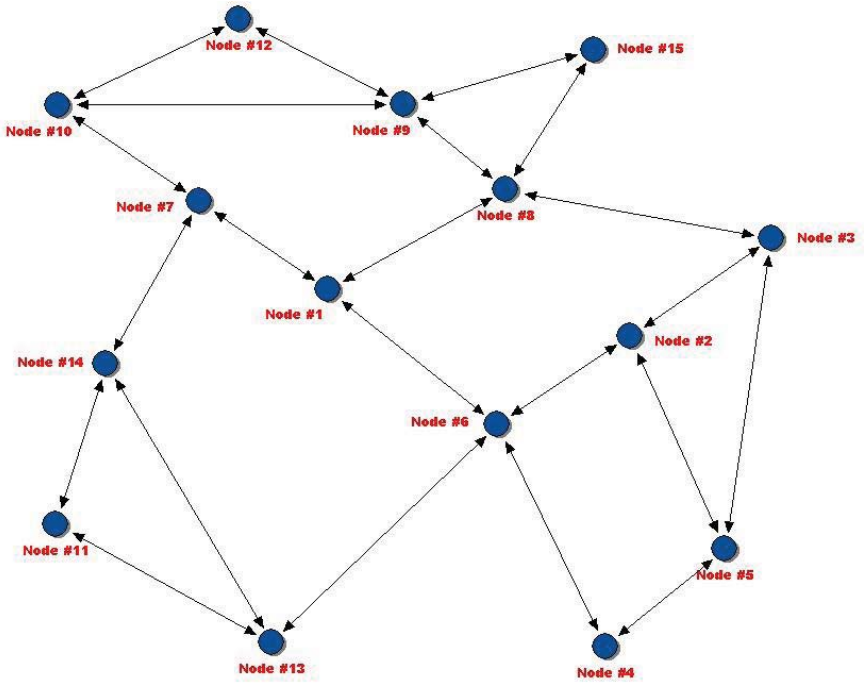


Figure 22 Atlanta network

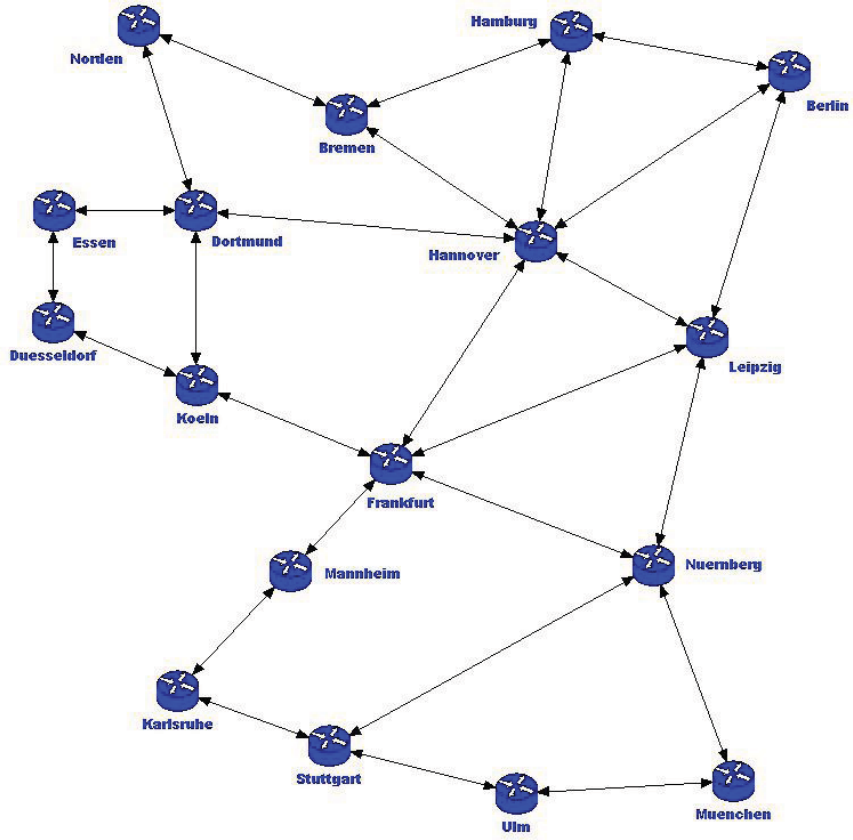


Figure 23 German network

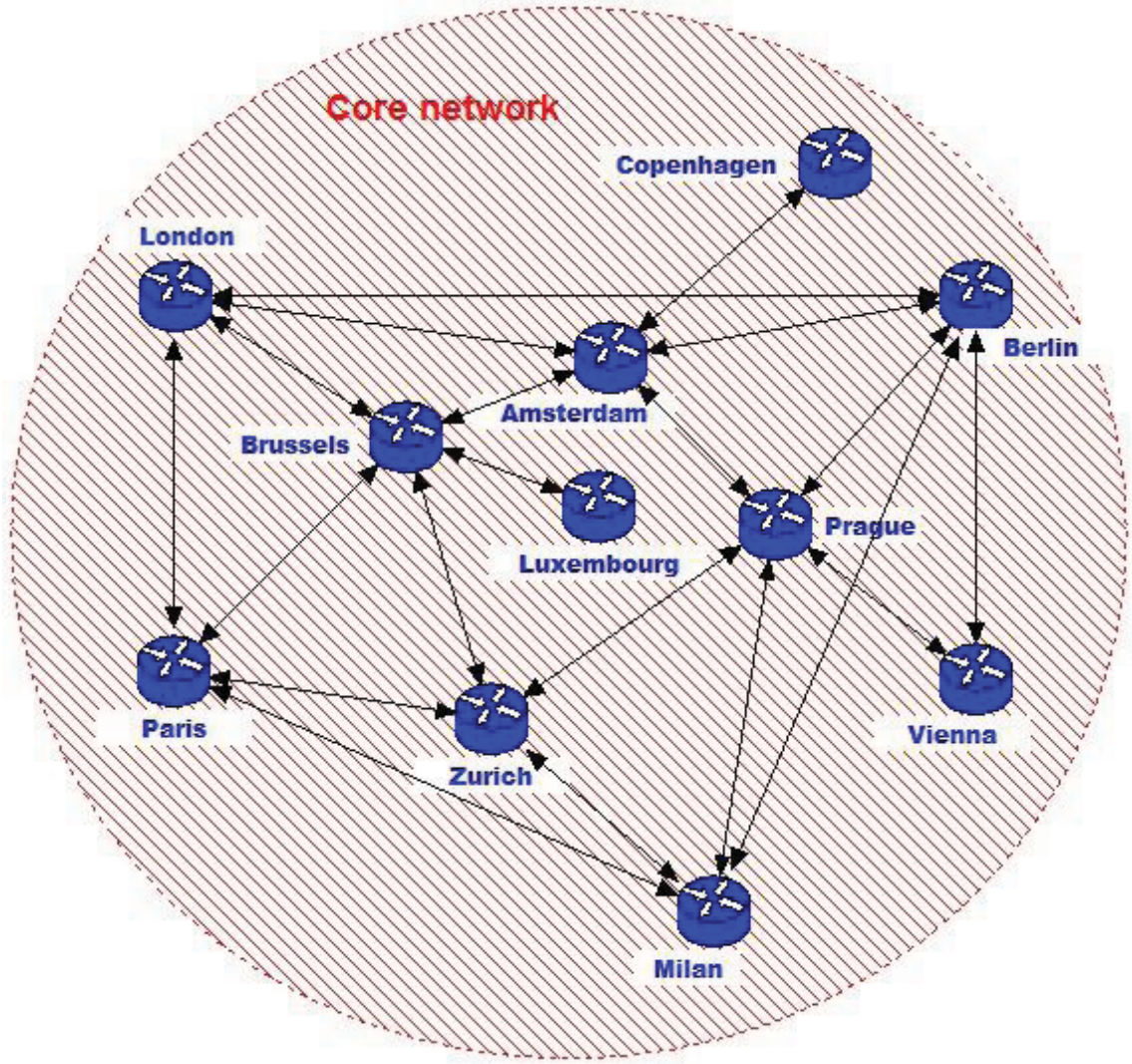


Figure 24 COST 239 network

Table 7 Real test networks

Networks	Nodes	Links	Nodal Degree	Fully-meshed Demands	Non-zero Demands	Total Cycles
Atlanta [37]	15	22	2.93	105	50	80
German [38]	17	26	3.06	136	55	135
COST239 [39]	11	26	4.73	55	53	3531

The last test network COST 239 has a much higher connectivity of 4.73, which results in a total of 3531 simple cycles. The number of active demand relations is 53, which is

extremely close to the total number of possible demand relations. Prior work in [40] clearly demonstrates that large *p*-cycles yield prohibitive backup path lengths, whereas small *p*-cycles also outperform the large ones in terms of their requirements on the address and state. Moreover, according to the findings in [41], the much simpler process of limiting the circumference of cycles is conceived to be a highly effective surrogate to the more precise procedure of explicit hop-limiting on paths, with respect to direct limitations on backup path lengths in *p*-cycle based design problems. Shorter backup paths are beneficial to the reduction of traffic delay, as well as packet loss. Therefore, we choose a preset number of shortest cycles to be eligible for COST239 instance.

Further details regarding the network topology and demand set in use can be found in Appendix A. Though scaled with different factors, the span cost of these real networks is proportional to the geographic distance between nodes. A k-shortest path algorithm with link-distance as a metric determines the eligible working paths for each node pair. In all the tests, 2 eligible working path candidates were generated for every relation exchanging a non-zero demand. The value of this parameter could be set higher to give the solver a large number of options, but was kept relative low in our studies of the problem to a manageable size, given our limited hardware conditions. A depth-first search algorithm for cycle enumeration of a graph was adapted from [9] and implemented with the C++ language for all test instances. In addition, as we stated above, there should be a preliminary step to select a set of eligible cycles from all the elementary cycles, subject to an upper bound on the circumference or the number of hops, particularly for the COST239 case. For reference comparison, corresponding solutions were generated using the FIPP-SCP model as well, with the requirement of node-disjointness and absolute exclusion of z-cases. Note that all non-binary variables in the FIPP-SCP model are set to be continuous, for the reason that the capacity is based on Megabits in the MPLS context, not a discrete value any longer. All the solutions were obtained by CPLEX within a runtime limit of 10 hours.

5.2 Results and Discussion

5.2.1 9-Node Network Family

Figure 25 to 28 show the total capacity required by different models for the corresponding 9-node network (see Figure 21 for the network topologies). The numbers in brackets indicate the range of the demand set, e.g., (0,10) means the demand values in this set were uniformly distributed between 0 and 10; (5,5) was used to specify one of the demand sets, whose values are all equal to 5. The notation SCP represents the results were obtained with the FIPP-SCP model, whereas BR, NBR, SNBR are the three models in our MFIPP-JCA network design. Note that SCP is short for FIPP-SCP models throughout this thesis, whereas SCA is used to denote a type of network design, in which only spare capacity is concerned. More details about all the tests can be found in Appendix B.

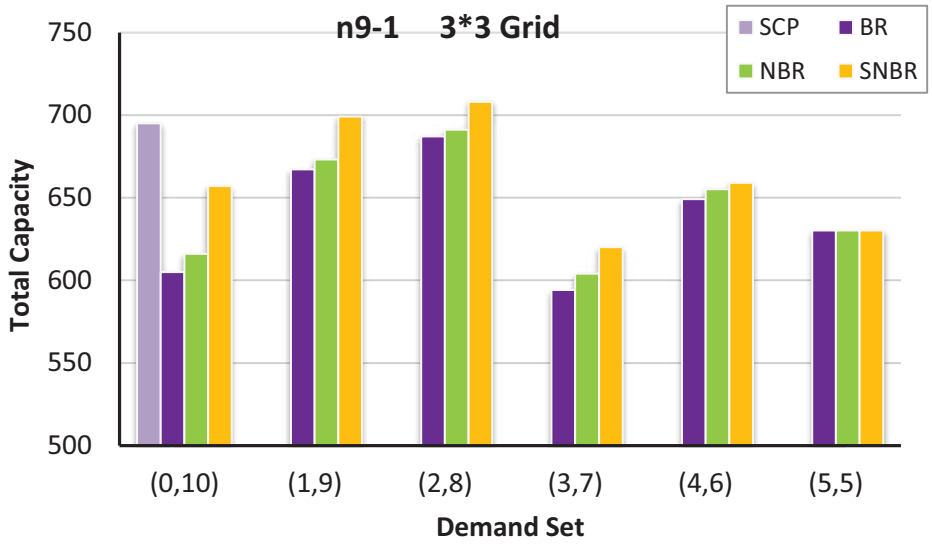


Figure 25 Total capacity for n9-1

As can be seen from Figure 25, among the tests in the SCP model for the first 9-node network, only the instance with demand set (0,10) can be solved. This is due to the fact that there are 4 zero demands out of a total of 36 relations in this demand set. The corresponding 4 working paths for those inactive demands were not involved in the

optimization process. As a result, 32 working paths, rather than 36, need to be protected by the same set of cycles. This yields a greater opportunity to find a solution that conforms to the path disjointness requirement on the working paths that are protected with the same cycle. For the instance with the (0,10) demand set, our models results in less total cost for the network capacity, relative to the SCP solution. Moreover, all the costs of the working capacity for our methods are the same as that for the SCP method, which is required by the (single) shortest path routing. By contrast with the outcome of the SCP solution, for the cases with other demand sets, all of our MFIPP-JCA models are capable of producing solutions of good quality, the redundancies of which are less than 100%, with the least being 73.7%.

Although as for the parameters stated above, 2 eligible paths per demand were input to the optimization, NBR limits the final number of working paths that can serve a demand relation to be 1. Accordingly, the resilient network is eventually to be deployed with a single path routing mechanism. This model can be classified as an extreme case of BR, in which no restriction is imposed on the number of working paths that can be used for each demand. Hence, it is fairly reasonable to expect that BR will yield a less costly solution, relative to the NBR model. Similarly, SNBR requires only one working path per demand with one cycle to protect it, and functions as a special case of NBR. Compared to NBR, a greater total cost can be predicted in case of SNBR. Based on an examination of Figure 25, we see that BR generates the least cost in all cases except the last one, where the results for the three models turn out to be the same. The total costs for NBR are slightly more expensive than that for the BR model, but less expensive than that for SNBR in the solution with the first 4 demand sets.

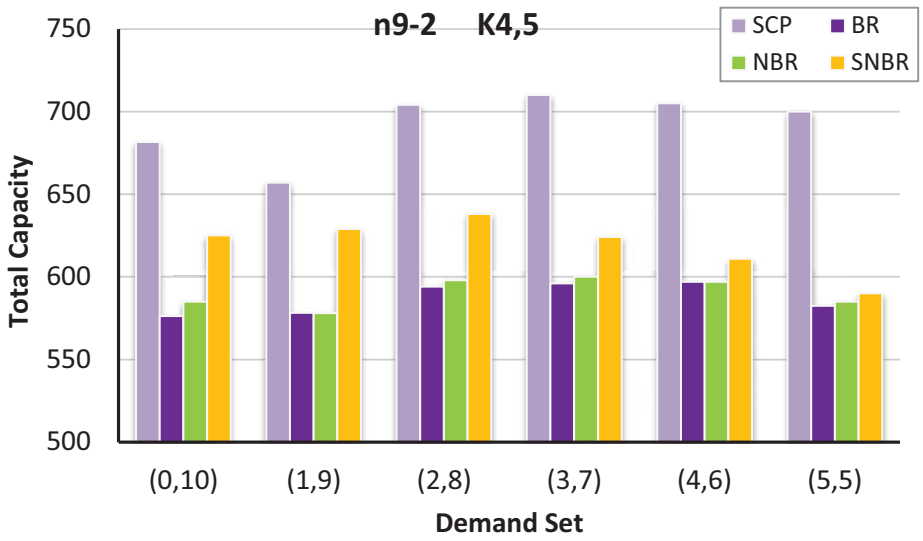


Figure 26 Total capacity for n9-2

A similar trend can also be found in other networks that have a distinct topology. However, it happens occasionally for those networks that the total cost of BR exceeds that obtained from NBR. It is obviously contrary to the theoretical estimation and can be attributed to the fact that none of those instances for BR and NBR were solved to optimality within the given time limit. On the other hand, solutions for SNBR, without exception, were based on complete CPLEX terminations with a MIPGAP of 0.01. As illustrated in all of the test cases, the total costs for SNBR are no less than those for NBR or BR, which is truly reflective of the theory upon which they were based.

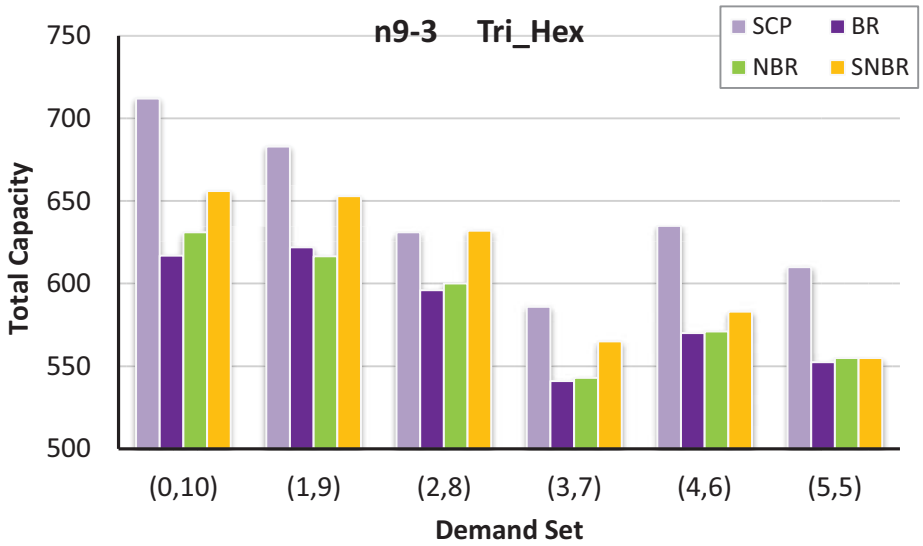


Figure 27 Total capacity for n9-3

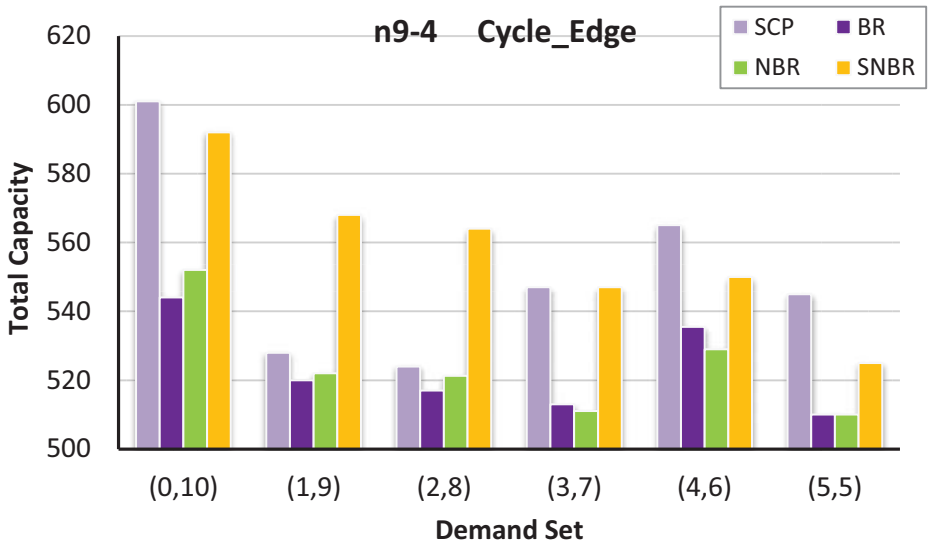


Figure 28 Total capacity for n9-4

In general, the NBR method is usually comparable with SCP. The reason is that they both insist on a single path routing with no restriction on the number of cycles that can be used to protect a path. The difference is, more eligible paths are allowed to be involved in the NBR case, whereas in SCP, the shortest paths are fixed as the default for each demand prior to the optimization process. NBR can be regarded as the strict JCA counterpart to SCP. Given more alternative paths as input, NBR should produce more capacity-economic solutions. In addition, since BR relaxes the routing to allow bifurcation, it can further improve the network performance in terms of total capacity. This is confirmed with the results for the whole 9-node network family. The following discussion is based on the corresponding costs for total capacity using SCP as a benchmark. For the instance of the n9-2 network with the demand set of (5,5), the reduction in the total cost for BR reaches a peak of 16.8%, relative to the solution for SCP. The parallel result for NBR also amounts to the best among all the tests using the same method, with a reduction of 16.4%. It is worth noting that such an improvement is obtained at the expense of a slightly more expensive investment in the working capacity, 2.9% more than that required by shortest path routing with both BR and NBR.

The performance of SNBR fluctuates dramatically. Tests on network n9-2 shows the

total costs for SNBR are consistently lower than those for SCP, even reaching a maximum cost-saving of 15.7%. Meanwhile, according to Figure 28, within the 6 different demand sets, the results for 3 cases are superior to the corresponding SCP solution, and for the instance based on the demand set (3,7), the total costs are identical for both SCP and SNBR. However, in the remaining two cases, the total costs for SNBR are greater than those for SCP, both increasing by 7.6%. Despite the fact that SNBR potentially yields a better solution in terms of total capacity, it is more likely to generate worse results relative to the SCP case. An interpretation of this outcome can be offered by noting that allowing more candidate paths to be input does have a positive effect on improving these solutions, but the restriction that only one cycle can be used to protect a path is a more dominant factor for SNBR. The extent of sharing spare capacity decreases as a consequence of the limited number of cycles per path protection.

As shown in Figure 29, the average number of FIPP *p*-cycles for the instances involving 6 demand sets is summarized for the 9-node network family. In most cases, NBR produces fewer FIPP *p*-cycles than for the BR and SCP cases. A significant observation is that SNBR generates the least number of FIPP *p*-cycles among the four methods under consideration.

Although the total number of cycles in the four 9-node networks is already very small (none of them exceeds 15), the SNBR solutions produce network protection plans with no more than 8 FIPP *p*-cycles. Note that they are capable of providing path protection for the test cases with a fully-meshed demand pattern, which has 36 relations. The results of the tests on this artificial network family are consistent with our expectations. With respect to the total cost of an FIPP *p*-cycles protected network, the NBR model is superior to the SCP model while the BR method further enhances the performance. Although SNBR might not be a good choice for those who are seeking an extremely good cost-saving network protection scheme, it yields a very small number of FIPP

p-cycles and requires much less operational effort.

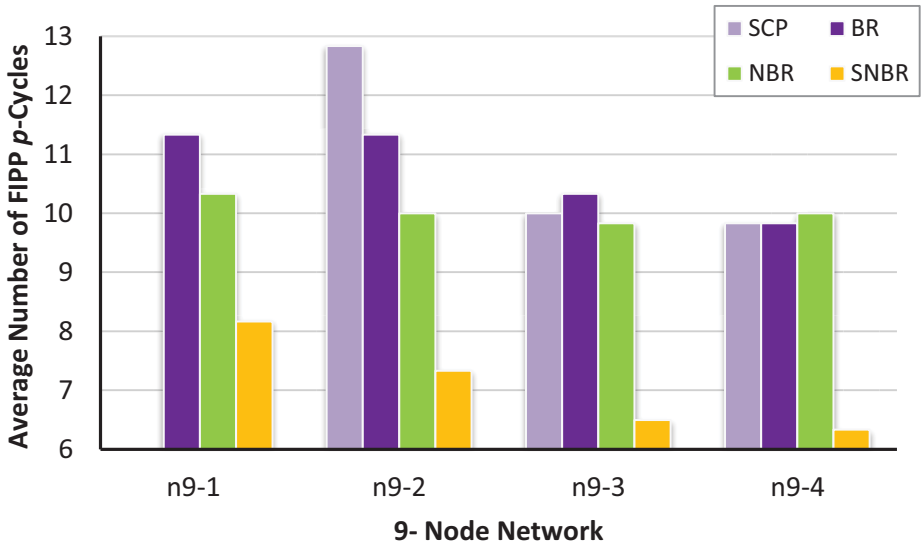


Figure 29 Average number of FIPP *p*-cycles in the solution of all models for the 9-node network family

5.2.2 Real Networks

This section is a first attempt to apply our joint designs to realistic networks. The detailed results for the Atlanta and German networks are documented in Table 8 and 9, respectively. These tables contain the final design costs for working capacity, spare capacity, as well as total capacity assignments. Furthermore, the number of cycles that are selected to be FIPP *p*-cycles in the solution is also reported, along with the MIPGAP[7] at which the optimization terminated. Note that the COST 239 test case is not involved within the current section, a discussion of this case is deferred to the following section.

[7] MIPGAP is the gap between the best found MIP solution and the optimal of the relaxed LP.

Table 8 Results for the Atlanta network

Model	Cost of Work	Cost of Spare	Total Cost	Number of Cycles	MIPGAP (%)
SCP	284876	350392	635268	39	47.7
BR	292529	291822	584351	50	27.8
NBR	291476	298971	590447	35	22.3
SNBR	291593	384460	676053	12	9.1

Table 9 Results for the German network

Model	Cost of Work	Cost of Spare	Total Cost	Number of Cycles	MIPGAP (%)
SCP			N/A		
BR	352507	352478	704985	89	33.7
NBR	350007	356948	706955	68	23.1
SNBR	350988	437223	788211	11	16.6

The trend can be seen from both of the tables that BR generates the most cost-efficient protection schemes and NBR is the second cost-saving methods, whereas the SNBR solution is the most expensive among all methods in the comparison. In fact, the BR and the NBR solutions are fairly close to each other in terms of their total cost of network capacity. The difference is within 1% for both cases of the Atlanta and German networks. On the other hand, the total cost for SNBR is 15.7% more than the corresponding BR solutions in the Atlanta instance and 11.8% more in the German test case. This again verifies the theoretical statements that BR, as the most flexible resilient network design method, yields the most capacity efficient solution for the FIPP *p*-cycles protection scheme. Additionally, it is unlikely to obtain an equally cost-economic solution for SNBR, where the sharing protection is strictly limited.

Despite the fact that the solver failed to return a solution to the problem for the German network for the SCP method, which was identified as being infeasible, all of the three

MFIPP-JCA models produce acceptable solutions. Moreover, the cost improvement when comparing the BR and NBR results to the SCP solutions is as much as 8.0% and 7.1% for the Atlanta network. This is clearly illustrated with a line graph (see Figure 30).

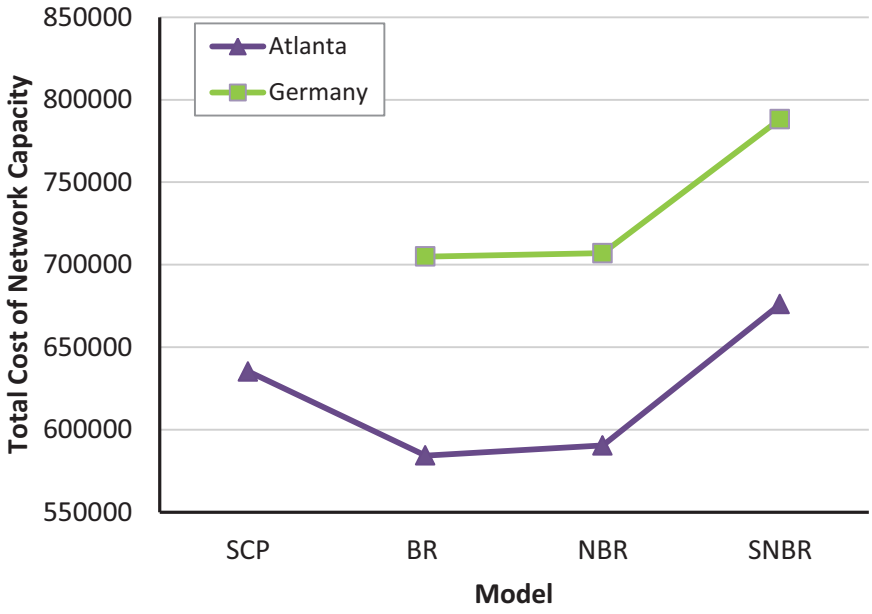


Figure 30 Total cost of network capacity for Atlanta and German network

Unfortunately, relative to the results for SCP, the solution for SNBR requires an additional 6.4% cost for the total capacity of the Atlanta network. However, the resulting protection scheme only contains 12 FIPP *p*-cycles, which is far fewer than that needed in the corresponding solutions for the other models. By contrast, the BR and the NBR solutions require 50 and 35 cycles to be FIPP *p*-cycles, respectively. Note that many of them (for example, 17 in 50 cycles for BR in the Atlanta case) occupy a minimum spare capacity of 0.00001. The reason for this is that the decision of FIPP *p*-cycles is made by a pair of inequalities on the fly, which are numerically sensitive. The NBR method is superior to both SCP and BR models, in terms of the number of FIPP *p*-cycles that should be handled operationally.

Furthermore, the cost of working capacity only increases a little, compared to that

required by the shortest path routing. For all MFIPP-JCA models, none of the growths exceed 2.7% for the Atlanta case, and 1.3% for the German case. This means the working paths in the solution do not significantly deviate from the shortest paths. Recall that there were two eligible path choices per demand in these tests. Both the NBR and the SNBR methods allow only one working path per demand. Accordingly, the decision on the first and second working path options is carefully examined for both NBR and SNBR cases. An interesting finding is that the number of second shortest paths in use for NBR and SNBR is the same, for both the Atlanta and German networks. Thus, Figure 31 illustrates the percentage of the first and the second shortest paths in the final working path set, for the Atlanta and German case respectively. Again, it is identical for both the NBR and the SNBR solutions; hence we depict only one chart for the two methods. In all cases, around 10% of the working paths are the second shortest ones. The results for NBR confirm that adding just one alternative for the candidate paths allows a more flexible selection of the working paths and results in a more cost-efficient network design, relative to the SCP method.

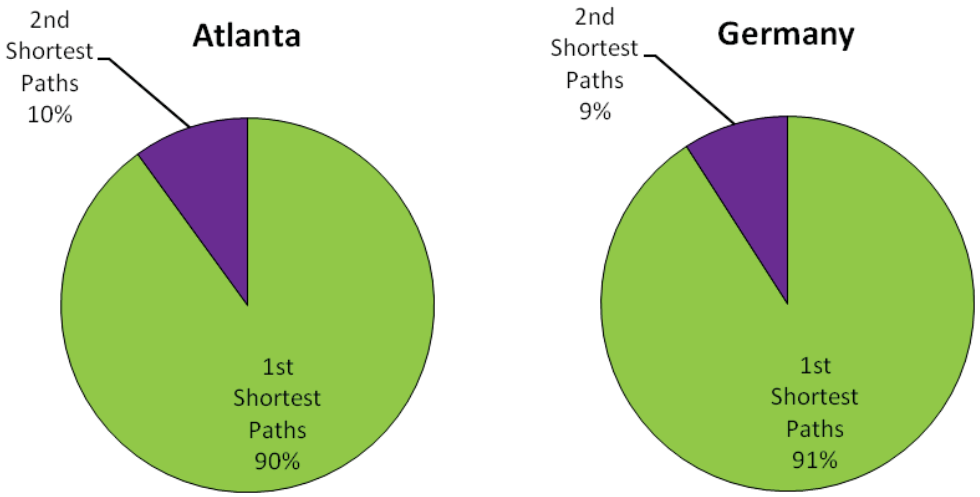


Figure 31 The proportion of the first and second shortest paths in the working path set for the NBR (and SNBR, which is the same) solution for Atlanta and German cases

Instead, the BR model is configured to allow bifurcated routing. Therefore, both the first and the second shortest paths have the possibility to become the final working paths.

Upon closer examination of the resulting working paths for the BR method, we computed the frequency for the first and the second shortest paths that turned out to be the final working paths for the two real networks, separately. The results are summarized in Figure 32. Nearly all the first shortest paths finally become the working paths, along with a portion of the second shortest paths, 24.0% and 30.9% respectively for the Atlanta and German cases.

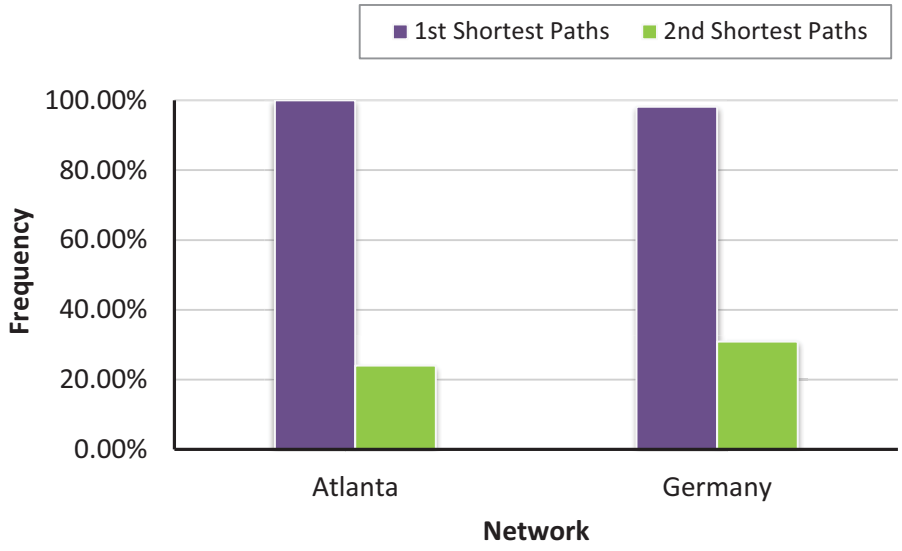


Figure 32 The composition of the final working path set for the BR model

Lastly, none of the problems were solved to optimality in the given time of 10 hours. It can be seen from Table 8 that the MIP gap for the BR and the NBR model is nearly half of that for SCP in the Atlanta instance. Indeed, the solution for SNBR is the closest to optimal, with a MIP gap of only 9.1%. Similar results can be obtained for the German network case as well. Moreover, the solver for the NBR model terminated with a gap of 23.1%, which is much less than that for the BR case with the German network. In summary, it is much easier for the solver to get the SNBR method to solve the problem to completion, and the NBR is a better choice than BR, from the standpoint of how long it takes to get to an optimal solution.

5.2.3 Changes with the Number of Eligible Shortest Cycles

While the previous experiment does demonstrate the pros and cons of our joint designs, it does not give information about the quality of solutions in the case that only part of the cycles are taken into account. In fact, the higher the connectivity of a network, the more cycles in the graph. The densest real test instance in our research is the COST 239 network, where the total number of cycles is extremely large. It would turn out to be a vain attempt to include all those cycles as eligible cycles. Also, the problem size is far beyond what our machine can cope with. Instead, as discussed in the above section, we selected the N-shortest cycles to be eligible in our test on the COST 239 network. The experiment was split into two trials. The metric used for the shortest cycles was their circumference for the first trial, and the number of hops for the second trial. Actually, 5 different groups of N-shortest cycles were generated for each trial, with the value of N ranging from 100 to 300. Those cycles were selected to be the candidates and input into the linear programs.

In both the trials, the cost cannot be improved further beyond a certain point. Figure 33 shows the total cost of network capacity for our MFIPP-JCA models, as well as for the SCP case, varying with the number of eligible shortest cycles for the first trial. The results for different models tend to decrease in a similar way, when more short cycles are given as the candidates in the tests. Relative to the corresponding cases with the 100-shortest cycles, the solutions obtained from using 150-shortest cycles are improved dramatically, by 6.1%, 7.5%, 10.2% 5.7% for the SCP, BR NBR and SNBR cases respectively. However, the trend in the decline of the total cost slows down remarkably at the point of 200-shortest cycles. The differences in the cost are within 1.2% between the cases of 200- and 250-shortest cycles, and never exceeds 1% in the comparisons of the 250 and 300 cases for all models. In the tests using hop counts as the metric, the total costs experience a reduction for all models as well, with an increasing number of shortest cycles, as shown in Figure 34. The decline as a whole is more gradual,

especially for the case of SCP, BR and NBR. When the number of shortest cycles changes from 250 to 300, the improvement in the total cost is rather low, not exceeding 0.7% for the case of SCP, BR and NBR. By contrast, adding 50 more shortest cycles to the 200 case still shows a reduction of more than 2% on the total cost for those models.

To some extent, the larger the value of N, the longer the eligible cycles are, regardless of the metric in effect. Longer eligible cycles have both positive and negative effects on the quality of solutions. The benefit is that they provide more opportunity to protect more paths all with a single cyclic pattern, which allows the exploration of a solution with a higher degree of sharing. However, the cost for a unit of capacity on the larger

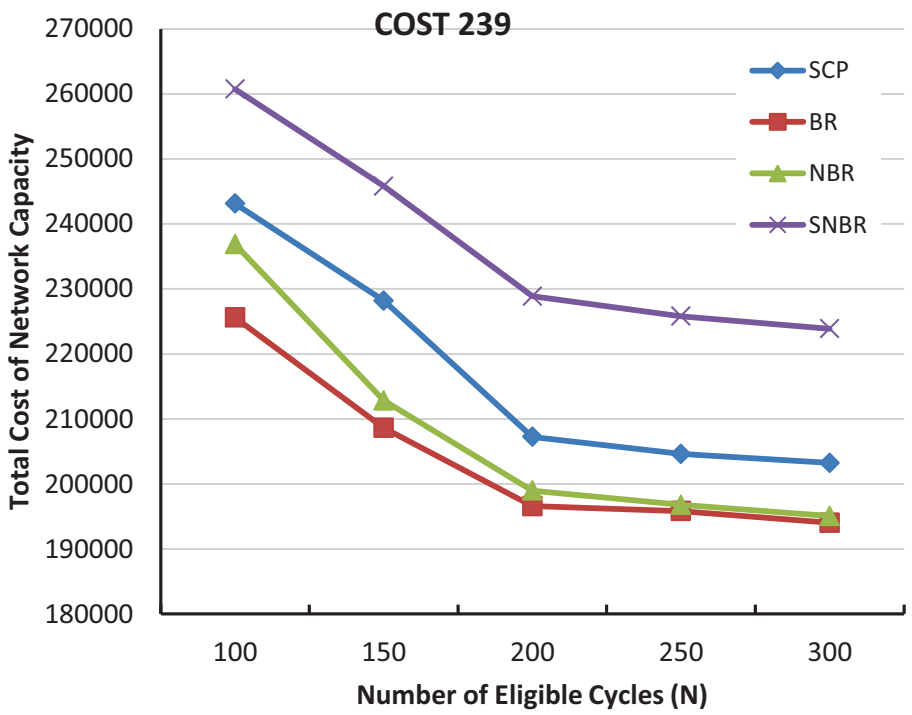


Figure 33 Total cost of network capacity versus different group of N-shortest cycles, using circumference as metric

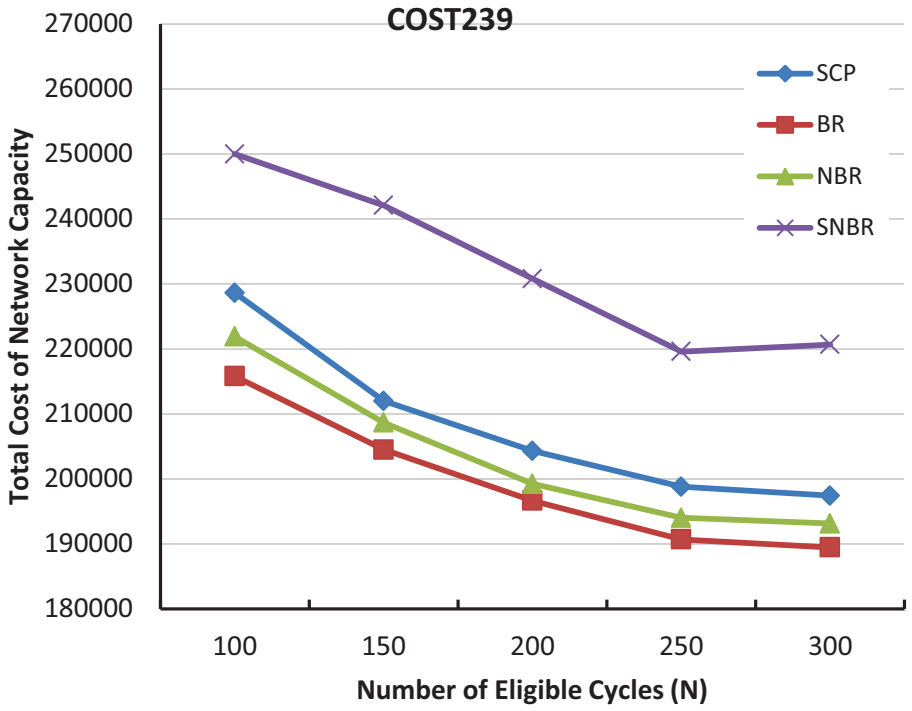


Figure 34 Total cost of network capacity versus different group of N-shortest cycles, using hop counts as metric

cyclic pattern is higher. At this point, the solution tends to be more costly when the larger cycles are involved in the problem. Therefore, the total cost of an FIPP *p*-cycle based network does not reduce continuously, as more shortest cycles become eligible. Similar to the behaviour of the span-protecting *p*-cycles that was reported in [41], FIPP *p*-cycled networks exhibit a threshold effect on the total cost in both the cases of circumference-limited and hop-limited requirements.

In most cases, the results from the tests on hop-based selection of cycles are superior to the corresponding circumference-based solutions. To illustrate, the differences between the total costs of the two trials are listed in percentage terms in Table 10.

Table 10 Comparison of the total costs between circumference-based and hop-based selection of cycles

N	Models			
	SCP	BR	NBR	SNBR
100	6.0%	4.3%	6.3%	4.1%
150	7.1%	2.0%	2.0%	1.5%
200	1.4%	0.0%	-0.1%	-0.8%
250	2.8%	2.6%	1.4%	2.7%
300	2.9%	2.3%	1.0%	1.4%

For example, the total cost for NBR, using 100-shortest cycles that are selected on a circumference basis, is 6.3% percent more than the corresponding solution obtained for the hop-basis case. The tests with the hop-based selection of cycles yield more

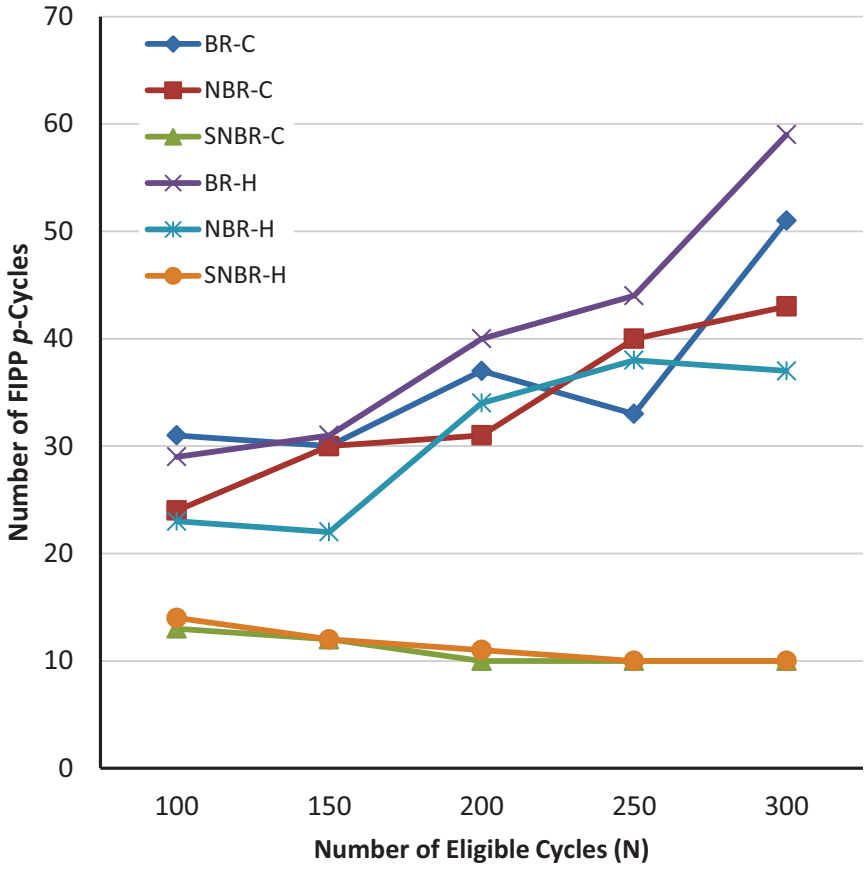


Figure 35 The number of FIPP *p*-cycles versus the N-shortest cycles for MFIPP-JCA designs in the two trials

cost-efficient solutions than the circumference-based ones for the SCP model. This is also true for the other models, exclusive of the cases in which the number of eligible cycles is 200. This leads us to the conclusion that the hop-based selection of the shortest cycles helps to reach a more economical solution for FIPP *p*-cycle protecting networks.

The change in the number of FIPP *p*-cycles is illustrated in Figure 35, for our joint designs. In the legend, the two characters after the hyphen, i.e., "C" and "H", are short for the circumference and the hop based selection of shortest cycles. In both trials, the number of FIPP *p*-cycles rises gradually with an increasing number of eligible cycles for the NBR solutions. In contrast, the rise for the BR method is more rapid. Especially when it comes to the case of 300 cycles, the increase (compared to the previous one) on the number of FIPP *p*-cycles reaches 54.5% and 34.1% in the two trials, respectively. More significantly, the NBR solutions, in most cases, generate fewer FIPP *p*-cycles than in the BR case. On the other hand, the results from SNBR are slightly reduced, and stay within the range of 10 to 15 in either trial. Again, SNBR considerably outperforms all the other models with respect to this performance. All of these conclusions conform to the observations made for the solutions considered with the other test networks.

5.3 Summary

As a starting point to testing, a detailed description of the experimental setup was given, followed by a discussion of the results. The following statements are supported by the results from both the artificial 9-node network family and our selection of realistic networks:

- ***Feasibility of solutions***

  Problems for joint optimization are more likely to be solved, whereas some of the instances for SCP turn out to be infeasible.

- ***Total cost of capacity***

  BR is shown to be the most cost-saving method for FIPP *p*-cycled resilient networks. Also, the costs for NBR are always less than those for the SCP model, with a maximum improvement of 16.4% in the 9-node network family and 7.1% in the Atlanta case. Otherwise, the SNBR solution tends to be costly relative to the SCP case, especially for real networks. Substantially, there is not a great deal of difference between the costs for the BR and NBR cases.

- ***Number of FIPP p-cycles***

  SNBR successfully yields an extremely small number of FIPP *p*-cycles, which greatly reduces both the computational and operational complexity. In most cases, the NBR solutions require fewer FIPP *p*-cycles than the corresponding case for SCP and BR.

- ***Solution time***

  Although none of the tests on real networks terminated completely, it can be predicted, according to the MIP gap of the solutions, that the problem for SNBR is likely to be solved to optimality in the shortest time, and NBR is superior to BR at this point, both of which outperform the time-consuming SCP method.

- ***Changes with the number of eligible shortest cycles***

  Two trials were undertaken using the circumference and the hop counts as the metric for selection of the shortest cycles, separately. In both cases, the decline in the total cost tended to reach a threshold, for the reason that large cycles have both a positive and a negative effect. Also, the number of FIPP *p*-cycles arises for both BR and NBR case, whereas it drops slowly and remains to be a very small value in the SNBR solutions. Finally, it is more likely to get a relatively cost-efficient solution for the hop-based selection of shortest cycles, than in the

case of circumference selected shortest cycles.

Chapter 6 FIPP *p*-Cycles Implementation in MPLS Networks

6.1 Introduction

It is known that most deployments of path recovery methods in MPLS network are quite simple, especially for path protection, because the backup path, also referred to as the secondary path, is another option for a working LSP. In protection methods, the backup path is pre-determined and can be pre-established as an explicitly routed LSP. (We have mentioned these in previous chapters.) As a path protection method, is it necessary to further discuss implementation issues for these FIPP *p*-cycles?

Before answering this question, let us suppose that we need to deploy FIPP *p*-cycles just as we do in other path protection schemes. This means we establish protection paths as backup LSPs. More specifically, the protecting FIPP *p*-cycle segment will be treated as the backup LSP for each primary LSP. But how can we assign the reserved capacity of a FIPP *p*-cycle among its segments which are the backup LSPs of different primary LSPs? In other words, how can these backup LSPs share the reserved capacity of the FIPP *p*-cycle which they cross over? Note that the bandwidth is simply reserved for each LSP. The high capacity efficiency of FIPP *p*-cycles comes precisely from the sharing structure and possible straddling protection relationships. So it is against what we expect of an FIPP *p*-cycle. Not to mention the scalability of such a deployment. The number of backup LSPs is proportional to the number of primary LSPs just as for other path recovery schemes.[8] Furthermore, when the primary LSP is performing a pure straddling function for its protecting *p*-cycle, there will be two backup LSPs for the sake of improving capacity efficiency. Thus, the number of backup LSPs for a particular

[8] See Section 4.3.2 for a detailed discussion on this issue.

primary LSP depends on the number of protecting *p*-cycles and the relationship between these *p*-cycles and the primary LSPs. It is not advantageous to reduce the number of backup LSPs at all.

To sum up, FIPP *p*-cycles cannot be implemented in such a way that regards the actual protecting segments as backup LSPs. Instead, it is acceptable to treat whole FIPP *p*-cycles as backup LSPs. Recall that in the MPLS context, FIPP *p*-cycles are constructed as a virtual structure, not on the basis of a unit-capacity structure. Since in our network design, FIPP *p*-cycles are modelled as bidirectional structures, it requires two backup LSPs, one for each direction to fulfil an FIPP *p*-cycle in an MPLS network. The backup LSP in an FIPP *p*-cycles based MPLS network will be called a cycle LSP in context, in order to reflect the nature of a loop and be distinguished from ordinary backup paths. So FIPP *p*-cycles here refer to as a cyclic protection structure, while the cycle LSP emphasises the unidirectional property. Unless specifically indicated, all LSPs mentioned in this thesis are explicitly routed.

6.2 Observations on Applying *p*-Cycles to IP Networks

As mentioned in Section 4.2.1, extending original span-protection *p*-cycles from the SONET or WDM context to a link-protecting counterpart in an IP context has been proposed and studied in [19, 32]. We examined their work carefully to draw on their ideas for our case. In both IP and MPLS environments, switching is undertaken by looking up entries in the corresponding routing tables, replacing the real optical switching in the physical layer. Virtual *p*-cycles, instead of unit-capacity *p*-cycles should be deployed in accordance with the characteristics of capacity in the logical layer. At both the IP and MPLS layers, there is no rigid distinction between working and backup capacity, compared to the case in the physical layer.

Meanwhile, we realize that there are still many differences in applying *p*-cycles in an IP network and an MPLS network. The reasons lie in the distinct switching mechanism in these two networks. IP is a protocol used for communicating data across a packet-switched internetwork, whereas MPLS networks are based on label switching. Though similar in some aspects, they are separate in essence. Traditional IP routing relies on the IP header to make forwarding decisions. By contrast, a head-end LSR pushes a label onto an IP packet, and the packet is routed along a particular LSP label interpretation and through label swapping. The actual routing decision for an IP packet is made only once, just at the head-end router. The subsequent routing is completely predefined by the label switching sequence which defines the LSP. Bandwidth is consumed only when traffic is being carried, but in MPLS, it is allocated on an LSP basis, not just for links as in pure IP networks. That is the reason we regard path-oriented recovery schemes as the top-ranked choice to take advantage of the splendid protection granularity.

In [19, 32], since taking pure IP networks into consideration, virtual *p*-cycles are realized with a set of routing table entries created with reserved or otherwise unallocated regular IP addresses. In addition, a novel "*p*-cycle packet" was introduced, which contains a *p*-cycle ID field, an original route cost field, a destination address field and, as payload, the original IP packet. IP packets affected by link or node failure are designed to be deflected onto a *p*-cycle, which has been assigned to protect the link or node. It is achieved by encapsulating the original IP packet in a "*p*-cycle packet" and re-entering the same routing. The following action is to look up the routing table for the encapsulation IP address (i.e., the destination address in *p*-cycle packet) and send the packet out through the matching port. The packet then travels through the *p*-cycle according to routing entries at other nodes that have been pre-established to define the logical *p*-cycle. Further handling depends on the failure type. For link restoration, only the router on the other side of the dead link is required to react, since it is the exit point

where the packet needs to leave the *p*-cycle. The node decapsulates the detoured packet and routes it normally towards the final destination. While on the contrary, it is more complicated in the case of a node failure. Every intermediate router has to test the packet, in order to determine whether it should remove the packet from the *p*-cycle by decapsulating it and resume its normal route, or it should continue to forward the packet along the *p*-cycle. The test is a significant process to detect the appropriate decapsulation node and avoid endless looping of the packet. Furthermore, it ensures that the packet will be returned to normal IP forwarding on an existing route that is closer to its final destination than its entry point into the *p*-cycle.

Setting aside the fact that an IP address (e.g. IPv4) is a scarce resource, as introduced above, the implementation of *p*-cycles in pure IP networks involves encapsulation and decapsulation of IP packets on relevant routers and additional tests at transit nodes in the case of a node failure. By contrast, Path-protecting *p*-cycles can be more smoothly integrated with current operations in MPLS networks. MPLS is essentially a means to form virtual circuit-like tunnels within an IP environment. Irrespective of the original motivation, it could not be more suitable to deploy path-protecting *p*-cycles. There is no need to introduce such a special "*p*-cycle packet", which encapsulates an affected IP packet. *P*-cycle IDs in the header of the "*p*-cycle packet" can be replaced by labels, while deflection of a packet into a *p*-cycle can be transferred in another way just by pushing a new label, which is attached to the corresponding *p*-cycle. The labelled packet is routed along pre-established backup LSPs, rather than being switched hop-by-hop. No extra test is required to be taken at the transit routers, since the remaining routing is already pre-defined and conforms to the existing label switching sequence. As a result, the original route cost has no effect anymore and can be removed. Suppose that the actual destination is informed of when a failure occurs, then it can take the responsibility to ensure that the restored traffic will be terminated at the right place, by swapping the next hop of the label, which related to the corresponding *p*-cycle, to the

LSR itself. Details about the operation are deferred to the next section.

6.3 Operation

The FIPP *p*-cycle based recovery method is classified as a protection scheme. It is based on a centralized algorithm and can be realized by offline tools. Therefore, lengthy solution times are not a big issue since the whole protection scheme is completely pre-determined. Further, it can work with the global default restoration mechanism in MPLS, whereby the failure is notified to the head-end LSR by means of RSVP and the routing protocol, which in turn recomputes a new path and finally re-signals the LSP along that new path. In this way, the FIPP *p*-cycles are envisaged as the "fast" part of a "fast plus slow" overall recovery process. FIPP *p*-cycles can deal with the immediate real-time traffic protection part of the problem, while the default global restoration will proceed to develop new optimal paths from the global view. But in the interval of real time before new paths take effect, FIPP *p*-cycles will be in use to prevent the loss of packet. When the slower procedure for path computation and establishment finishes, the affected flows will redirect to the new paths and the traffic on the FIPP *p*-cycle will automatically be discontinued. This makes the sub-optimal protection paths which FIPP *p*-cycles provide will be insignificant due to their being just a temporary state in the recovery process.

An important issue, but beyond the scope of our study, concerns the failure detection mechanism. Briefly, there are two families of failure detection mechanisms. One is to resort to lower layer failure notifications, whereas the other is based on hello protocols, in IGP, RSVP, for instance. As a global protection method, traffic recovery is performed on the head-end LSR in FIPP *p*-cycles scheme. Regardless of failure detection techniques in use, the fault detection time (plus fault notification time) is a key component of the total recovery time. It is especially difficult to keep them as short as

possible for a global recovery scheme, due to the propagation delay of a notification. Besides, in our FIPP *p*-cycles based resilient network, it requires the tail-end LSR to be informed of a failure occurrence as well, in that it regards cycle LSP, as opposed to the actual protecting segment on cycle, as a backup LSP tunnel. A recently proposed low-overhead hello mechanism, named Bidirectional Forwarding Detection (BFD)[42] seems to be more applicable in FIPP *p*-cycles based resilient networks. It can be applied to fast detect a data plane failure in the forwarding path of an MPLS LSP and can be used in conjunction with LSP Ping [43] to enhance the overall functionality.

In the following sub-sections, we take into account two separate modes of operation in terms of FIPP *p*-cycles implementation in MPLS networks.

6.3.1 General Mode

Each cycle LSP is pre-signalled (e.g. by RSVP) according to an offline computation and bound with a unique label. For cycle LSPs, loop detection should be suppressed, for the reason that they are cyclic by nature. The relationship between the primary LSP and its cycle LSP needs to be maintained at both the head-end LSR and the tail-end LSR. The information should include the label of the protecting cycle LSP, the next hop, and corresponding operations to perform on the packet's label stack. It can be stored as an ordinary NHLFE in forwarding tables. There is no need to introduce a new table to contain the protection relationship due to the excellent compatibility with the current switching mechanism in MPLS. If holding the protection information in a standalone table, after getting the affected FEC information, it should look up this new table, and retrieve the label of cycle LSP. However, it is the ILM table that designates the actual NHLFE, which is responsible for providing the new label, the next hop and the following label forwarding actions. Thus, there should be a special treatment to integrate these two tables together. So it is more straightforward to insert the FIPP *p*-cycle label into the NHLFE table as a second choice for that particular FEC, just as

other path recovery schemes do.

Once a failure occurs, both the head-end router and the tail-end router will be notified through any potential fast failure detection mechanisms. After gathering the FEC information about the affected LSP, the head-end LSR will choose another NHLFE from the set of NHLFEs predefined in the FTN for that FEC. This new NHLFE will trigger the operation to push the label of the protecting cycle LSP onto the label stack and forward the labelled packet according to the next hop attached to the label. Back to the protection scheme itself, there are two different protection relationships between an LSP and a candidate *p*-cycle. If the LSP is an on-cycle path with respect to the cycle, it is protected by only one corresponding cycle LSP. Otherwise, when the LSP is completely straddling over the cycle, it can make use of both cycle LSPs as backup LSPs. In the latter case, the restoration flows of affected demands have to be split over the two cycle LSPs. It can be achieved by having the ILM map a label to a set containing more than one NHLFE.

The packets will follow the cycle LSP which has established as a loop. All intermediate LSRs only make their forwarding decision by label switching accordingly. The label of incoming packets is mapped to a NHLFE through ILM table and then the packets are forwarded based on the new label in NHLFE. It is exactly the same way as label swapping works under normal conditions. No peculiar actions are required in these intermediate LSRs.

It is the tail-end LSR that has responsibility for completing the restoration. Thus it should be paid more attention. One operation should be done after the tail-end LSR detects or is notified of the failure in this protection scheme. It needs to map the incoming label of the cycle LSP to a new NHLFE whose next hop is the tail-end LSR itself. In such a way, the packet coming from the FIPP *p*-cycle will stop and the label

related to the cycle LSP will be popped from the stack as it is the top label on the stack. At this point, the affected traffic is successfully restored along the backup cycle LSP.

The following action on the egress LSR depends on the depth of the packet's label stack as for ordinary LSPs. If the stack is empty, the packet now is a native IP packet and should be forwarded according to the network layer address. Otherwise, there are remaining labels in the stack. The packet is still a labelled packet which should be handled in MPLS forwarding tables.

Penultimate hop popping can be realized in the restoration state, i.e. for packets restored on the cycle LSPs, with an additional notification sent from egress LSR to its upstream LSR. This signalling message tells the upstream LSR that it should pop the label of those packets that have arrived through the corresponding cycle LSP. Henceforth, the egress LSR will be allowed to do a single lookup.

6.3.2 TTL-Based Mode

Another possible way to implement the protection and restoration in FIPP *p*-cycles based schemes, is to utilize the time-to-live (TTL) field in an MPLS shim header. Please note that in our research we are considering MPLS over IP.

In MPLS/IP network, TTL in labelled packets is inherited completely under the same consideration for the IP layer. The processing of TTL for LSPs depends on the model which these LSPs conform to [5]. Generally, in the Pipe model, the packet will be assigned with a TTL value, which is independent of the TTL value in the IP header, whereas in the Uniform model, the incoming TTL value will be equal to the IP TTL value. Regardless of the actual value, as the packet traverses its LSP, each LSR decrements the TTL by one. Once it reaches zero, the packet is no longer forwarded.

Suppose we are dealing with LSPs in the Pipe model, when a labelled packet needs to be switched along such an LSP, the TTL field can be given the same value as the hop count of that LSP. We have mentioned that each cycle LSP is pre-determined and pre-signalled. The protection relationship between a primary LSP and its cycle LSPs is maintained at both the head-end LSR and tail-end LSR. Here, one more piece of data must be appended. That is the exact hop count of the actual recovery path. It is not difficult to achieve this, because protection is fully pre-planned and deterministic (due to the failure-independent property of FIPP *p*-cycles). The actions on the head-end LSR are nearly the same as in the previous plan. After detecting the failure and gaining the corresponding FEC information, the head-end LSR chooses a different NHLFE and triggers the operation to push the label of the protecting cycle LSP onto the label stack. At the same time, the hop count of the recovery path provided by the FIPP *p*-cycle will also be retrieved (e.g., from the NHLEF). Now, the packet is encapsulated in a new MPLS shim header, with a new label and TTL value, both are related to that particular protection path as a part of the cycle LSP. Finally, the labelled packet is well prepared and can be forwarded as usual.

As it traverses over the cycle LSP, each intermediate LSR checks the TTL value as usual. The TTL value received is reduced by one. If the result is larger than zero, label swapping will be taken for this labelled packet. If the outcome of the TTL check is false, (i.e., after decrement by one the resulting TTL becomes zero), the LSR will know that it is the egress-router for the traffic on cycle LSP. The top label will be popped afterwards. The forwarding action along cycle LSP ends at this node. Further procedures can be the same as those in the above plan.

With regard to Uniform model, it requires one more step at the egress LSR. After stopping forward actions, the tail end LSR needs to retrieve the hop count of the recovery path, and recalculate TTL value for the remaining MPLS or IP packet. Next, it

operates according to whether there is any label in the label stack or not. Details are the same as in the previous plan except for the handling of the TTL field.

6.4 Directional Issue for FIPP *p*-Cycles

In an MPLS network, every LSP has its own direction. Recall that the source router named ingress router or head end router in different literatures. Corresponding target router is called as egress router or tail end router. So no matter which kind of FIPP *p*-cycles we used in computation, only unidirectional paths (e.g., cycle LSPs) can be established as backup LSPs.

As a loop structure, the earliest span-protecting *p*-cycles[1] are conducted as bidirectional. Though people realize that it can be used as either unidirectional or bidirectional, there is quite a few research on unidirectional *p*-cycles. The main reason is due to the doubled amount of links and eligible cycles. It will produce more variables and constraints in LP models. The direction property of FIPP *p*-cycles in design phase is optional and has strong relation to the characteristics of network demands.

6.4.1 Bidirectional FIPP *p*-Cycles

The original FIPP *p*-cycles [20, 21] were designed as bidirectional. If demand is modelled as undirected, it implies that the links and routes we consider can be undirected too. More specifically, the demands between a pair of nodes will be routed over the same routes regardless of their actual transmission direction. The FIPP *p*-cycle selected to protect an end-to-end path thus is bidirectional. In MPLS, this requires two cycle LSPs, one for each direction, to fulfil an FIPP *p*-cycle. Obviously, there is no other choice but to assign the same capacity for these two LSPs as the total capacity need by the FIPP *p*-cycle. Henceforth, the total cost of backup capacity will be twice as much as the cost for capacitating the bidirectional FIPP *p*-cycles themselves. Degradation of

capacity efficiency cannot be avoided as a result of fitting the bi-directional FIPP *p*-cycles directly into unidirectional use.

In the case where demands are characterized as being directional, bidirectional FIPP *p*-cycles are still an option on the premise that the same route will be used for demands between a particular node pair. The key issue is how to allocate reserved capacity on each cycle LSP in order to achieve 100% guaranteed protection for every demand stream. Our resilient network design is made under an assumption of symmetric demand in an undirected graph. The demand volumes are given as the sum of the two demand streams required, one from node A to node B and the other from node B to node A respectively. To illustrate, a simple example is given below. Figure 36a shows two unidirectional demands (the yellow curves) both having a demand of 5 units between node A and B. In our study, since they are symmetric demands, they are merged together and protected as a single element, as depicted in Figure 36b.

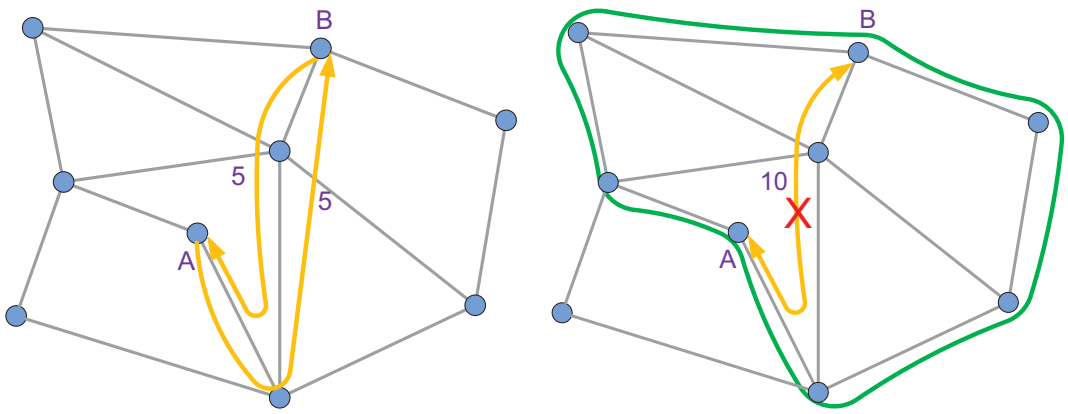


Figure 36 a) Two symmetric demands between a node pair
b) A bidirectional demand and its FIPP *p*-cycle

When a failure occurs along the path, the affected unidirectional demands will be protected by the corresponding cycle LSPs, as illustrated in Figure 37. The reserved capacity over each cycle LSP is just half of that on the corresponding FIPP *p*-cycle. Thus, the total cost of backup capacity is identical to that required by FIPP *p*-cycles, as modelled in our MIP problems.

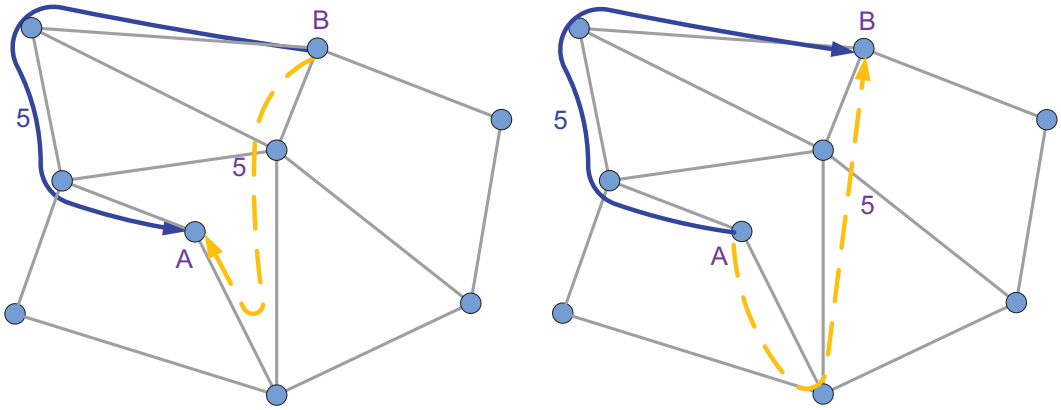


Figure 37 The portion of capacitated cycle LSPs response to each working path for symmetric demands

However, this is not the case for asymmetrical traffic. Suppose in the above example, the demand volume required from node A to node B and the one from node B to node A are not the same any more. Let us make them 9 and 1 respectively, for instance. See Figure 38.

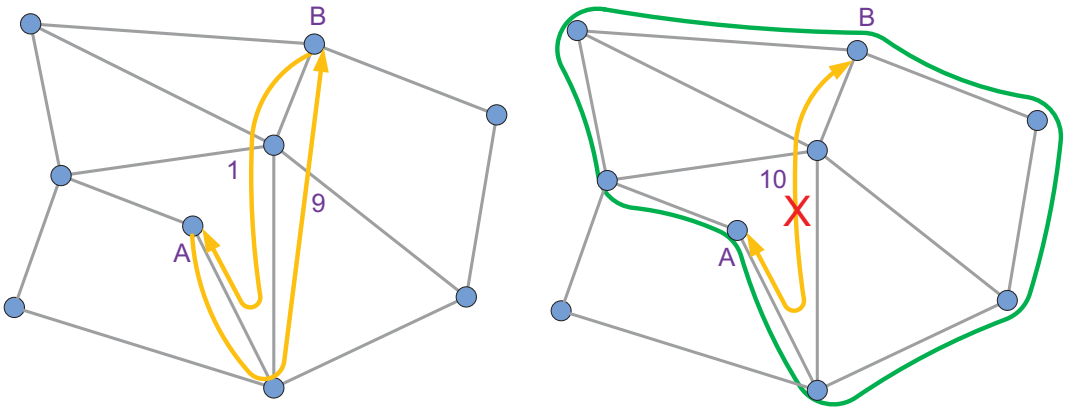


Figure 38 a) Asymmetric demands between a node pair
b) An undirected demand and its FIPP *p*-cycle

The resulting capacity for the FIPP *p*-cycle is still 10 as shown in Figure 38b and the required reserved bandwidth on cycle LSP1 and cycle LSP2 actually changes to 9 and 1 respectively. However, there is no way to know exactly how much capacity is needed exactly for the two anti-clockwise cycle LSPs, due to the bidirectional property of FIPP *p*-cycles and the merging operation on demand volumes. There is no alternative but to

capacitate the two cycle LSPs with the equal bandwidth as the reserved capacity for their corresponding FIPP *p*-cycle, e.g., 10 in this example, as shown in Figure 39. Definitely, the capacity allocated to one of the two cycle LSPs is wastefully used and incurs an unnecessary cost.

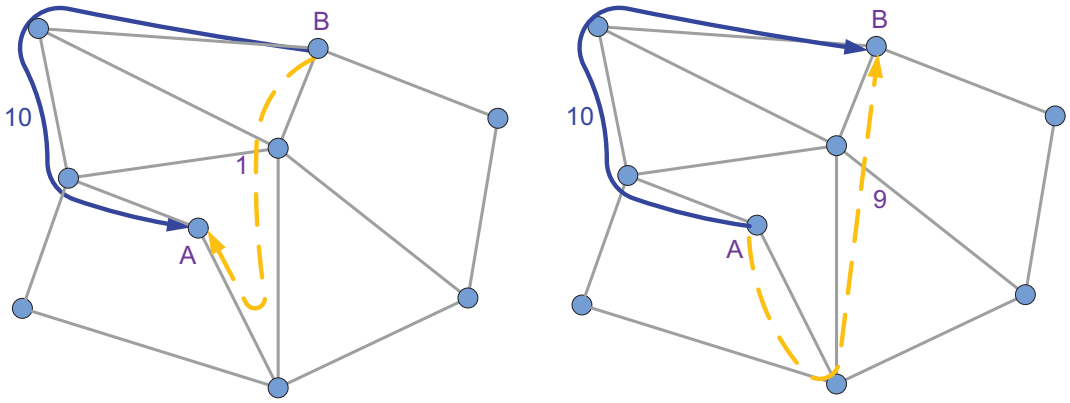


Figure 39 The portion of capacitated cycle LSPs response to each working path for asymmetric demands

6.4.2 Unidirectional FIPP *p*-Cycles

For directional demand patterns, unidirectional FIPP *p*-cycles are more straightforward and suitable. All links, paths and eligible cycles are considered to be directional as well. Instead of building two cycle LSPs for each FIPP *p*-cycle in bidirectional case, a unidirectional FIPP *p*-cycle calls for only one cycle LSP as the backup path for protecting the primary LSPs. Thus, there is a one-to-one relationship between unidirectional FIPP *p*-cycles and cycle LSPs. The cycle LSP is exactly the same as the unidirectional FIPP *p*-cycle itself, in terms of both the construction and the amount of reserved bandwidth. Protection is undertaken on the basis of a unidirectional demand stream, which has finer granularity relative to that in the case of bidirectional FIPP *p*-cycles.

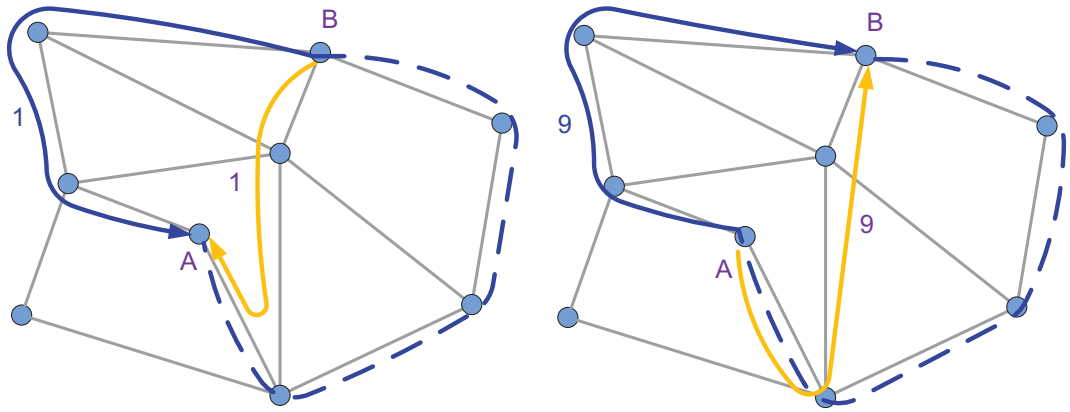


Figure 40 Unidirectional FIPP *p*-cycles for asymmetric demands

To illustrate, let us explain it step by step with an example, such as in the above case, where demands required between node A and node B are quite different. Since now we are directly dealing with unidirectional demands, the capacity allocation on corresponding cycle LSPs is fully determined by the protected demand alone. Therefore, despite being seemingly similar as depicted in Figure 39, the cycle LSPs in Figure 40 are independent and are designated with an irrelevant amount of capacity, which is decided just according to the volume of demand and the protection relationship. Furthermore, the demand stream from node B to node A and the one from node A to node B are now under protection from independent unidirectional FIPP *p*-cycles. This means that they can now be protected by FIPP *p*-cycles with distinct structures. For example, the FIPP *p*-cycle (i.e., the cycle LSP in this case) for demand from node A to node B can be a different one, as shown in Figure 41a below. Such a relaxation gives more freedom on selection of an optimal FIPP *p*-cycle, with the objective to minimize the cost of backup capacity. Last, but not least, since the demand is described as unidirectional, this means a single demand stream can be routed differently from the demand stream in the opposite direction. An example for the above case is shown in Figure 41b. Recall that in the bidirectional FIPP *p*-cycles based scheme, candidate paths are undirected and the demands between a pair of nodes will be routed over the same routes regardless of their direction.

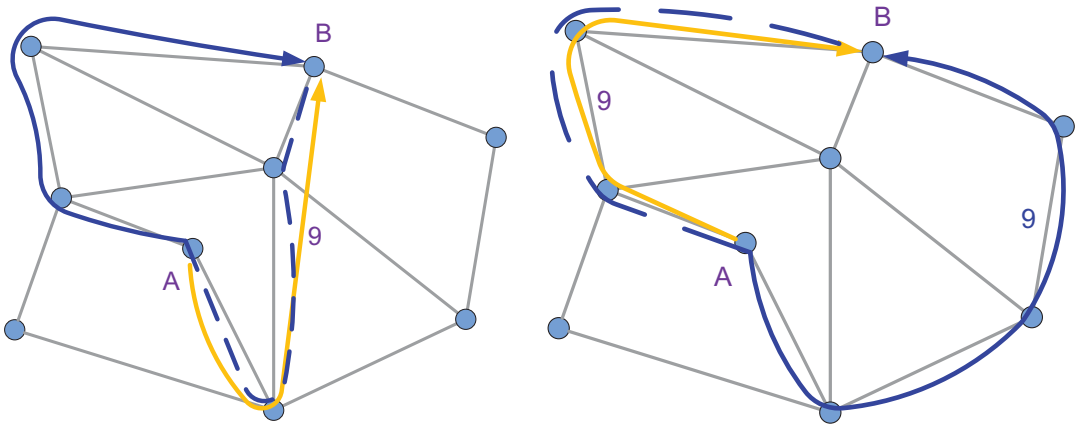


Figure 41 a) A different FIPP *p*-cycle in use b) A distinct path for primary LSP

By contrast, unidirectional FIPP *p*-cycles are capable of protecting a single demand stream from one node to the other. This allows the two demand bundles between a node pair to be carried along separate routes. Further improved capacity efficiency can be obtained due to the more flexible routing with accompanying shared protection. The symmetric characteristic of traffic is irrelevant to the concluding total cost of backup capacity, which is always the same as the cost of capacitating the unidirectional FIPP *p*-cycles entirely. Despite these advantages, a significant factor preventing it from being favoured is that there is a doubled number of links, paths, demand bundles and eligible cycles. The number of variables and constraints will be more than twice those in the bidirectional FIPP *p*-cycles counterpart. Extensive extra computational effort is inevitable - owing to the immense problem size. Nevertheless, it is still a potential and feasible protection method.

6.5 Summary

In this chapter, we focussed on potential implementation issues for FIPP *p*-cycles in an MPLS environment. First, we stated the differences and advantages of adapting *p*-cycles within an MPLS layer over the case in an IP layer in terms of implementation. Next, we proposed two possible modes of operation for FIPP *p*-cycles based protection schemes in an MPLS network. Finally, the difference between bidirectional and unidirectional

FIPP *p*-cycles was discussed using some simple examples, along with some general ideas about the advantages and disadvantages of each. Our network design is taken under the assumption of symmetric demand distributions and thus we prefer bidirectional FIPP *p*-cycles, in which case the problem size is relatively smaller.

Chapter 7 Conclusions and Future Work

7.1 Conclusions

This thesis set out to explore the use of *p*-cycle extensions for improving the reliability of MPLS networks and focussed on a recently introduced end-to-end path protecting scheme, known as FIPP *p*-cycles. Our investigation was carried out on the basis of bidirectional FIPP *p*-cycles.

FIPP *p*-cycles are an attractive protection mechanism due to their many excellent properties, including failure independence, pre-connection, end-node switching of recovery action, and high capacity efficiency. Until now, only two basic methods for FIPP *p*-cycles were available in the literature, originally denoted as FIPP-SCP and FIPP-DRS. In order to get further insight, the corresponding ILP models were studied, followed by a deep analysis of the theory behind these models. Although they were both developed for generating FIPP *p*-cycles designs, they are distinct in the way that path-disjointness under a cycle protection is established. In fact, the SCP solution is an exact reflection of the stringent disjointness requirements, either from the cyclic pattern view, or referring to a unit of capacity on the cyclic structure to be an FIPP *p*-cycle. Instead, the DRS method allows non-disjoint paths to be protected with a common cyclical pattern. The difference may be not a significant for optical network design problems, but should certainly be considered in the context of MPLS.

Attempts to extend the FIPP-SCP method to a joint network design for MPLS networks gave rise to a series of MFIPP-JCA models. Tests were carried out with both artificial networks and realistic samples. The jointly optimized problems are more likely to obtain a feasible solution, as opposed to those that only consider spare capacity allocation. The parallel (feasible) solutions from BR, NBR and SNBR were compared

with one another, as well as with the SCP case, in several aspects including total costs of capacity, the number of FIPP *p*-cycles in use, and the solution time. The performances for BR and NBR are close to each other, and both superior to that for SCP. To be precise, the NBR method is the parallel joint network design for the basic SCP model and the cost reduction can be completely attributed to its joint nature. Generally, NBR is a better choice, for the reason that there is a single working path for every demand, which gives the benefit of achieving a higher quality of service. Rather, it should be borne in mind that the corresponding solution for bifurcated routing is capable of providing a further reduction in the total cost, and is suitable as a measure in comparative studies. As for SNBR, the fact that it produces a fairly small number of FIPP *p*-cycles brings it to the top of our concern with respect to operational efforts. Moreover, it can be used to offer an upper bound on the capacity cost for an FIPP *p*-cycles protected network design, as the solver finds it easier to arrive at the optimal solution in a relatively short time. All three MFIPP-JCA models can be applied to network design problems, in which only N-shortest cycles are permitted to be eligible. Additionally, they do exhibit a trend towards reaching a cost threshold, with an increase in the number of cycles. In general, hop-based selection of shortest cycles has an advantage over the circumference-based one, in terms of the total cost. Another important observation is that the resulting number of FIPP *p*-cycles goes up accordingly for the BR and the NBR solutions, whereas in SNBR cases, it continues to be a fairly small value and even shows a slight decline with the rise in candidate cycles.

Lastly, two major issues regarding the implementation of the FIPP *p*-cycles in MPLS networks were discussed. Firstly, two options for the operational mode were proposed. One is a general solution, and the other is based on the particular TTL field of an MPLS shim header. Unidirectional FIPP *p*-cycles were not selected for our research due to the exhausting and unmanageable problem size. However, it is a potential method to further enhance the quality of solutions, especially the total cost of capacity.

7.2 Future Work

There are a lot of options for future research. Seeking an efficient heuristic method for FIPP *p*-cycles should be uppermost. Though the FIPP-DRS method is a successful approach, one should keep in mind that the paths that are protected by a common cyclic pattern are no longer required to be disjoint from each other.

In Section 4.3, we analyzed the pros and cons of FIPP *p*-cycles as an MPLS-based recovery method, along with a comparison of the network scalability with other network survivability schemes that had previously been presented. There is a need to make an in-depth comparison of those schemes, and illustrate them with practical experiments.

We investigated the impact of using N-shortest cycles as candidates for the performance of our joint network designs. In fact, all the cycles in the graph are still required to be enumerated one by one first, and given as the input to a bubble sort process in order to generate the N-shortest cycles. Future research can be done to find more efficient ways for the pre-selection of cycles, along with the goal of maintaining the quality of solutions.

According to the results in Section 5.2, there is a trade-off between the total cost of capacity and the number of FIPP *p*-cycles required in a resilient network design. We suggest a bi-criteria study on FIPP *p*-cycle based network design problems. The idea is to develop a model with the dual aims that are of concern and the importance of which can be adjusted by a weighting factor. The motivation is to improve the capacity efficiency with very small or no penalty on the final number of FIPP *p*-cycles.

Another area to consider is the extension of FIPP *p*-cycle designs to a network supporting differential services, such as data, voice and video. Henceforth, it is more suitable to employ a multiple quality of protection framework. The bandwidth

protection can be allowed on per-class basis, and applied in various degrees for services according to their required QoS. For example, the traffic for a real time service deserves full bandwidth protection, whereas for data traffic, offering protection to a portion of the bandwidth may be more reasonable. All the MFIPP-JCA models can be extended to take the multiple services into account.

Appendix A Test Networks

A.1 Atlanta Network

Topology

Nodes:

Node	X	Y
N1	283	248
N2	451	201
N3	516	230
N4	324	43
N5	459	31
N6	339	151
N7	185	323
N8	384	298
N9	318	353
N10	147	403
N11	56	132
N12	253	466
N13	216	59
N14	97	218
N15	463	431

Links:

Link	Source	Target	Length
S1	N1	N6	112
S2	N1	N7	123
S3	N1	N8	113
S4	N2	N3	71
S5	N2	N5	170
S6	N2	N6	123
S7	N3	N5	207
S8	N3	N8	148
S9	N4	N5	136
S10	N4	N6	109
S11	N6	N13	154
S12	N7	N10	89
S13	N7	N14	137
S14	N8	N9	86
S15	N8	N15	155
S16	N9	N10	178
S17	N9	N12	130
S18	N9	N15	165
S19	N10	N12	123
S20	N11	N13	176
S21	N11	N14	95
S22	N13	N14	199

Demand Pattern

Demand	Source	Target	Value
D1	N1	N2	131
D2	N1	N3	18
D3	N1	N4	12
D4	N1	N5	9
D5	N1	N6	35
D6	N1	N7	10
D7	N1	N8	13
D8	N1	N9	11
D9	N1	N10	12
D10	N1	N11	6
D11	N1	N12	15
D12	N1	N13	7
D13	N1	N14	34
D14	N1	N15	15
D15	N2	N3	58
D16	N2	N4	49
D17	N2	N5	17
D18	N2	N6	134
D19	N2	N7	40
D20	N2	N8	108
D21	N2	N9	6
D22	N2	N10	43
D23	N2	N11	17
D24	N2	N12	24
D25	N2	N13	12
D26	N2	N14	9
D27	N2	N15	41
D28	N3	N4	7
D29	N3	N6	6
D30	N3	N8	25
D31	N3	N15	9
D32	N4	N5	22
D33	N4	N6	20
D34	N4	N13	33
D35	N6	N8	19
D36	N6	N10	13
D37	N6	N15	6
D38	N7	N9	6
D39	N8	N15	11
D40	N9	N12	6
D41	N9	N15	6
D42	N10	N11	9
D43	N10	N12	26
D44	N10	N13	20
D45	N10	N14	25
D46	N10	N15	15
D47	N11	N13	13
D48	N11	N14	8
D49	N12	N15	21
D50	N13	N14	7

A.2 German Network

Topology

Nodes:

Node	X	Y
N1	13.48	52.52
N2	8.8	53.08
N3	7.48	51.51
N4	6.78	51.22
N5	7	51.44
N6	8.66	50.14
N7	10.08	53.55
N8	9.8	52.39
N9	8.41	49.01
N10	7.01	50.92
N11	12.38	51.34
N12	8.49	49.49
N13	11.55	48.15
N14	7.21	53.6
N15	11.08	49.45
N16	9.12	48.73
N17	9.99	48.4

Links:

Link	Source	Target	Length
S1	N1	N7	306
S2	N1	N8	298
S3	N1	N11	174
S4	N2	N7	114
S5	N2	N8	120
S6	N2	N14	144
S7	N3	N5	37
S8	N3	N8	208
S9	N3	N10	88
S10	N3	N14	278
S11	N4	N5	36
S12	N4	N10	41
S13	N6	N8	316
S14	N6	N10	182
S15	N6	N11	353
S16	N6	N12	85
S17	N6	N15	224
S18	N7	N8	157
S19	N8	N11	258
S20	N9	N12	64
S21	N9	N16	74
S22	N11	N15	275
S23	N13	N15	179
S24	N13	N17	143
S25	N15	N16	187
S26	N16	N17	86

Demand Pattern

Demand	Source	Target	Value
D1	N1	N3	8
D2	N1	N4	9
D3	N1	N6	29
D4	N1	N7	12
D5	N1	N8	14
D6	N1	N10	9
D7	N1	N11	22
D8	N1	N13	10
D9	N1	N15	8
D10	N1	N16	11
D11	N2	N6	13
D12	N2	N11	8
D13	N3	N4	8
D14	N3	N6	19
D15	N3	N8	9
D16	N3	N10	8
D17	N3	N11	11
D18	N4	N5	8
D19	N4	N6	22
D20	N4	N7	8
D21	N4	N8	9
D22	N4	N10	12
D23	N4	N11	13
D24	N4	N16	8
D25	N5	N6	16
D26	N5	N11	9
D27	N6	N7	26
D28	N6	N8	29
D29	N6	N10	24
D30	N6	N11	41
D31	N6	N12	14
D32	N6	N13	23
D33	N6	N14	144
D34	N6	N15	19
D35	N6	N16	29
D36	N6	N17	17
D37	N7	N8	14
D38	N7	N10	9
D39	N7	N11	17
D40	N7	N13	9
D41	N7	N16	10
D42	N8	N10	10
D43	N8	N11	19
D44	N8	N13	9
D45	N8	N15	8
D46	N8	N16	11
D47	N10	N11	13
D48	N10	N16	9
D49	N11	N13	14
D50	N11	N15	12
D51	N11	N16	17
D52	N11	N17	10
D53	N13	N16	10
D54	N15	N16	8
D55	N16	N17	9

A.3 COST 239

Topology

Nodes:

Node	X	Y
N1	140	281
N2	315	350
N3	304	298
N4	356	235
N5	447	308
N6	417	161
N7	242	159
N8	250	214
N9	185	208
N10	128	143
N11	344	50

Links:

Link	Source	Target	Length
S1	N1	N2	820
S2	N1	N3	600
S3	N1	N6	1090
S4	N1	N8	400
S5	N1	N9	300
S6	N1	N10	450
S7	N2	N3	320
S8	N2	N5	820
S9	N2	N9	930
S10	N3	N4	565
S11	N3	N5	730
S12	N3	N8	350
S13	N4	N5	320
S14	N4	N6	340
S15	N4	N8	730
S16	N4	N11	740
S17	N5	N6	660
S18	N6	N7	660
S19	N6	N11	390
S20	N7	N8	390
S21	N7	N9	210
S22	N7	N10	550
S23	N7	N11	760
S24	N8	N9	220
S25	N9	N10	390
S26	N10	N11	1310

Demand Pattern

Demand	Source	Target	Value
D1	N1	N2	5
D2	N1	N3	6
D3	N1	N4	1
D4	N1	N5	2
D5	N1	N6	11
D6	N1	N7	5
D7	N1	N8	1
D8	N1	N9	7
D9	N1	N10	10
D10	N1	N11	1
D11	N2	N3	6
D12	N2	N4	1
D13	N2	N5	3
D14	N2	N6	9
D15	N2	N7	2
D16	N2	N8	1
D17	N2	N9	2
D18	N2	N10	3
D19	N3	N4	1
D20	N3	N5	3
D21	N3	N6	11
D22	N3	N7	3
D23	N3	N8	1
D24	N3	N9	6
D25	N3	N10	3
D26	N3	N11	1
D27	N4	N5	1
D28	N4	N6	2
D29	N4	N7	1
D30	N4	N8	1
D31	N4	N9	1
D32	N4	N10	1
D33	N4	N11	1
D34	N5	N6	9
D35	N5	N7	1
D36	N5	N8	1
D37	N5	N9	1
D38	N5	N11	1
D39	N6	N7	8
D40	N6	N8	2
D41	N6	N9	6
D42	N6	N10	8
D43	N6	N11	3
D44	N7	N8	1
D45	N7	N9	4
D46	N7	N10	5
D47	N7	N11	1
D48	N8	N9	1
D49	N8	N10	1
D50	N8	N11	1
D51	N9	N10	4
D52	N9	N11	1
D53	N10	N11	1

A.4 Six Sets of Demands for the 9-Node Network Family

Demand sets

Demand	Source	Target	Value					
			(0,10)	(1,9)	(2,8)	(3,7)	(4,6)	(5,5)
D1	N1	N2	4	3	3	7	4	5
D2	N1	N3	1	2	6	6	5	5
D3	N1	N4	6	7	2	4	4	5
D4	N1	N5	9	3	7	6	5	5
D5	N1	N6	9	1	4	5	6	5
D6	N1	N7	10	9	8	4	5	5
D7	N1	N8	0	8	5	6	4	5
D8	N1	N9	4	2	5	5	6	5
D9	N2	N3	9	7	5	7	4	5
D10	N2	N4	1	8	5	6	5	5
D11	N2	N5	2	8	6	4	6	5
D12	N2	N6	0	3	6	3	5	5
D13	N2	N7	0	4	5	3	4	5
D14	N2	N8	5	3	5	5	6	5
D15	N2	N9	2	5	8	3	5	5
D16	N3	N4	0	8	8	5	4	5
D17	N3	N5	3	1	4	4	6	5
D18	N3	N6	3	1	3	7	5	5
D19	N3	N7	6	9	6	3	6	5
D20	N3	N8	4	8	8	3	5	5
D21	N3	N9	7	6	3	5	5	5
D22	N4	N5	4	4	5	6	4	5
D23	N4	N6	9	1	3	5	6	5
D24	N4	N7	5	5	6	4	6	5
D25	N4	N8	4	3	2	3	5	5
D26	N4	N9	3	5	3	6	5	5
D27	N5	N6	10	8	6	5	4	5
D28	N5	N7	8	5	6	6	5	5
D29	N5	N8	10	8	5	7	6	5
D30	N5	N9	3	3	2	7	6	5
D31	N6	N7	10	6	5	3	5	5
D32	N6	N8	1	7	6	6	4	5
D33	N6	N9	7	4	4	4	5	5
D34	N7	N8	8	3	3	6	5	5
D35	N7	N9	10	5	6	3	6	5
D36	N8	N9	5	8	7	5	4	5

Appendix B Result Tables

B.1 9-Node Network Family

Network	n9-1	"3*3 Grid"			TotalCycles	13						
Demand	WorkCost				SpareCost				TotalCost			
	FIPP	BR	NBR	SNBR	FIPP	BR	NBR	SNBR	FIPP	BR	NBR	SNBR
(0,10)	339	339	339	339	356	266	277	318	695	605	616	657
(1,9)	363	365	365	363		302	308	336		667	673	699
(2,8)	375	377	375	381		310	316	327		687	691	708
(3,7)	338	342	338	338		252	266	282		594	604	620
(4,6)	367	367	367	367		282	288	292		649	655	659
(5,5)	360	360	360	360		270	270	270		630	630	630

Network	n9-2	"K4,5"			TotalCycles	14						
Demand	WorkCost				SpareCost				TotalCost			
	FIPP	BR	NBR	SNBR	FIPP	BR	NBR	SNBR	FIPP	BR	NBR	SNBR
(0,10)	333	334.3	335	333	348.5	242	250	292	681.5	576.3	585	625
(1,9)	345	345.8	345	345	312	232.5	233	284	657	578.3	578	629
(2,8)	350	352	350	350	354	242	248	288	704	594	598	638
(3,7)	348	348	348	348	362	248	252	276	710	596	600	624
(4,6)	349	349	349	349	356	248	248	262	705	597	597	611
(5,5)	350	360	360	350	350	222.5	225	240	700	582.5	585	590

Network	n9-3	"Tri_Hex"			TotalCycles	14						
Demand	WorkCost				SpareCost				TotalCost			
	FIPP	BR	NBR	SNBR	FIPP	BR	NBR	SNBR	FIPP	BR	NBR	SNBR
(0,10)	349	357.5	353	361	363	259.5	278	295	712	617	631	656
(1,9)	340	341	341	344	343	281	275.5	309	683	622	616.5	653
(2,8)	340	346.5	345	340	291	249.5	255	292	631	596	600	632
(3,7)	318	325	323	318	268	216	220	247	586	541	543	565
(4,6)	337	338	337	337	298	232	234	246	635	570	571	583
(5,5)	330	330	330	330	280	222.5	225	225	610	552.5	555	555

Network	n9-4	"Cycle_Edge"	TotalCycles	15							
Demand	WorkCost				SpareCost				TotalCost			
	FIPP	BR	NBR	SNBR	FIPP	BR	NBR	SNBR	FIPP	BR	NBR	SNBR
(0,10)	314	332.5	326	323	287	211.5	226	269	601	544	552	592
(1,9)	314	318.5	316	319	214	201.5	206	249	528	520	522	568
(2,8)	311	313.5	311	311	213	203.5	210.3	253	524	517	521.3	564
(3,7)	309	313.5	312	309	238	199.5	199	238	547	513	511	547
(4,6)	323	325.5	323	323	242	210	206	227	565	535.5	529	550
(5,5)	315	315	315	315	230	195	195	210	545	510	510	525

Network	n9-1	"3*3 Grid"	TotalCycles	13							
Demand	NumCycles				MIPGap				SolveTime			
	FIPP	BR	NBR	SNBR	FIPP	BR	NBR	SNBR	FIPP	BR	NBR	SNBR
(0,10)	11	13	10	8	0.00	4.48	0.01	0.01	16	36000	8364	42
(1,9)		11	12	9		2.23	0.01	0.01		36000	28545	472
(2,8)		12	12	8		0.01	0.01	0.01		28265	10646	94
(3,7)		12	10	8		4.76	4.97	0.01		36000	36000	473
(4,6)		12	10	8		3.80	4.69	0.01		36000	36000	279
(5,5)		8	8	8		3.49	5.69	0.01		36000	36000	71

Network	n9-2	"K4,5"	TotalCycles	14							
Demand	NumCycles				MIPGap				SolveTime			
	FIPP	BR	NBR	SNBR	FIPP	BR	NBR	SNBR	FIPP	BR	NBR	SNBR
(0,10)	14	13	11	7	0.01	8.50	8.42	0.01	40	36000	36000	93
(1,9)	12	12	12	8	0.01	3.83	0.71	0.01	105	36000	36000	136
(2,8)	13	13	11	8	0.00	16.75	9.48	0.01	94	36000	36000	1256
(3,7)	12	11	9	7	0.01	9.01	5.74	0.01	21	36000	36000	188
(4,6)	14	9	9	7	0.01	10.50	9.27	0.01	24	36000	36000	286
(5,5)	12	10	8	7	0.01	11.03	8.55	0.00	43	36000	36000	6

Network	n9-3	"Tri_Hex"	TotalCycles	14							
Demand	NumCycles				MIPGap				SolveTime			
	FIPP	BR	NBR	SNBR	FIPP	BR	NBR	SNBR	FIPP	BR	NBR	SNBR
(0,10)	9	9	10	7	0.01	4.91	0.01	0.01	15	36000	19232	30
(1,9)	11	11	11	8	0.01	15.10	8.33	0.01	60	36000	36000	296
(2,8)	12	12	12	6	0.01	17.13	10.72	0.01	205	36000	36000	646
(3,7)	11	10	11	6	0.00	12.69	6.78	0.01	50	36000	36000	854
(4,6)	9	10	9	6	0.01	13.93	8.14	0.01	41	36000	36000	119
(5,5)	8	10	6	6	0.00	15.11	8.67	0.00	38	36000	36000	13

Network	n9-4	"Cycle_Edge"	TotalCycles	15							
Demand	NumCycles				MIPGap				SolveTime			
	FIPP	BR	NBR	SNBR	FIPP	BR	NBR	SNBR	FIPP	BR	NBR	SNBR
(0,10)	8	10	10	7	0.01	13.63	6.15	0.01	97	36000	36000	81
(1,9)	11	11	11	7	0.01	14.71	8.72	0.01	3426	36000	36000	244
(2,8)	10	10	10	6	0.01	18.56	12.56	0.01	15459	36000	36000	794
(3,7)	10	10	9	6	0.01	14.58	8.37	0.01	1446	36000	36000	801
(4,6)	12	10	12	6	0.00	15.76	9.73	0.01	1343	36000	36000	568
(5,5)	8	8	8	6	0.01	15.22	9.82	0.01	842	36000	36000	203

B.2 COST 239

Model	SCP		Metric	Circumference	
N	WorkCost	SpareCost	TotalCost	NumCycles	MIPGAP
100	132235	110888	243123	26	4.64
150	132235	95962	228197	32	18.45
200	132235	75014	207249	50	18.79
250	132235	72374	204609	51	20.92
300	132235	71005	203240	56	25.87

Model	BR		Metric	Circumference	
N	WorkCost	SpareCost	TotalCost	NumCycles	MIPGAP
100	141653	83953	225606	31	5.76
150	139643	68968	208611	30	8.34
200	137759	58839	196598	37	12.07
250	139011	56811	195822	33	12.48
300	138307	55721	194028	51	13.33

Model	NBR		Metric	Circumference	
N	WorkCost	SpareCost	TotalCost	NumCycles	MIPGAP
100	133725	103149	236874	24	2.43
150	139110	73715	212825	30	5.66
200	135135	63846	198981	31	6.15
250	134820	61961	196781	40	6.71
300	134960	60094	195054	43	7.31

Model	SNBR		Metric	Circumference	
N	**WorkCost**	**SpareCost**	**TotalCost**	**NumCycles**	**MIPGAP**
100	147795	112943	260738	13	0.01
150	149800	96005	245805	12	0.18
200	135470	93398	228868	10	0.65
250	140700	85098	225798	10	7.48
300	133125	90758	223883	10	7.78

Model	SCP		Metric	Hop counts	
N	**WorkCost**	**SpareCost**	**TotalCost**	**NumCycles**	**MIPGAP**
100	132235	96390	228625	36	5.86
150	132235	79770	212005	42	8.38
200	132235	72078	204313	62	24.38
250	132235	66565	198800	47	25.4
300	132235	65193	197428	69	26.22

Model	BR		Metric	Hop counts	
N	**WorkCost**	**SpareCost**	**TotalCost**	**NumCycles**	**MIPGAP**
100	141254	74549	215803	29	7.42
150	139819	64705	204524	31	9.99
200	140957	55694	196651	40	10.82
250	140209	50509	190718	44	10.46
300	138440	51045	189485	59	10.44

Model	NBR		Metric	Hop counts	
N	**WorkCost**	**SpareCost**	**TotalCost**	**NumCycles**	**MIPGAP**
100	135630	86270	221900	23	2.9
150	133645	75029	208674	22	4.81
200	133150	66092	199242	34	6.45
250	134905	59130	194035	38	8.26
300	135260	57873	193133	37	8.72

Model	SNBR		Metric	Hop counts	
N	**WorkCost**	**SpareCost**	**TotalCost**	**NumCycles**	**MIPGAP**
100	140065	109930	249995	14	0.01
150	139365	102713	242078	12	0.01
200	136675	94120	230795	11	2.16
250	140275	79320	219595	10	3.4
300	140060	80615	220675	10	7.3

Appendix C A Brief Description of Code Developed during the Study

We developed three programs to automatically generate AMPL data files for FIPP-SCP and MFIPP-JCA models. The relationship between the program and the model is shown as follows:

Program	Model
fipp-scp	FIPP-SCP
mfipp-jca	BR and NBR
mfipp-jca3	SNBR

Each program contains 7 source files, including 3 header files, 3 implementation files and a main file. The functions of these files are summarized in the following tables. Note that codes for main.cpp are different in the programs that serve for different models, but the program for the BR and NBR models are identical.

File	Description
Init.h	• Read two files that contain the information of the topology and demand pattern for the test instance. The corresponding file names must be *.top and *.dem. (An example for these files is given later in this Appendix.) • Initialize a *Network* object with the given nodes, links and demands.
Path.h	• Define the struct of *Node* and *Link*, the class of *Path* and *Cycle* (a derived class of Path), along with their member function prototypes.
Path_Imp.cpp	• Implements the functions for class *Path* and *Cycle*. The member functions of class *Path*, includes:

	1. Add and delete one hop on a path; 2. Display the information about a path; 3. Check if the portion of two paths matches up to a given node; 4. Create a sub-path between a given pair of node along the current path; 5. Basic operations: <=, <, >, >=, +; 6. Check if both of the nodes and links on the current path are disjoint from those on another path. The unique member functions of class ***Cycle*** to support the identification of z-cases, includes: 1. Get a node position on the cycle; 2. Separate a cycle into left and right segments; 3. Decide to which segment a link belongs.
Network.h	• Define the struct of ***Demand***, the class of ***Network*** and the member function prototypes for ***Network***.
Network_Imp.cpp	• Implements the member functions for class ***Network***, including: 1. Get the identification details of a node or a link; 2. Add or delete a node or a link; 3. Display the information about a network; 4. Reset the status of the network resources; 5. Generate the shortest path using the Dijkstra algorithm, the K-shortest path, and all possible paths; 6. Generate N-shortest cycles, and all simple cycles.
main.cpp	• The only file which is specific to a program and the corresponding model(s). • Provides the user interface for obtaining the input file names (and any unspecified data). • Output an AMPL data file that includes the data of sets and parameters for the relevant model.
Param.cpp	• Appends some complicated parameters to the AMPL data file, including:

	1. Check the disjointness of two paths; 2. Find out the subset of eligible cycles that are capable of protecting a particular demand relation; 3. Get the protection relationship between paths and cycles.

Examples of *.top and *.dem files are presented below for the small network with 4 nodes and 5 links, which was used for the discussion in Section 3.4.

n4s5.top

```
NODES:
1   N1
2   N2
3   N3
4   N4

SPANS:
N1   N2   1
N1   N4   1
N2   N3   1
N2   N4   1
N3   N4   1
```

n4s5.dem

```
DEMANDS:
N1   N2   2
N1   N3   2
N1   N4   2
N2   N3   2
N2   N4   2
N3   N4   2
```

References

[1] W. D. Grover and D. Stamatelakis, "Cycle-Oriented Distributed Preconfiguration: Ring-like Speed with Mesh-like Capacity for Self-Planning Network Restoration," in *Proc. of IEEE Int. Conf. Communication (ICC)*, Atlanta, GA, Jun. 7-11, 1998, pp. 537-543.

[2] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," IETF RFC 3031, 2001.

[3] I. Minei and J. Lucek, *MPLS-Enabled Applications: emerging developpments and new technologies*, 2nd ed. Chichester, West Suussex, England: John Wiley & Sons, Ltd., 2008.

[4] F. Le Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, and J. Heinanen, "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services," IETF RFC 3270, 2002.

[5] P. Agarwal and B. Akyol, "Time to Live (TTL) Processing in Multi-Protocol Label Switching (MPLS) Networks," IETF RFC 3443, 2003.

[6] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," IETF RFC 3209, 2001.

[7] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, and S. Molendini, "RSVP Refresh Overhead Reduction Extensions," IETF RFC 2961, 2001.

[8] V. Sharma and F. Hellstrand, "Framework for Multi-Protocol Label Switching (MPLS)-based Recovery," IETF RFC 3469, 2003.

[9] W. D. Grover, *Mesh-Based Survivable Networks: Options and Strategies for Optical, MPLS, SONET, and ATM Networking*. Upper Saddle River, New Jersey, USA: Prentice Hall PTR, 2003.

[10] J. P. Vasseur, M. Pickavet, and P. Demeester, *Network Recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS* San Francisco: Morgan Kaufmann 2004.

[11] P. Pan, G. Swallow, and A. Atlas, "Fast Reroute Extensions to RSVP-TE for LSP Tunnels," IETF RFC 4090, 2005.

[12] W. D. Grover and J. Doucette, "Advances in Optical Network Design with *p*-Cycles: Joint optimization and pre-selection of candidate *p*-cycles," in *Proc. of IEEE LEOS Summer Topicals*, Mont Tremblant, QC, Canada, Jul. 15-17, 2002, pp. 49-50.

[13] D. A. Schupke, C. G. Gruber, and A. Autenrieth, "Optimal Configuration of *p*-Cycles in WDM Networks," in *Proc. of IEEE International Conference on Communications (ICC 2002)*, New York, 2002, pp. 2761-2765.

[14] D. A. Schupke, M. C. Scheffel, and W. D. Grover, "An Efficient Strategy for Wavelength Conversion in WDM *p*-Cycle Networks," in *Proc. of Design of Reliable Communication Networks (DRCN 2003)*, Banff, Alberta, Canada, Oct.

19-22, 2003.

[15] D. Stamatelakis and W. D. Grover, "Theoretical Underpinnings for the Efficiency of Restorable Networks Using Preconfigured Cycles ("p-cycles")," *IEEE Transactions on Communications,* vol. 48, no. 8, pp. 1262-1265, Aug. 2000.

[16] C. G. Gruber and D. A. Schupke, "Capacity-efficient Planning of Resilient Networks with *p*-Cycles," in *Proc. of Networks, The International Telecommunication Strategy and Planning Symposium*, 2002.

[17] W. D. Grover, J. Doucette, A. Kodian, D. Leung, A. Sack, M. Clouqueur, and G. Shen, "Design of Survivable Networks Based on *p*-Cycles," in *Handbook of Optimization in Telecommunications*, M. G. C. Resende and P. M. Pardalos, ed., New York, USA: Springer Science+Business Media, Inc., 2006, pp. 391-434.

[18] M. S. Kiaei, C. Assi, and B. Jaumard, "A Survey on the *p*-Cycle Protection Method," *IEEE Communications Surveys and Tutorials,* vol. 11, no. 3, 2009.

[19] D. Stamatelakis and W. D. Grover, "IP layer Restoration and Network Planning Based on Virtual Protection Cycles," *IEEE J. Sel. Areas Commun. ,* vol. 18, no. 10, pp. 1938-1949, Oct. 2000.

[20] W. D. Grover and A. Kodian, "Failure-Independent Path Protection with *p*-Cycles: Efficient, Fast and Simple Protection for Transparent Optical Networks," in *Proc. of the 7th International Conference on Transparent Optical Networks (ICTON 2005)*, Barcelona, Spain, Jul. 3-7, 2005, pp. 363-369.

[21] A. Kodian and W. D. Grover, "Failure-Independent Path-Protecting p-Cycles: Efficient and Simple Fully Preconnected Optical-Path Protection," *Journal of Lightwave Technology,* vol. 23, no. 10, pp. 3241-3259, Oct. 2005.

[22] A. Kodian, W. D. Grover, and J. Doucette, "A Disjoint Route-Sets Approach to Design of Path-Protecting *p*-Cycle Networks," in *Proc. of the 5th International Workshop on Design of Reliable Communication Networks (DRCN 2005)*, Naples, Italy, Oct. 16-19, 2005, pp. 231-238.

[23] G. Shen and W. D. Grover, "Extending the *p*-Cycle Concept to Path Segment Protection for Span and Node Failure Recovery," *IEEE Journal on Selected Areas in Communications (JSAC),* vol. 21, no. 8, pp. 1306-1319, Oct. 2003.

[24] W. D. Grover and D. P. Onguetou, "A New Approach to Node-Failure Protection with Span-Protecting *p*-Cycles," in *Proc. of the 11th International Conference on Transparent Optical Networks (ICTON 2009)*, Azores, Jun. 28-Jul. 2, 2009, pp. 1-5.

[25] M. Pioro and D. Medhi, *Routing, Flow, and Capacity Design in Communication and Computer Networks*. San Francisco, CA: Morgan Kaufmann 2004.

[26] M. E. H. Pedersen, "Tunning and Simplifying Heuristical Optimization," PhD Thesis, Computational Engineering and Design Group School of Engineering Sciences, University of Southampton, 2010.

[27] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*, third edition ed. Hoboken, New Jersey: John Wiley & Sons, Inc., 2005.

[28] H. A. Taha, *Operations Research: An Introduction*, 8th ed. Upper Saddle River, New Jersey: Pearson Education, Inc., 2007.

[29] IBM. (Nov. 20th, 2010). *IBM ILOG V12.1 User's Manual for CPLEX. 2010.* Last accessed on Nov. 20th, 2010: ftp://public.dhe.ibm.com/software/websphere/ilog/docs/optimization/cplex/ps_usrmancplex.pdf.

[30] C. Rocha and B. Jaumard, "Revisiting *p*-Cycles / FIPP *p*-Cycles vs. Shared Link / Path Protection," in *Proc. of the 17th International Conference on Computer Communications and Networks (ICCCN '08)*, Aug. 3-7, 2008, pp. 1-6.

[31] D. Baloukov, W. D. Grover, and A. Kodian, "Toward Jointly Optimized Design of Failure-Independent Path-Protecting *p*-Cycle Networks," *Journal of Optical Networking,* vol. 7, no. 1, pp. 62-79, Jan. 2, 2008.

[32] W. D. Grover and D. Stamatelakis, "Bridging the Ring-Mesh Dichotomy with *p*-Cycles," in *Proc. of Design of Reliable Communication Networks (DRCN 2000)*, Munich, Germany, Apr. 9-12, 2000, pp. 92-104.

[33] J. Kang and M. J. Reed, "Bandwidth Protection in MPLS Networks Using *p*-Cycle Structure," in *Proc. of the 4th International Workshop on the Design of Reliable Communication Networks (DRCN 2003)*, Banff, Alberta, Canada, Oct. 19-22, 2003, pp. 356-362.

[34] S. A. El Shazely, O. A. M. Mohsen, and K. Shehatta, "Enhancing MPLS Network Fault Recovery Using *p*-Cycle with QoS Protection," in *Proc. of the 5th International Conference on Information and Communications Technology (ICICT 2007)*, 2007, pp. 197-202.

[35] W. Grover, J. Doucette, M. Clouqueur, D. Leung, and D. Stamatelakis, "New Options and Insights for Survivable Transport Networks," *IEEE Communications Magazine,* vol. 40, no. 1, pp. 34-41, 2002.

[36] D. Baloukov, "Studies in Failure Independent Path-Protecting *p*-Cycle Network Design," Department of Electrical and Computer Engineering, University of Alberta, Canada, 2009.

[37] O. Günlük. [Online]. Last accessed on Nov. 20th, 2010: http://sndlib.zib.de/home.action.

[38] A. Betker, C. Gerlach, R. Hulsermann, M. Jager, M. Barry, S. Bodamer, J. Spath, C. Gauger, and M. Kohn, "Reference Transport Network Scenarios," Technical Report, BMBF MultiTeraNet, July 2003. Last accessed on Nov. 20th, 2010: http://www.pt-it.pt-dlr.de/_media/MTN_Referenz_Netze.pdf.

[39] P. Batchelor et al., "Ultra high capacity optical transmission networks: Final report of action COST 239," Technical Report, Faculty of Electrical Engineering and Computing, Unversity of Zagreb, 1999.

[40] T. Čičić, A. Kvalbein, A. F. Hansen, and S. Gjessing, "Resilient Routing Layers and *p*-Cycles: Tradeoffs in Network Fault Tolerance," in *Proc. of High Performance Switching and Routing (HPSR 2005)*, May 12-14, 2005, pp. 278-282.

[41] A. Kodian, A. Sack, and W. D. Grover, "*p*-Cycle Network Design with Hop Limits and Circumference Limits," in *Proc. of the 1st International Conference on Broadband Networks (BROADNETS '04)*, 2004, pp. 244-253.

[42] R. Aggarwal, K. Kompella, T. Nadeau, and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)," IETF RFC 5884, 2010.

[43] K. Kompella and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures," IETF RFC 4379, 2006.