ROUTING ALGORITHMS IN SURVIVABLE TELECOMMUNICATION NETWORKS

By
János Tapolcai

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY (Ph.D.)

Supervised by

Dr. András Recski
Department of Computer Science and Information Theory

Dr. Tibor Cinkler
Department of Telecommunications and Media Informatics
High Speed Networks Laboratory
Budapest University of Technology and Economics

Budapest, Hungary
March 2004
# Table of Contents

Table of Contents

Kivonat

Abstract

Acknowledgements

Introduction

0.1 Preface ................................................................. 1
0.2 Economical Aspects and Network Protocols ......................... 1
   0.2.1 Network Architecture ........................................ 2
0.3 Multi-Service Survivable Transport Networks ........................ 3
   0.3.1 Provisioning Priority ....................................... 3
   0.3.2 Restoration Time ........................................... 4
0.4 Routing in Transport Networks .................................... 4
0.5 Mathematical Models and Algorithms ................................ 5
0.6 Overview of the Dissertation and Claims ........................... 5

1 Shared Path Protection (SPP) in Mesh Communication Networks with Bandwidth Guaranteed Tunnels 8
   1.1 Introduction ....................................................... 9
   1.2 Problem Definition ................................................ 11
      1.2.1 A Concept of Shared Risk Group (SRG) ..................... 11
      1.2.2 Traffic Model .............................................. 12
      1.2.3 Dynamic Routing ......................................... 13
      1.2.4 Modeling MPLS Networks ................................ 13
      1.2.5 Modeling Optical or GMPLS Networks .................... 13
      1.2.6 Load Balancing ............................................ 15
      1.2.7 Definition of Cost Functions ............................. 15
      1.2.8 Sharable Spare Capacity Matrix .......................... 17
      1.2.9 Cost Function in Matrix Form ............................ 22
   1.3 Problem Definition ................................................ 24
   1.4 Claim 1.1: Two-Step-Approach with Asymmetrically Weighted Disjoint Path-Pairs 26
      1.4.1 Integer Linear Programming Formulation ................... 27
      1.4.2 Performance Evaluation .................................... 29
      1.4.3 Conclusion .................................................. 31
   1.5 Claim 1.2: Two-Step-Approach with Maximum Likelihood Relaxation (MLR) .... 35
      1.5.1 Introduction .................................................. 35
1.5.2 Maximum Likelihood Relaxation (MLR) .................................. 35
1.5.3 Performance Evaluation ..................................................... 36
1.5.4 Conclusion ........................................................................ 40

2 Shared Segment Protection (SSP) in Mesh Communication Networks with Bandwidth Guaranteed Tunnels ........................................... 41
2.1 Introduction ......................................................................... 42
2.2 An Overview on SSP ............................................................. 44
  2.2.1 Definition of Cost Functions ........................................... 45
  2.2.2 Sharable Spare Capacity Matrix ....................................... 46
  2.2.3 Cost Function in Matrix Form ......................................... 47
2.3 Problem Definition ............................................................... 49
2.4 Claim 2.1: ILP Formulation of Shared Segment Protection .......... 50
  2.4.1 Performance Evaluation ................................................... 57
  2.4.2 Conclusion ..................................................................... 58
2.5 Claim 2.2: Extending the ILP of SSP with Constraints on Restoration Time and with Further Network Architectures .................................................. 60
  2.5.1 Motivation ..................................................................... 60
  2.5.2 A Network Modeling ....................................................... 62
  2.5.3 ILP Formulation for SSP with Constraints on Restoration Time and Adapting Further Network Architectures .................................................. 62
  2.5.4 A Heuristic of Improving the Runtime of Solving ILP: The SSP Algorithm .......................................................... 65
  2.5.5 Evaluation of the S/M Capability of the Network .................. 67
  2.5.6 Simulation Results .......................................................... 72
  2.5.7 Conclusion ..................................................................... 79
2.6 Claim 2.3: Complexity Analysis of Shared Segment Protection Problem .......................................................... 81
  2.6.1 Problem Formulation ........................................................ 81
  2.6.2 The proof of NP-completeness of SSP ................................. 81

3 Distributed Control Architecture for Shared Path Protection in Mesh Communication Networks with Bandwidth Guaranteed Tunnels 84
3.1 Introduction ......................................................................... 85
  3.1.1 Link-State Dissemination and Spare Provision Matrix Reconstruction .......................................................... 86
3.2 Claim 3.1: Reduced Information Scenario for Shared Path Protection .......................................................... 89
  3.2.1 The Survivable Routing .................................................... 91
  3.2.2 Performance Evaluation .................................................... 93
  3.2.3 Conclusion ..................................................................... 94
3.3 Claim 3.2: Reduced Information Scenario for Shared Segment Protection .......................................................... 95
  3.3.1 Introduction ..................................................................... 95
  3.3.2 DPIM-SAM ..................................................................... 96
  3.3.3 ILP Formulation for Shared Segment Protection at DPIM-SAM Scheme .......................................................... 96
  3.3.4 Performance Evaluation .................................................... 98

4 Conclusion ............................................................................. 100

Bibliography ............................................................................. 101
Kivonat

Disszertációm célja új és hatékony útvonalválasztó algoritmusok kidolgozása modern hibavédett távközlő hálózatokhoz, ahol az előre definiált szolgáltatás-minőségi feltételeket (QoS - Quality-of-Service) is figyelembe kell venni.

A problémák részéről megoldásához matematikai modelleket alkalmaztam, amelyek útmutatást adhatnak további hatékony heurisztikák kifejlesztéséhez. Referencia módszerként kidolgoztam több optimumot adó exponenciális futási idejű algoritmust is. Ezek segítettek a heurisztikák kiértékelésében. Az eredményeimet az alábbi három téziscsoportba fogáltam össze:

Az első téziscsoportban megosztott útvédelemmel foglalkozom. A feladatot lineáris algebrai módszerrel részféladatokra bontottam, majd heurisztikákkal oldottam meg.


Végül a harmadik téziscsoportban azt vizsgáltam, hogy az útvonalválasztási probléma, hogyan módosul elosztott rendszer esetén. Egy lineáris algebra alapuló módszerrel kiértékeltem a meglévő és új elosztott útvonalválasztó algoritmusokat. Végül szimulációval is igazoltam az elméleti eredményeket.

A disszertációban az ipari partnerekkel való eszmecserék során felmerült problémákkal foglalkoztunk. Ezek segítségével a jövő védett távközlési hálózataiban az útvonalválasztók szolgáltatásmínőségi feltételeket is figyelembe tudnának venni.

Kifejlesztettem egy hálózat-szimulációnos softvert az útvonalválasztó algoritmusok teszteléséhez, amivel igazolható ezek hatékonysséga.

A disszertációban ismertetett eredményeimet 14 lektorált nemzetközi publikáció támasztja alá, ebből 5 folyóirat és 9 konferencia cikk.
Abstract

The aim of my dissertation is to present effective routing algorithms in survivable mesh communication networks that meet several Quality-of-Service requirements.

The problems are formulated using mathematical models; that allow some necessary properties of the problem can be deduced and the performance of the heuristics can be improved. Exponential algorithms for evaluation of the performance of the heuristics are also introduced in the thesis.

The new results are classified in three areas, which are strongly correlated:

The thesis will firstly investigate shared protection in mesh telecommunication networks. Linear algebra is used to subdivide the whole problem into subproblems and solved with novel heuristics.

Shared Segment Protection (SSP) is the second topic investigated in the thesis. The Integer Linear formulation for SSP used to be an open problem, and the thesis provides a solution that can derive the optimal solution. The formulation is extended to respect several QoS properties. The impacts on the performance of applying different QoS constraints are also investigated.

The thesis also focuses on the distributed routing algorithms. A new algebraic way to investigate the performance of different distributed control architectures for shared protection is introduced. Using this mathematical model a classification of these results is suggested. A performance evaluation with simulation is conducted to verify the proposed distributed control architecture.

All the algorithms and solutions of this dissertation are strongly motivated by the telecommunication industry. They can equip the switches of the future survivable mesh telecommunication networks considering QoS requirements.

I implemented simulation tools to test the performance of the algorithms and to ensure their benefits.

The obtained results are supported by 14 refereed publications: 5 journal papers and 9 conference presentations.
Acknowledgements

I would like to thank my supervisors; András Reckši, whose mathematical supervision was indispensable and to Tibor Cinkler, whose help and support were essential in becoming a researcher in the field of telecommunication.

I would like to thank Pin-Han Ho (University of Waterloo, Ontario, Canada), who provided me with much help, advice and care, which were essential for my Dissertation. It was my pleasure to cooperate with him.

My work was done in the research cooperation framework between Ericsson and the High-Speed Networks Laboratory (HSNLab) at the Budapest University of Technology and Economics. I am grateful to Miklós Boda (Ericsson) and Tamás Henk (HSNLab) for their continuous support.

I also had the pleasure of spending a semester at Queen’s University as the guest of Dr. Hussein T. Mouftah. I would like to thank to Lajos Rónyai and all my co-authors particularly: Péter Laborczi, Csaba Gáspár, Szabolcs Szentes, and Ferenc Nizsalovszki (my high school math teacher), for helping me develop a love for this wonderful field of science.

Of course, I am grateful to my parents, László and Irina Tapolcai, for their patience and love; without them my dream of being a researcher would never have come true. I would like to thank all of my relatives - especially my grandfather and my cousins, Jósa and Marci and my grandmother in Russia.

Last but not least, I wish to thank my lovely friends for all the fun we had together - which is essential for productive work. In no particular order: Tamás Csoknyai, for working out this effective, bohemian lifestyle; Ferenc Bodon, for his entertaining stories; Csaba Kiss, for sharing with me the secrets of photon-mapping; Dániel Tófalvi, for his home movie theatre; Réka Tabák, for being my essential doctor; Klára Rusznák, for the time we spent together at Queen’s University; and of course to Eszter Székács, Nimród Major, Gábor Csóka, Kata Kádár, Szilvi Zsargó, Attila Novákí, Georgina Juhász, Anna Fodor, Koppány Kelemen, György Szldlenárík, Anikó Kosaras, Károly Kovács and Sundeep Ray.

Budapest, Hungary

March 21, 2004

János Tapolcai
Introduction

0.1 Preface

The modern communication networks are equipped with ultra-high speed switches to meet the dynamically changing traffic demands. Survivable routing is recognized as one of the best strategies to equip the networks with service continuity by pre-planning disjoint protection paths for working capacity [1, 2, 3, 4, 5, 6, 7, 8].

Due to the numerous Quality-of-Service (QoS) constraints the routing problem is NP-hard; thus, most of the papers presented approximation algorithms. Some of them are published in the mathematical conferences and journals; with theoretical polynomial time algorithms, which have some restrictions on topology and approximate the problem within a factor. The others are published in telecommunication conferences and journals based on effective heuristics, which solve the problem by assigning a sophisticated cost function to the edges and use shortest path search to approximate the problem.

Since the optimization problem is really complex, even the corresponding sub-problems are NP-hard, which has caused a huge gap in the research fields among different research groups.

During the period of my graduate studies (September 2000 - December 2003) my aim was to provide a good compromise and fill up the gap between the research fields. The problems are formulated using mathematical models, while the algorithms use the results of Graph Theory and Combinatorial Optimization. Integer Linear Programming (ILP) or also called Integer Programming (IP or InP) [9] is used to attack the NP-hard problems.

0.2 Economical Aspects and Network Protocols

According to TeleChoice [10], Multi-Protocol Label Switching (MPLS) has a huge potential to impact profitability of the carrier (and its customers) by offering a robust platform with strong traffic engineering mechanisms and QoS functions that lead to a revenue-generating environment. The long-term telecom survivors will be those who can find the balance between the technological innovation, improved customer service, and the efficient allocation of network resources. Efforts of finding that happy medium will position MPLS as the favorable core control framework to deliver all types of Internet traffic - in both the public and private network.

Most importantly, MPLS translates to more content on the network, which in turn represents high-bandwidth service opportunities and increased revenues. Streaming video, voice and other broadband services can be pushed to the edges by receiving priority over the others for achieving better QoS services than that by less time-sensitive applications (including e-mail).
The routing algorithms of the dissertation are mainly applicable on single-layer circuit-switched and virtual circuit-switched networks, like MPLS, ATM, SDH/SONET, Next Generation SDH/SONET and ASON.

Networks with MPLS and Asynchronous Transfer Mode (ATM) are virtually circuit-switched. ATM has been largely adopted by legacy networks developed in the late 1980s and early 1990s, which are still in the backbone to transport IP traffic, in access networks such as ADSL-based networks passive optical networks, and cellular telephony. The MPLS architecture can be seen as an extension of ATM, and it has been used to introduce QoS schemes in the IP networks.

Networks with Synchronous Optical Network/Synchronous Digital Hierarchy (SONET/SDH), Next Generation SDH/SONET and Dense Wavelength Division Multiplexing (DWDM) are circuit-switched, where SONET/SDH is one of the main underlying transport technologies supporting telephone systems, while DWDM is also a circuit-switched technology with optical circuit-switched connections known as lightpaths.

Generalized Multi-Protocol Label Switching (GMPLS) defined by Internet Engineering Task Force (IETF) aims to cope with the extremely complex multilayer architecture that has been cobbled together to carry IP services over networks designed to support voice and fixed circuits. GMPLS can be deployed in the following two ways: either the overlay model or the peer model. In the overlay model, User-Network Interface (UNI) is defined, where the router is a client to the optical domain and interacts only with the directly adjacent optical nodes. The actual physical lightpaths are routed/initiated by the carrier of the optical core instead of by the users. Since the routing algorithms introduced in the dissertation are mainly designed to work in a single domain, they are more suitable to the network environments with the overlay model rather than that adopting the peer model.

The International Telecommunications Union-Telecommunications (ITU-T) has introduced two protocol-independent framework models: the general Automatic Switched Transport Network (ASTN) and the Automatic Switched Optical Network (ASON). Since GMPLS can be mapped to ASTN/ASON models, it may well become the adopted standard for such implementations. The ASTN/ASON model focuses on providing the optical transport network with an intelligent optical control plane, incorporating dynamic network provisioning combined with network survivability, protection, and restoration.

### 0.2.1 Network Architecture

To manage the huge amount of data traffic, a well designed control plane supported by ultra high-speed switching devices steers the whole network. MPLS based control plane is favoured for its strong functionality in equipping networks with a multi-service environment and the ability of performing traffic engineering. In [11] an overview is given on the MPLS-based control plane.

The IP/MPLS control box is configured with MPLS control plane functional elements, which mainly contains the following three modules: First, the CSPF (Constraint-based Shortest Path First) path selection module is in charge of calculation for both working and protection paths upon different constraints such as reservable bandwidth of a link, wavelength availability, and diversity requirement, according to the traffic engineering (TE) database and Shared Risk Group (SRG) information. Second, the signaling component is activated to send control packets
to setup the connection corresponding to the routing decision made by the CSPF module. In addition, it also responds to requests from the other nodes in the network for establishing connections. Third, there is an ISIS/OSPF routing protocol module with ISIS/OSPF OPT and TE extensions for a dynamic dissemination of link-state and TE metrics so that the other nodes in the domain can verify/update the information in their link-state and TE database. The functional diagram of a Label Switch Router (LSR) is shown in Fig. 1.

Figure 1: Functional diagram for nodes equipped with IP/MPLS control plane.[11]

0.3 Multi-Service Survivable Transport Networks

In the circuit switched and virtual circuit switched transport layer, instead of using delay, jitter, or packet-discard policies that distinguish class of services in packet-switching networks, there are mainly two notable aspects of Quality-Of-Service (QoS) criteria:

0.3.1 Provisioning Priority

Latency of path provisioning has been one of the most critical issues in a dynamically routed networks. It is important to reduce the computation latency in making routing decision. The path selection algorithms should address higher priority on path setup requests with QoS requirements. If necessary, a QoS connection request should be able to take the network resources originally in use by best effort paths. The interrupted best effort paths have to search for some other network resources to resume the original sessions. In this case, to develop an efficient algorithm for achieving inter-class resource sharing can improve network throughput.

A successful setup of a path corresponding a connection request implies an immediate service available to a group of end users with the same (or similar) QoS requirement. Therefore, the path provisioning priority can be taken as a QoS metric for the effort of achieving a multi-service network environment. To distinguish the commercial importance of each path, connection requests with lower class of service should suffer a larger chance of blocking, during which the requests are buffered for the next available path or merged into some other paths of the same class of service. Connection requests with higher class of service should have smaller chance of blocking and get an immediate service. Reference [12] has conducted a simulation-based study on the topic of QoS routing in optical networks with wavelength assignment, in which schemes with different resource sharing policies are compared in terms of the revenue generated.
0.3.2 Restoration Time

In the Internet, a 1+1 dedicated protection is preferred along most physical conduits for the sake of simplicity in signaling mechanisms and fastest restoration services to millions of end users. Another important reason is that the Internet is usually sparsely meshed, where a design of sharing spare capacity among different working paths may not bring much capacity saving. In dynamic routed networks, a QoS path needs a pre-scheduled protection and restoration plan. In addition, the restoration time taken by the recovery mechanism after the occurrence of failure should be up to a limit according to its Service Level Agreement. Requirements for restoration time of a path are determined by several aspects, such as the characteristic of services carried by the path, the buffer space of the two end nodes that terminate the path, and the bandwidth of the path. A protection scheme is needed that can guarantee a maximal restoration time without increasing much computation complexity.

0.4 Routing in Transport Networks

Path selection can be performed in a centralized manner, in which all the computation tasks are taken by the Network Management System (NMS). In the centralized control, the data inconsistency among TE database of each node (see Fig. 1) can never occur, the NMS has to deal with every network event one after the other, and obviously cannot scale well. In addition, reliability of the control plane is low. Therefore, it is preferred to distribute the control efforts to all the network nodes for achieving better robustness and scalability. Each node disseminates its link-state periodically to all the other nodes according the OSPF/IS-IS protocol for keeping data consistency of the TE database of each node. However, the distributed control architecture may suffer from the following two problems. First, stale link-state may exist due to either a periodical or asynchronous dissemination process, which may result in a conflict between resource reservation processes initiated by two different source nodes upon the same network resources. In the optical layer, this problem can be solved by the holding/preemption parameters in the path setup message specified in RSVP and Constraint based Label Distribution Protocol (CR-LDP). The connection request with lower holding/preemption priority has to withdraw and restart a new resource reservation process. The second problem is also centered on the link-state dissemination process. Flooding of link-state from every node to the whole network may congest the control plane and disturb the other signaling mechanisms. In addition, the periodic flooding is also the reason for the stale linkstate mentioned above.

Chapters 1 and 2 of the Dissertation are based on the centralized, while Chapter 3 on the distributed control architecture.

A path selection process which is adaptive to traffic variation is called an adaptive routing. In terms of the extent of adaptation, a path selection algorithm can be either fully-adaptive, partially-adaptive, or non-adaptive. We will only address the fully-adaptive routing since it can achieve load-balancing in a dynamic network. With fully-adaptive routing, a routing decision is made upon current network-wide link-state. With the partially-adaptive routing, only a limited extent of traffic variation is considered in path selection, such as Fixed Alternate.

---

1Shortest Path First (SPF) uses Dijkstra's algorithm, which is one of the most commonly used approaches for finding a shortest path between two nodes.
Routing.

Survivable Routing with Dynamic Traffic

In the Dissertation several protection schemes are introduced to pre-plan spare capacity for restoring traffic subjected to failure. The most widely recognized strategy of performing protection is to find spare capacity that is physically disjoint from the corresponding working resources, over which the data flow could be switched over during any failure. Path selection schemes that consider both working and protection paths for a connection request are called diverse routing or survivable routing.

In an optical network, the number of lightpaths coexisting is much less than that in conventional IP/MPLS networks. Therefore, the traffic variation in optical networks is more "discrete" than that in packet-switching networks. There might exist tens of seconds to several minutes between any two arrivals of network events and opens the door to use routing algorithms, which require much calculation time.

0.5 Mathematical Models and Algorithms

The mesh telecommunication networks are modeled by graphs. The edges are undirected or bi-directed since the communication links are considered to be symmetric. It should be noted that the algorithms are designed such that every link is represented by two directed edges in the opposite direction. There are capacity and cost functions on the edges. Due to the hierarchical structure of telecommunication networks a common approach [13] to achieve scalability is to use aggregating information as shown in Fig. 2. This approach keeps the size of the graph where the routing decision is made between 20 to 100 nodes. The graphs are at least 2-link connected with average nodal degree of 3 to 4.

The implemented graph algorithms used in the simulations are the following:

- Dijkstra's algorithm [14, 15, 16],
- Suurballe's algorithm [17] for diverse routing,
- Minimum Cost Network Flow [18]
- Multi-commodity Network flow [19], with ILP
- algorithm to calculate Gomory-Hu trees [20]
- and a matrix algorithm to calculate Singular Value Decomposition [21]

Flows and cuts in the graphs play an important role of the design strategy of routing algorithms.

Integer Linear Programming (ILP) [16] is used to formulate the NP-hard problems and commercial software e.g., CPLEX [22] and free softwares e.g., LP-Solve [23] are available for solving instances of ILP formulations.

0.6 Overview of the Dissertation and Claims

The Dissertation consists of an Introduction and 3 chapters.

In Chapter 1 the Dissertation will investigate shared path protection (SPP) in mesh telecommunication networks. Linear algebra is used to subdivide the whole problem into sub-problems for formulating new heuristics. It consists of two claims introducing two effective heuristics.
Figure 2: Hierarchical model of the network [13].

Shared segment protection (SSP) is the second topic investigated in Chapter 2 of the thesis. The Integer Linear formulation for SSP used to be an open problem, and the thesis provides a solution that can derive the optimal answer. The formulation is extended to take several QoS properties into consideration. The impacts on the performance applying different QoS constraints are also investigated.
In Chapter 3 of the Dissertation the distributed routing algorithms are in focus. A new algebraic way to investigate the performance of different distributed control architectures for shared protection is introduced. Using this mathematical model a classification of these results is suggested. The performance evaluation with simulation is conducted to verify the proposed distributed control architecture.
Chapter 1

Shared Path Protection (SPP) in Mesh Communication Networks with Bandwidth Guaranteed Tunnels
1.1 Introduction

Survivability has emerged as the most important issue in the design of the control and management plane for the next-generation networks. To deal with any unexpected interruption caused by accidental events (such as rodent bite on communication fibers), pre-planning a protection (or backup) path with sufficient bandwidth for each working (or active) path has been widely accepted as the most effective solution. The strategy is also known as survivable routing in the path selection stage. Imnumerable research has been conducted since the mid 1990's to equip the Internet and Metro-area networks with reliability and resilience to any unexpected interruption [24, 25, 26, 2, 27, 28, 11, 6, 29, 30, 31].

For the networks with connection requests arriving one after the other without any knowledge of future arrivals, it is important to develop a suite of inter-operable strategies that can, in real-time, find a link-disjoint working and protection path-pair upon the current link-state with high capacity-efficiency.

With path protection, working and protection path-pairs are link- or node-disjoint, in which two types of protection are defined – dedicated and shared protection. The difference between the two lies in whether or not spare capacity resource sharing is allowed between different protection paths. In dedicated protection, each working and protection path-pair is pre-configured, and is launched with the same copy of data between a source-destination (s - d) pair at the same time during a normal operation. Although dedicated protection (e.g., 1 + 1) provides a very fast restoration service, the ratio of redundancy (i.e., the ratio of capacity taken by protection and working paths in the network) usually reaches 120%. On the other hand, approaches in the shared path protection only working paths are launched with data flows. The spare capacity reserved by protection path is pre-planned without the necessity of being pre-configured, and may be shared with the other protection paths in the shared protection mode. It has been observed that the spare capacity resource sharing between different protection paths can substantially reduce the ratio of redundancy required to achieve 100% restorability at the expense of a little longer restoration time [2, 31].

For the survivable routing problem, Suurballe’s algorithm, reported in the early 70’s, is famous for its polynomial computation complexity in solving optimal disjoint path-pairs in terms of the cost sum of the two paths in a directed graph [32, 17, 25]. It is notable that Suurballe’s algorithm uses the same suite of link-state to derive the two paths. For shared protection, the constraint of spare capacity sharing must be investigated upon each network link before the best protection path can be derived for a working path. Whether or not a link has sharable spare capacity for a protection path depends on the physical location of the corresponding working path. This is also known as the dependency of the protection path on its working path.

The dependency has imposed different design criteria such that the existing linear formulations and most of the reported schemes cannot address the problem. In order to explore the dependency, the most straightforward way is to use Two-Step-Approach (discussed in Claim 1.1) to derive the link- or node-disjoint path-pairs [33, 26, 5, 34, 2, 27, 28, 6, 29, 30, 35]. However, the Two-Step-Approach is neither general to different network topologies nor systematic in deriving the best working and protection path-pairs upon the current link-state. In
[26], a method is provided to find an optimal protection path given a working path for shared protection, called Pool-Reserved Backup Sharing. However, the study did touch the problem how to select the working paths. To address the allocation of working paths, the studies in [5, 27, 6, 29, 3] inspect $k$-shortest paths between each $s - d$ pair one after the other until the least-cost working and shared protection path-pair is derived. In [29], a similar method to that of [26] is adopted along with the inspection of the $k$-shortest paths. The study in [27] creates special data structure to quickly derive the $k$-shortest paths for solving the optimal diverse routing problem. However, it fails to specifically define the sharing constraint to achieve 100% restorability. The study in [5] proposes an approach that further assigns cost to those links with sharable spare capacity, in which the probability for a link to be used by some other affected working paths due to failure is considered. However, this method does not guarantee the availability for the spare capacity along the derived protection paths at the occurrence of failure, and is not effective in some cases where the restorability of traffic flows is strictly required. In [6], a novel link cost metric for finding working paths is provided. The cost for the working path to take link $j$ is determined by the maximum spare capacity among all the other links in the network which protects link $j$. The proposed link metric can encourage the working path to take links yielding smaller maximum non-sharable spare capacity along some other links for the protection path, and is reported to have a better performance behavior (in terms of blocking probability) than simply using hop count as the cost function. All the above schemes take the approach of finding the shortest, the second shortest, ... the $n^{th}$-shortest paths between $s$ and $d$, which wears out the novelty and leaves a large space to improve.

In addition to the end-to-end protection, the studies in [33, 34, 2, 28, 30] suggest to segment each working path and find a protection path segment for each working path segment. In [34], a heuristic algorithm is developed, which, however, did not consider the backup bandwidth sharing until the physical routes of the backup segments are defined. In [33] and [30], two related dynamic algorithms are proposed to switch over for each link from its immediate upstream node and merge back to the original path at the immediate downstream node and any of the downstream nodes, respectively. However, none of the studies impose any limitation on the length of the backup paths, and may impair the overall performance. It is notable that all the above schemes are Active-Path-First-based [6] (abbreviated as APF, which means that the working path is derived first), in which little attempt has been made to jointly solve the working and shared protection path-pair. In [2, 28], a framework called Short Leap Shared Protection (SLSP) along with a dynamic algorithm called CDR (Cascaded Diverse Routing) is proposed to perform segmented shared protection, in which the enumeration of $k$-shortest paths in each segment of the working path is performed. The next two chapters contain a detailed study on the above topic. The performance is turned out to outperform its shared protection counterparts at the expense of much larger computation complexity. In addition, the link-based or segment-based shared protection takes extra signaling efforts and is imposed of high requirements on the hardware responsiveness.

However, this chapter turns back to the path shared protection aiming at achieving high computation efficiency without losing much performance. We first define the optimality for a working and protection path-pair in shared protection. Then the optimal survivable routing problem is formulated as an Integer Linear Programming process, in which the optimal working
and protection path-pair are jointly found. To avoid the high computational complexity caused by the NP-completeness in solving the ILP formulation, we developed two heuristic schemes to explore the best efficiency in solving the problem. The first (in Claim 1.1), based on an algorithm published in [32], solves the problem called the asymmetrically weighted node-disjoint path-pair problem. It is a generalization of both Suurballe’s algorithm and the shortest path search and used to compute the working path then in the next step the optimal shared protection path is calculated. The second scheme (in Claim 1.2), called Maximum Likelihood Relaxation (MLR), is a modified Dijkstra’s algorithm which yields polynomial time complexity. The MLR scheme considers the allocation of the protection path while the working path is being calculated, which aims to select a working path in such a way that the number of links containing enough sharable spare capacity weighted by the reciprocal of the link cost is maximized. We evaluate the performance of the proposed schemes by making a comparison with their counterparts using four network topologies with dynamic traffic pattern.

1.2 Problem Definition

A few general rules about notations:

- all scalar variables are printed with lower letters. (e.g. \(a\))
- all the sets are printed with capital letters. A path is also printed with capital letters as it is a set of links. In the implementations of the paths the links are usually stored in order (from the source node till the destination), though this is not necessary for the formulas. (e.g. \(A\))
- all vectors are printed with underlined capital letters. (e.g. \(A\))
- all matrices are printed with double underlined capital letters. (e.g. \(A\))

For language precision, in the following context the physical network consists of links and nodes. A link of the network is always bi-directional and represents a connection between two network nodes. In optical networks where the wavelength of the optical channel is distinguished, multiple links should be used representing each wavelength between the same nodes. The graph represents the network at logical level. It consists of edges (or arcs) and vertices. An “arc” is always directed while an “edge” is undirected. Sections 1.2.4 and 1.2.5 give an overview on different graph models applied onto telecommunication networks.

1.2.1 A Concept of Shared Risk Group (SRG)

Shared Risk Group (SRG) is defined as a group of network elements (i.e., either links, nodes, physical devices, software/protocol identities, etc, or a mix of them) possibly subject to the same risk of single failure. In practical cases an SRG may contain several seemingly unrelated and arbitrarily selected network elements. We define that a working path is involved in an SRG if it traverses through any network element that belongs to the SRG. A path may be involved in several SRG’s. Two working paths share the same risk of a single failure if they are involved in any common SRG. A working path is said to be SRG-disjoint with its protection path if the two paths are not involved in any common SRG. In this Dissertation, the SRG-disjointedness for a working and protection path-pair is the major effort of achieving 100% restorability for the working data flows under the single failure scenario. In addition, The SRG-disjointedness must be kept for the working paths whose protection paths share spare capacity, such that the
restoration will not be subject to resource contention when two working paths are interrupted by a single failure.

The spare capacity taken by backup path segments is called spare capacity that are only reserved but not configured during the normal operation. Therefore, the spare capacity can also be used by some best-effort traffic that can tolerate a service interruption. In the event that a failure occurs that interrupts a working path (a failure of any SRG, such as a fibre-cut or loss of signal due to the failure of any network element), the switching fabric structures in the nodes along the corresponding protection path are configured by prioritized signalling followed by traffic switchover to recover the original service supported by the working path. Therefore, the protection paths of different working paths can reserve the same spare capacity if the working paths are not involved in a common SRG, and are considered to share the same risk of single failure. In this case, two working paths can share the same risk of single failure if they take any common SRG in the network. In other words, whether or not two protection paths can share a spare capacity depends on the physical location of their working paths. A simple example is shown in Fig. 1.1. $W_1$ and $P_1$ form a working and protection path-pair. The protection path of $W_2$ (any other working path) should exclude the possibility of using any of the spare capacity taken by $P_1$ because $W_2$ traverses link $A - B$, which shares the same risk of a single failure with $W_1$.

![Figure 1.1: An example to illustrate the SRG constraint.](image)

We take an assumption of single failure scenario, where the algorithm only deals with the situation that a single link in the total network is unexpectedly interrupted at a moment.

Most of the past studies focused on the case where each single network element in the network topology serves as an SRG. This simplification comes from the assumption that the probability of each physical conduit to be subject to a failure is small and thus can be taken as independent events even under the single failure scenario. A special case can be seen in the solving of the link-disjoint diverse routing problem for shared protection. However, when a general definition of the SRGs is desired, a more complicated description and further elaborations are required to achieve an efficient implementation of any survivable routing algorithm for shared protection.

1.2.2 Traffic Model

Evaluation of the algorithm is carried out using dynamic traffic patterns, where working and protection paths are set up for each connection request as it arrives and the paths are released after a finite amount of time. The dynamic routing problem [36] refers to problems that arise during path establishment in network with dynamic traffic demands.
In order to be able to compare different routing algorithms and strategies, a dynamic traffic pattern was generated and stored in advance. The arrival and holding times of the connection requests for each node pair are generated according to an estimated traffic matrix, such that an Interrupted Poisson Process and Pareto inter-arrival time \cite{37} are integrated together with exponential holding time. For each network a traffic matrix in year 2005 is estimated according to \cite{38}, which is a slightly improved model compared to \cite{39}.

1.2.3 Dynamic Routing

In dynamic routing the routing decision must be made as the connection request arrives in the network. In case there are insufficient network resources to set up a connection, the request waits for a limited amount of time before it is blocked. There are several objectives evaluated in order to compare different routing strategies: blocking probability is the most realistic objective used, which refers to the number of blocked connection requests divided by the total number of connection requests. However its deviation is high unless the blocked requests are weighted according to their estimated capacity need. This estimation usually depends on the minimum hop distance between the source and destination of the connection request. Therefore, we prefer to use a more simple objective called the average waiting time. This has infinite limits on waiting before being blocked, where all arriving connection requests are routed and the total waiting time is evaluated. Another fundamental issue is the fairness; we want to establish connections so that none of them "starve". Although it is an intuitively natural concept, finding a concrete definition of fairness that captures the goals of efficient routing, is a subtle issue. We wish to prevent the starvation of individual connections in a way that allows high network utilization at the same time. For evaluating the fairness of different dynamic routing algorithms, we consider the maximal waiting time. The network utilization, or network load, is also evaluated and this is the amount of capacity allocated divided by the total available capacity of the network. The final objective is the work load, which is the amount of capacity allocated for working routes divided by the total available capacity of the network.

1.2.4 Modeling MPLS Networks

An MPLS network can be modeled by a directed graph where the node is the vertex of the graph. The physical link of the network is modeled by two arcs between two nodes with opposite directions in the graph, since all routing algorithms introduced in the dissertation are based on directed graphs. Note that the arcs connected to the same link should be in the same SRG. However the communication links are considered to be symmetric, and the network can be modeled by an undirected graph as well.

1.2.5 Modeling Optical or GMPLS Networks

The basic concept of DWDM technology is the ability to simultaneously transmit data on multiple wavelengths (WIs) on a single fiber. Each wavelength channel operates typically at 2.5-10 Gbps. The most common commercial systems use 8-32 wavelengths per fiber. Each cable contains many (e.g. 20 or more) fibers. The improvements of the latest Optical Cross-connects (OXC) offer an ability to dynamically change the optical layer connectivity within milliseconds.
In DWDM networks OXCs are used to switch individual wavelengths optically and establish lightpath between nonadjacent nodes. A lightpath is an optical path established between two nodes of the network, carrying only optical signals. Two lightpaths can use the same links if and only if they use different wavelengths. OXCs may or may not have wavelength converters, i.e., devices that transform data streams coming in at one specific wavelength into an outgoing data stream at another specific wavelength.

The price of an optical wavelength converter is approximately as high as that of an $8 \times 8$ optical switch. Thus most of the all-optical networks have been built without any wavelength converter. Instead of wavelength conversion capable optical cross-connects network providers use the significantly cheaper opto-electronical cross-connects (EXCs). In EXCs the optical signal is first converted to electrical signal then the electrical space-switching is performed and then it is converted back to optical domain again of to any wavelength. By using EXCs in the network we loose the total transparency of bit rates and signal formats.

The input of the routing algorithms introduced in this dissertation contains a graph with SRGs defined as a set of edges. The aim of this section is to summarize the graph models built up on networks.

Simple graph model

The first graph model introduced has a very simple structure. Due to the simplicity it has some restriction on the optical networks. It is based on optical networks having both opto-electrical and all-optical cross-connects. [40] defined the necessary conditions on the place of EXCs in the network, such that every set of lightpaths can be routed with a number of wavelengths equal to its congestion bound. In other words the network can be modeled as an MPLS network (see previous subsection), where a lightpath is represented with a path of capacity one and the capacities of the edges equal to the number of wavelength channels on the corresponding optical link.

The necessary conditions on placing EXCs (or full wavelength conversion capable OXCs) in the network are published in [40]. They are the followings: Let $G'$ be an undirected graph where the vertices are the OXCs of the optical network (with no wavelength converter) and has also additional vertices representing the links connected to EXCs (or full wavelength conversion capable OXCs). The edges of $G'$ are

- the optical links between OXCs (with no wavelength converter)
- if a vertex represents a link to an OXC (with no wavelength converter) then the corresponding two vertices are connected by an edge in $G'$
- vertices representing the same link are connected to each other by an edge in $G'$.

A spider is a tree with at most one node of degree greater than two. If each component of $G'$ is a spider an optimal wavelength assignment in the optical network can be done with efficient polynomial algorithms and this simple graph model can be applied.

In [41] a polynomial time approximation algorithm was given for finding the sufficient set of EXC (or full wavelength conversion capable OXC) in the network.
Wavelength graph model

In [42] a wavelength graph was introduced, to model wavelength channels of the optical links and all types of optical cross connects. Since each arc of the graph belongs to a network element, the SRGs can be defined on the arcs of the graph as well.

1.2.6 Load Balancing

For the networks with connection requests arriving one after the other without any knowledge of future arrivals, the general goal is to develop a suite of inter-operable strategies that has a superb overall performance with low blocking probability, short average and maximal waiting time of establishing connections, and low network utilization.

A very common idea is to use load balancing functions, which set the weights on the links \(c_l\) such that a good overall performance can be expected using capacity-efficient routing algorithms for each connection request. These routing algorithms minimize the weighted capacity allocation (see next section for details). The idea was first introduced in Open Shortest Path First (OSPF) internet routing protocol and called OSPF weights in optical networks as well. The natural concept is to use higher cost on critical edges. Several papers deal with this topic, e.g. [43], where the cost of each edge is a piece-wise linear increasing and convex function of the load.

Finding good cost function is a key factor of the overall performance of the algorithms. We implemented the load balancing routine of [43] and improved its performance by some state of the art methods. Finding a universal load balancing method is a further research topic of this field.

1.2.7 Definition of Cost Functions

Now we define the cost function of a connection. It represents the weighted capacity allocation and is composed of the cost of the working path and the protection path. Given a network with a set of links \(L\) and nodes \(N\). The task is to find a working (denoted as \(W\)) and protection (denoted as \(P\)) path between the source \(s\) and destination node \(d\) with bandwidth \(b(W)\).

For easier understanding this section defines the problem at the level of the network. Note that in the graph model described in sections 1.2.4 and 1.2.5 each arc belongs to a network element, thus using this relationship the definitions of this section can be easily transformed for the graph model.

The cost of each link \(j\) in the network is denoted by \(c_j\) and is calculated by the load balancing function. The capacity along links \(j, \forall j \in L\), can be categorized into the following three types:

- **Free capacity** (denoted as \(f_j\)), which is the unreserved link capacity that can be reserved as either working or spare capacity.

- **Spare capacity** (denoted as \(v_j\)), which is the link capacity reserved by some backup path.

- **Working capacity** (denoted as \(q_j\)) which is the link capacity already taken by some working path, and cannot be taken for any use until the corresponding working path is torn down.

In order to find a working path a cost function on the set of links is defined as follows:
\[ c_j^W = \begin{cases} b(W) \cdot c_j + \varepsilon & \text{if link } j \text{ is reservable} \\ \infty & \text{otherwise} \end{cases} \]  

(1.2.1)

where \( c_j \) is the cost for each unit of bandwidth taken by working paths along link \( j \), and \( \varepsilon \) is a small number defined as \( \varepsilon = \frac{\min_{j \in L} c_j}{|L|} \). The fact that link \( j \) is not reservable by \( W \) can only be due to \( b(W) > f_j \), where \( f_j \) is the free capacity along link \( j \) (as illustrated in Fig. 1.2). The total cost of the working path is \( \sum_{j \in W} (b(W) \cdot c_j + \varepsilon) \). Instead of using \( \infty \) as a cost, it is more common to define feasible conditions (see Section 1.3). The purpose of additionally imposing the small number \( \varepsilon \) in the cost function is to match the cost of backup path, which will be defined later.

![Figure 1.2: An illustration of the categories of capacities along link \( j \).](image)

For finding the backup path of \( W \), we need first to define the corresponding spare link-state and cost functions, which are specific to \( W \), and can only be derived by correlating all the other working paths on the same SRG with \( W \). **Spare link-state** terms the link-state assigned to the protection path (shareable spare capacity, and the non-shareable spare capacity), which depends on the corresponding working path.

With the presence of \( W \), the spare capacity along link \( j \) can be further categorized into the following two types:

**Shareable spare capacity** (denoted as \( h_j^W \)), which is the link capacity that has been reserved by some other backup paths, and is shareable with the backup path of \( W \).

**Non-shareable spare capacity** (denoted as \( s_j^W \)), which is the link capacity that has been reserved by some other protection paths, and is not shareable with the protection path of \( W \) due to the SRG constraint. Note that \( v_j = s_j^W + h_j^W \), which is the total spare capacity along link \( j \).

The protection path may traverse link \( j \) in any one of the following three states:

- the case where the link has sufficient shareable spare capacity (i.e., \( h_j^W \geq b(W) \)), in which the backup path can take this link with the smallest cost (denoted as \( \varepsilon \));
- the case where \( f_j + h_j^W \geq b(W) > h_j^W \), and the backup path must partly (or totally) take free capacity along this link with an extra cost. In this case the spare link-state is \( b(W) \cdot r_j^W \cdot c_j + \varepsilon \), where \( r_j^W \) is a \([0,1]\) scaling parameter determined by the location of \( W \) and will be defined later.
• The link does not have sufficient sharable spare capacity and free capacity (i.e., \( f_j + h_j^W < b(W) \)), thus the backup path cannot traverse through this link by any means. In this case the cost is \( \infty \). Due to the dependency between the working and spare capacities in the network, the parameters \( r_j^W \), \( h_j^W \), and \( s_j^W \) cannot be defined until the presence of \( W \).

\( r_j^W \) is defined as

\[
r_j^W = 1 - \frac{h_j^W}{b(W)}
\]

(1.22)

for any link \( j \in L \). It is clear that \( r_j^W \) is 1 if there is no sharable spare capacity available along link \( j \) and is approaching to 0 if \( h_j^W \) is close to \( b(W) \). In the former case (i.e., the case of \( r_j^W = 1 \)), the cost for the backup segment to take this link is \( b(W) \cdot c_j + \varepsilon \), which is the same as that for the working path since all the reserved bandwidth have to be from the free capacity region as shown in Fig. 1.2. The spare link-state for the protection path of \( W \) can be expressed as:

\[
c_j^P = \begin{cases} 
  b(W) \cdot c_j \cdot r_j^W + \varepsilon & \text{if } h_j^W \geq b(W) \\
  \infty & \text{if } h_j^W + f_j < b(W)
\end{cases}
\]

(1.23)

Fig. 1.3 shows the three situations defined in Eq. (1.23). In Fig. 1.3(a), the backup segment can have all \( b(W) \) in the sharable spare capacity region, therefore, the cost is \( \varepsilon \), as shown in the first condition in Eq. (1.23). In Fig. 1.3(b) and Fig. 1.3(c), the backup path of \( W \) may partly take the free capacity region and the sharable spare capacity region; therefore, the link cost is \( b(W) \cdot c_j \cdot r_j^W + \varepsilon \), which is shown in the second condition in Eq. (1.23). In Fig. 1.3(d), the link cost is infinity because the backup path of \( W \) cannot be supported by the residual capacity of the link, which is shown in the third condition in Eq. (1.23). Note that the protection path is assumed to take sharable spare capacity along a link whenever there is any sharable spare capacity available. If there is not enough sharable spare capacity along this link to cover the total bandwidth demand for protecting \( W \) (i.e., \( b(W) \)), the backup path takes free capacity after considering all the sharable spare capacity.

Note that the adoption of the small constant \( \varepsilon \) is to keep the continuity between the first and second condition in Eq. (1.23). In this case, the cost of link \( j \) is set to \( \varepsilon \) as \( h_j^W = b(W) \) for both of the conditions. This is also the reason we impose \( \varepsilon \) in the cost function for the working path shown in Eq. (1.21), in which the cost for the working and backup path segments to take free capacity can match each other.

Our objective is to determine \( c_j^P \) in Eq. (1.23) — the spare link-state that defines the cost of the backup path of \( W \) passing through link \( j \), in which \( h_j^W \) is the only variable that must be figured out (or equivalently, \( s_j^W \) since \( v_j = s_j^W + h_j^W \)). Note that \( h_j^W \) and \( s_j^W \) are network-wide link-state specific to the presence of \( W \).

1.2.8 Sharable Spare Capacity Matrix

SRG is defined as a group of network elements subject to the same risk of single failure. With the general definition of SRGs, a failure may attack any combination of network elements. The key idea is that the complete set of SRGs in the network, denoted as \( \overline{SRG} \), is further divided into the following two categories:
Figure 1.3: The possible situations of a different cost function defined in Eq. (1.2.3). $f_j$ denotes the amount of “free spare capacity” while $h_j^W$ for “sharable spare capacity”.

**First-type of SRG** *(denoted as $SRG_{res}$)* An SRG belongs to this type if the network is still kept connected after the failure affects such that 100% restorability can be achieved. In other words, the network elements of the SRG do not form a cut in the network topology; in this case, working flows affected by the failure are restorable.

**Second-type of SRG** *(denoted as $SRG_{cut}$)* An SRG belongs to this type if the network is turned into multiple isolated fragments when it is attacked by a failure. In other words, the network elements of the SRG form a cut of the network topology. Thus, the interruption upon the associate working paths can never be restored.

It is clear that $|SRG|$ *(i.e., the number of the total SRGs in the network)* is one of the dominating factors for the complexity of signaling mechanisms and protocol computation. Therefore, a way of reducing $|SRG|$ under consideration without losing 100% restorability for any possible failure is critical to the mathematical formulation. In the event that all the network elements of $B_1$ are also covered by $B_2$ ($B_1 \subset B_2$), and both of them belongs to $SRG_{res}$. $B_1$ does not need to be considered since the restoration plan for the bigger SRG will protect the smaller one. Since we are under the single failure scenario, it is impossible to have two failures on the network elements in $B_1$ and $B_2$ at the same moment.

This can be generalized as the following rule:

If an SRG $B_1$ is a part of another SRG $B_2$ ($B_1 \subset B_2$), and the failure of $B_1$ separates the nodes of the network into the same sets as the failure of $B_2$, then $B_1$ can be eliminated and need not to be considered.

Let $SRG'_{res}$ and $SRG'_{cut}$ denote a subset of $SRG_{res}$ and $SRG_{cut}$ with all the redundant SRGs being eliminated, respectively. Thus we can define a set of SRG: $SRG' = SRG'_{res} + SRG'_{cut}$. A special case can be that each edge in the network topology serves as an SRG; in this case $SRG'_{res} = SRG$ and $SRG'_{cut}$ is an empty set. Fig. 1.4 demonstrates an example, where the SRG’s circled with dotted lines are covered by a larger SRG, thus can be eliminated.
Figure 1.4: The classification of SRGs into $\text{SRG}_{\text{res}}$ (or $\text{SRG}'_{\text{res}}$) and the $\text{SRG}_{\text{cut}}$ (or $\text{SRG}'_{\text{cut}}$).

Note that entries in $\text{SRG}'$ are not necessarily disjoint; instead, each link can be contained in several entries of $\text{SRG}'$. Let us define a basic set contained in SRGs as $\text{BSS}$, which is a physically disjoint set of network elements most likely composed of a single link or node. The set of $\text{BSS}$'s (denoted as $\text{BSS}$) covers all the links of the network. Thus, each entry in $\text{SRG}'$ can be expressed as the union of some $\text{BSS}$'s with the number of $\text{BSS}$ (denoted as $|\text{BSS}|$) being kept minimal (see also Fig. 1.5). This can be simply calculated in linear time. Note that $|\text{BSS}| \leq |\text{L}|$ since $\text{BSS}$ covers $\text{L}$. If each edge in the network topology serves as an SRG, we have a one-to-one mapping between each $\text{BSS}$ and each entry in $\text{SRG}$.

Figure 1.5: The basic sets of SRGs (BSS) of the network on Fig. 1.4.

To derive the amount of sharable spare capacity along link $j$ demanded by the backup path $W$ (which is denoted by $h_j^W$) can be done by performing matrix operations on the spare provision matrix [44, 1]. However, the study in [44, 1] can only deal with link protection and with the case that each link is treated as an SRG (or called Shared Risk Link Group (SRLG)). To consider the general definition of SRGs and node protection, the mathematical formulation developed in [44, 1] must be solidly expanded. In addition, the study considers dynamic link metrics in the survivable routing process, while [44, 1] simply uses hop count as the metric for working paths, and 0 or $\infty$ for the cost for any spare capacity used by protection paths.

The spare provision matrix is denoted as $\mathbf{S}'$, which is a $|\text{L}| \times |\text{SRG}'|$ matrix and the entry $(j, i)$ of $\mathbf{S}'$ (denoted as $s'_{j,i}$, where $j = 1 \ldots |\text{L}|$, $i = 1 \ldots |\text{SRG}'|$) stores the amount of non-sharable spare capacity along link $j$ for the protection path if the corresponding working path
is involved in the $i^{th}$ entry of $\mathbf{SRG}'$. The most straightforward way of obtaining the matrix of $\mathbf{S}'$ is to simulate the failure of each $\mathbf{SRG}'$ and measure the amount of restoration traffic on each link.

In order to figure out the spare capacity required by the working capacity in each BSS, we define $Z^W$ as the set of SRG’s in which a failure will isolate nodes $s$ or $d$. Note that for the link protection case $Z^W$ is an empty set. We need another matrix $\mathbf{S}'$ (with each entry denoted as $s_{j,t}$, where $j = 1 \ldots |L|$, $t = 1 \ldots |\mathbf{BSS}|$) with a size of $|L| \times |\mathbf{BSS}|$ by converting $\mathbf{S}'$ using the following formula:

$$s_{j,t} = \max_{l \in \mathcal{I}_l} s_{j,l}$$  

(1.2.4)

In Eq. (1.2.4), we take the non-sharable spare capacity along link $j$ required by the working capacity on the $l^{th}$ entry of $\mathbf{BSS}$ as the maximum of those non-sharable spare capacities along link $j$ by the working capacity on the $l^{th}$ SRG, which are in the $l^{th}$ BSS contained in the $l^{th}$ entry of $\mathbf{SRG}'$ except the SRG’s of $Z^W$. We ignore the SRGs of $Z^W$ because $W$ cannot be restored in case of their failure. Note that a special case can be seen for link protection, where we have $|\mathbf{BSS}| = |\mathbf{SRG}|$ and leads to a much easier situation.

Let $D$ be the set of connections already in the network. The amount of non-sharable spare capacity along link $j$ provided that $W$ passes $l^{th}$ BSS of $\mathbf{BSS}$ can be formulated as follows:

$$s_{j,l} = \max_{q \in \mathcal{I}_q} (Q) \cdot b_{q,j} \cdot a_{q,l} \quad \text{for } \forall l \in W$$  

(1.2.5)

where $Q$ represents the $q^{th}$ working path of $D$, while $b_{q,j}$ and $a_{q,l}$ are two binary indicators defined as follows:

$$b_{q,j} = \begin{cases} 1 & \text{if the protection path of } Q \text{ traverse link } j \\ 0 & \text{otherwise} \end{cases}$$  

(1.2.6)

$$a_{q,l} = \begin{cases} 1 & \text{if } Q \text{ passes through the } l^{th} \text{ BSS and this } \\ & \text{BSS is not involved in any SRG of } Z^W \text{ or } Z^Q \\ 0 & \text{otherwise} \end{cases}$$  

(1.2.7)

With Eq. (1.2.5) we can derive the amount of non-sharable spare capacity upon link $j$ required by all the working paths involved in the $l^{th}$ BSS. It is the summation of bandwidth of all the restorable working paths on the $l^{th}$ BSS because the working paths may be subject to a failure at the same moment. In this case, all the affected working paths can be restored provided that the source and destination nodes are not isolated due to the failure (or the BSS subject to the failure does not belong to any $\mathbf{SRG}_{cut}$ of $Z^W$ or $Z^Q$).

With the single failure scenario, only one SRG could possibly be subject to an interruption at a moment. Thus, we can derive $s^W_j$ for $\forall j \in L$, by finding the maximum demand of spare capacity among all the BSS’s traversed by $W$, i.e.,

$$s^W_j = \max_{l \text{ is taken by } W} s_{j,l}$$  

(1.2.8)

The above derivation can be simply written as a matrix expression. We define the working path-SRG incidence matrix as $\mathbf{A}$, which is a $|D| \times |\mathbf{SRG}|$ matrix. The entry at the $q^{th}$ row along the $i^{th}$ column in $\mathbf{A}$ is noted as $a_{q,j}$, which is 1 if the $q^{th}$ working path is involved in the $i^{th}$ SRG, and 0 otherwise. Note that in the above case the failure on the $i^{th}$ SRG must
not isolate the source and the destination node of the $q^{th}$ working path, and must not isolate nodes $s$ or $d$ either. We also denote the backup path-link incidence matrix as $\underline{B}$ with a size of $|D| \times |L|$, where $b_{q,j}$ is the entry of $\underline{B}$ at the $j^{th}$ row and $q^{th}$ column, which is 1 if the $q^{th}$ protection path passes through the $j^{th}$ link in the network. We define $\underline{M}$ as a diagonal matrix of size $|D| \times |D|$ with the entry at the $q^{th}$ row and the $q^{th}$ column being denoted as $m_{q,q}$, which stands for the bandwidth of the $q^{th}$ working path. Note that $m_{i,j} = 0$ if $i \neq j$.

We define $\text{SRG}' - \text{BSS}$ incidence matrix as $\underline{G}$, which is a $|\text{SRG}'| \times |\text{BSS}|$ matrix, and the $g_{i,j}$ entry of $\underline{G}$ is 1 if the $i^{th}$ SRG contains the $j^{th}$ BSS.

$\underline{G}^W$ is defined as a matrix in the same way as $\underline{G}$ except that $\underline{G}^W$ does not cover the columns assigned to the SRGs contained in $Z^W$. Thus we have:

$$\underline{G}^W = (\underline{E} - \sum_{\text{SRG}' \in Z^W} \underline{e}_i \cdot \underline{e}_i^T) \cdot \underline{G} \tag{1.2.9}$$

where $\underline{E}$ is an $|\text{SRG}'| \times |\text{SRG}'|$ unit matrix and $\underline{e}_i$ is a unit column vector, with a size of $|\text{SRG}'|$ (thus $\underline{e}_i^T = \{0, 1, 2, \ldots, \frac{|\text{SRG}'|}{s}, 0\}$ where the $i^{th}$ entry is 1 and all the others are 0). The term $(\underline{E} - \sum_{s \in T} \underline{e}_i \cdot \underline{e}_i^T)$ is a diagonal matrix with all the entries in the diagonal as 1 except for those representing the SRGs contained in $Z^W$. Therefore, the entries in the columns of $\underline{G}^W$ assigned to the SRG of $Z^W$ are 0. Note that in case of link protection $\underline{G}^W = \underline{G}$.

We also define the $\odot$ operator for matrices, which is similar to matrix multiplication. The matrix multiplication of matrix $\underline{A}$ ($n \times l$) and $\underline{B}$ ($l \times m$) is a $\underline{C}$ ($n \times m$) matrix, which can be formulated in the following way:

$$\underline{C} = \underline{A} \odot \underline{B} \quad \text{where} \quad c_{i,j} = \sum_{k=1}^{l} a_{i,k} \cdot b_{k,j}$$

while in case of the $\odot$ operator a maximum function is taken on the $a_{i,k} \cdot b_{k,j}$ values:

$$\underline{C} = \underline{A} \odot \underline{B} \quad \text{where} \quad c_{i,j} = \max_{1 \leq k \leq l} |a_{i,k} \cdot b_{k,j}| \tag{1.2.10}$$

The advantage of the operator is that we can formulate the $\underline{S}$ spare provision matrix for $W$ as (see also 1.6):

$$\underline{S} = (\underline{B}^T \cdot \underline{M} \cdot \underline{A}) \odot \underline{G}^W \tag{1.2.11}$$

$\underline{S}$ is a $|L| \times |\text{BSS}|$ matrix. $\underline{B}^T \cdot \underline{M} \cdot \underline{A}$ is a $|L| \times |\text{SRG}'|$ matrix representing the spare capacity routed on link in case of a failure of upon each SRG. Thus, the column assigned to the $j^{th}$ BSS takes the maximum of the columns assigned to the SRGs containing the $j^{th}$ BSS except for the SRGs contained in $Z^W$ (please refer to Eq. (1.2.9) for verification). Note that the spare provision matrix does not depend on $W$ in case of protecting link failures only since $\underline{G}^W = \underline{G}$. Let us define a vector $\underline{W}$ of size $|1| \times |L|$ as the working path-link incidence vector, where $\underline{W} = \sum_{l \in W} \underline{e}_l^T$. An example is given as follows: $\underline{W} = \{1, 2, 3, x-1, x, x+1, \ldots, |L|\}$ depicts that $W$ traverses through two links, namely the second and the $x^{th}$ in the network. Let us define in a similar way the backup path-link incidence vector $\underline{P}$ with a size $|1| \times |L|$, and a vector $\underline{W}'$ of size $|1| \times |\text{BSS}|$ that stores the BSS’s of $W$. 


Applying a \( \max \) operation upon each column of \( S \) corresponding to the \( BSS \)s traversed by \( W \) will yield a \( 1 \times |L| \) vector \( S^W \), which keeps the amount of non-sharable spare capacity along each link provided with \( W \). It can be formulated as:

\[
S^W = S \odot W'
\]  

(1.2.12)

The \( 1 \times |L| \) vector \( H^W \), which stores the amount of sharable spare capacity along each link provided with \( W \), can thus be derived by referring to the relationship

\[
H^W = V - S^W
\]  

(1.2.13)

where \( V \) is a \( 1 \times |L| \) vector recording the amount of spare capacity along each link.

Finally we can define the sharable spare capacity matrix \( H \) of size \( |L| \times |BSS| \). We have know that the \( (j,l) \) entry of \( S \) stores the amount of sharable spare capacity along link \( j \) provided for protection path if the corresponding working path is involved in the \( j \)th \( BSS \). It can be expressed as

\[
h_{j,l} = v_j - s_{j,l}
\]  

(1.2.14)

We also define a matrix called \( R \). The entry \( (j,l) \) (i.e., \( r_{j,l} \)) standing for the ratio of the bandwidth required to be allocated as spare capacity along link \( j \) if \( W \) passes through the link \( l \) (similarly to 1.2.2). The ratio can be a negative number if there is more sharable spare capacity than the required bandwidth, in which case \( r_{j,l} \) is set to 0 since we need to allocate 0 bandwidth; thus we add a \( \max \{, 0 \} \) function to the formulation,

\[
r_{j,l} = \max \left\{ 1 - \frac{1}{b(W)} \cdot h_{j,l}, 0 \right\}
\]  

(1.2.15)

### 1.2.9 Cost Function in Matrix Form

With the knowledge of \( S \) (or \( H \)), the cost functions for both working and protection paths can be formulated in a matrix form. This formulation does not use \( \infty \) as a cost, thus it is only valid for feasible solutions. See Section 1.3 for feasibility conditions. Let us define a column vector, denoted as \( C \), for the cost along each link taken by \( W \). \( C \) is a vector of size \( |L| \) where the \( j \)th entry (denoted by \( c_j \)) is a non-negative value. Let us define a matrix \( W \) representing
\( W \), where \( W \) is a diagonal matrix of a size \(|L| \times |L|\) such that \( \text{diag}(W) = W \). In the same way, a matrix \( P \) is defined for \( P \) such that \( \text{diag}(P) = P \).

We also define a special matrix norm, which will be used in the rest of this dissertation. It is similar to \( \|A\|_1 = \max \sum_{j} |a_{i,j}| \) but the sum and max operations are swapped:

\[
\|A\| = \sum_{i=1}^{n} \max_{j} |a_{i,j}|
\] (1.2.16)

where \( n \) is the number of rows of \( A \). This is also a valid matrix norm, due to the following:

\[
\|A\| > 0 \text{ if } A \neq 0, \text{ and } \|0\| = 0
\] (1.2.17)

\[
\|c \cdot A\| = |c| \cdot \|A\|
\] (1.2.18)

\[
\|A + B\| \leq \|A\| + \|B\|
\] (1.2.19)

\[
\|A \cdot B\| \leq \|A\| \cdot \|B\|
\] (1.2.20)

Eq. (1.2.17) and Eq. (1.2.18) trivially holds, while Eq. (1.2.19) holds for the following reason:

\[
\|A + B\| = \sum_{i=1}^{n} \max_{j} |a_{i,j} + b_{i,j}| \leq \sum_{i=1}^{n} \max_{j} (|a_{i,j}| + |b_{i,j}|) \leq \sum_{i=1}^{n} \max_{j} |a_{i,j}| + \max_{j} |b_{i,j}| = \sum_{i=1}^{n} \max_{j} |a_{i,j}| + \sum_{j=1}^{m} \max_{j} |b_{i,j}| = \|A\| + \|B\|
\]

where the first \( \leq \) stands for the fact that \( |a_{i,j} + b_{i,j}| \leq |a_{i,j}| + |b_{i,j}| \), and the second is for the fact that \( \max_{j} (|a_{i,j}| + |b_{i,j}|) \leq \max_{j} |a_{i,j}| + \max_{j} |b_{i,j}| \). The proof of Eq. (1.2.20) is as follows:

\[
\|A \cdot B\| = \sum_{i=1}^{n} \max_{j} \sum_{k=1}^{m} |a_{i,k} \cdot b_{k,j}| = \sum_{i=1}^{n} \max_{j} \sum_{k=1}^{m} |a_{i,k}| \cdot |b_{k,j}| \leq \sum_{i=1}^{n} \max_{j} \sum_{k=1}^{m} |a_{i,k}| \cdot |b_{k,j}| = \sum_{i=1}^{n} \max_{j} |a_{i,j}| \sum_{k=1}^{m} \cdot |b_{k,j}| \leq \sum_{i=1}^{n} \max_{j} |a_{i,j}| \cdot \sum_{k=1}^{m} \max_{j} |b_{k,j}| = \|A\| \cdot \|B\|
\]

where \( m \) is the number of rows of \( B \). The first \( \leq \) stands for the fact that \( |a_{i,k}| \leq \max_{l} |a_{i,l}| \), and the second is for the fact that \( \max_{j} \sum_{k=1}^{m} |b_{k,j}| \leq \sum_{k=1}^{m} \max_{j} |b_{k,j}| \).

To formulate the cost of allocating \( W \) we need to sum up the cost along each link taken by \( W \):
\[ c(W) = b(W) \cdot \| C^T \cdot W \| \]  

(1.2.22)

Since \( C \) is a diagonal matrix, we have \( C^T = C \). Eq. (1.2.22) holds because \( C^T \cdot W \) (with a size \(|L| \times |L|\)) can be expressed as \( \sum_{k=1}^{|L|} c_{i,k} \cdot b_{k,j} \) and \( c_{i,k} = 0 \) if \( i \neq k \). Thus \( C^T \cdot W \) is nothing but a diagonal matrix with each entry as \( c_{i,i} \cdot b_{i,j} \) for \( i = 1 \) to \(|L|\). The norm of the matrix can be expressed as \( \| C^T \cdot W \| = \sum_{i=1}^{|L|} \max_j |c_{i,j} \cdot b_{i,j}| = \sum_{i=1}^{|L|} |c_{i,i} \cdot b_{i,i}| \), which is nothing but the total cost of \( W \) if we multiply it by the bandwidth of the connection \((b(W))\) as was defined at Eq. (1.2.1).

With these matrices the total cost of \( P \) can be expressed as:

\[ c(P) = b(W) \cdot \| P^T \cdot P^T \cdot R \cdot W' \| \]  

(1.2.23)

since \( P \) is a diagonal matrix, we have \( P^T = P \). Eq. (1.2.23) holds since \( P^T \cdot P \cdot W' \) is a matrix with the \((i,j)\) entry as \( \sum_{k=1}^{|L|} \sum_{l=1}^{|BSS|} p_{i,k} \cdot r_{k,l} \cdot w_{l,j} \). It is multiplied by a diagonal matrix \( P^T \) from left, which is equivalent to the case that each row of the matrix is multiplied with \( c_{i,i} \). Thus, the \((i,j)\) entry in \( P^T \cdot P^T \cdot R \cdot W' \) is \( \sum_{k=1}^{|L|} \sum_{l=1}^{|BSS|} c_{i,i} \cdot p_{i,k} \cdot r_{k,l} \cdot w_{l,j} \), which is a matrix with entries of \( c_{i,i} \cdot r_{k,l} \) if the \( k \)th link is a part of \( P \) and the \( l \)th BSS is traversed by \( W \), and 0 otherwise. The norm of this matrix equals to:

\[
\| C^T \cdot P^T \cdot R \cdot W' \| = \sum_{i=1}^{|L|} \max_j \sum_{k=1}^{|L|} \sum_{l=1}^{|BSS|} c_{i,i} \cdot p_{i,k} \cdot r_{k,l} \cdot w_{l,j} =
\sum_{i=1}^{|L|} \max_j \sum_{l=1}^{|BSS|} c_{i,i} \cdot p_{i,i} \cdot r_{i,l} \cdot w_{l,j} =
\sum_{i=1}^{|L|} \max_j c_{i,i} \cdot p_{i,i} \cdot r_{j,j} \cdot w_{j,j} =
\sum_{i \in P} \sum_{j \in W} c_{i,i} \cdot r_{i,j} =
\sum_{i \in P} c_{i,i} \cdot \max_{j \in W} r_{i,j} =
\sum_{i \in P} c_{i,i} \cdot r_{j}^{W}
\]

where the second equality is true, due to the fact that \( P \) is diagonal (i.e., \( p_{i,k} = 0 \) if \( i \neq k \)) and the third equality is true since \( W \) is diagonal (i.e., \( w_{l}^{l} = 0 \) if \( l \neq j \)). Finally, the last equality in the above equation can be verified by observing Eq. (1.2.8). After multiplying the norm by \( b(W) \) we get the same cost function defined in Eq. (1.2.3).

Summarizing this section the total cost can be expressed as:

\[ c_{\text{total}} = c(W) + c(P) = b(W) \cdot (\| C^T \cdot W \| + \| C^T \cdot P^T \cdot R \cdot W' \|) \]  

(1.2.24)

### 1.3 Problem Definition

Finally, we will define the problem of Shared Path Protection. Given a network with some already established connections and with a new demand. With load balancing functions (see Section 1.2.6) a cost function is calculated on the links. A directed graph is built up representing
the network as it was described in Section 1.2.4 or 1.2.5. Since each arc of the graph belongs to a network element, the SRGs and all notations defined in sections 1.2.7, 1.2.8, 1.2.9 can be defined on the arcs of the graph as well. Those arcs of the graph are erased where the free capacity plus the spare capacity is less than the bandwidth of the new demand (arc \( j \) is erased if \( f_j + v_j < b(W) \)). The \( S \) matrix is calculated as it was described in Section 1.2.8. The routing problem can be defined as:

**Given:**
- a directed graph \( G(V, A) \), with \( V \) and \( A \) being the set of vertices and arcs, respectively,
- the free capacity \( (f) \) and, the spare capacity \( (v) \) of each arc,
- the SRGs of \( G \) (sets on the arcs of \( G \)),
- the \( S \) matrix (it can be assigned to arcs as vectors of size \( |BS| \),
- the source node \( s \) and the destination \( d \) and the bandwidth \( b(W) \) of the new demand,

**Find:**
- working path \( W \),
- protection path \( P \),
- \( W \) and \( P \) should be SRG disjoint,
- the feasible condition of the working path is \( f_i \geq b(W) \) for \( \forall i \in W \)
- the feasible condition of the protection path is \( f_i + v_i \geq b(W) + \max_{v_j \in W} s_{i,j} \) for \( \forall i \in P \).
- such that

\[
\text{Minimize: } b(W) \cdot (\|C^T \cdot W\| + \|C^T \cdot P^T \cdot R \cdot W'\|)
\]
1.4 Claim 1.1: Two-Step-Approach with Asymmetrically Weighted Disjoint Path-Pairs

The most widely used shared protection heuristic is the Two-Step-Approach [1, 4, 6, 46]. In the first step the shortest path is routed as the working path. In the second step, an SRG-disjoint protection route is selected to permit sharing of the protection capacity on links of the protection route, if the working paths belonging to them have no SRGs in common. This scheme is currently favored in IETF deliberations for MPLS-layer protection and MPLS-controlled optical path protection [46]. Its advantage in distributed implementation, and the protocol is less complex [1].

From an algorithmic aspect, in the first step the costs \(c^W_j\) for all arcs \(j\) are assigned according to Eq. (1.2.1) and finding the shortest path can be done in linear average-case time [15]. In the second step the costs \(c^P_j\) for all arcs \(j\) of \(G\) are assigned according to Eq. (1.2.3) and finding a protection path with minimum capacity allocation (with optimum capacity sharing) can also be done by a shortest path search.

Ideally the whole protection path can be shared and the required capacity allocation equals zero. Calculating the edge cost needs \(O(\max(m, p))\) time, where \(m\) denotes the number of edges and \(p\) denotes the number of connections.

One of the big disadvantages of the two-step approach is the existence of trap topology. In trap topology the working path may block all the possible SRG-disjoint protection paths even though the network topology is two connected. In other words the second step may fail after choosing a bad working path, since no other SRG-disjoint path is available in the network (see also simulation results in [47],[46]). The main idea of the claim is to solve this dilemma.

In the rest of this section we will refer to any routing algorithm which calculates the working and protection paths in two steps as the two-step-approach. The original Two-Step-Approach, which assigns the shortest path as a working path, will be referred to "2D" (Two Dijkstra's). For easier understanding in this section we will assume to have an MPLS network model and the SRGs will be the links of the network. In other words the network is protected against link failure.

One possible solution is to use Suurballe’s algorithm [17, 25], which derives a disjoint path pair with a minimum total bandwidth allocation. Choosing the shorter route as a working path, the protection path is guaranteed. Note that this is not a sufficient condition, therefore, even if Suurballe’s algorithm fails there can be a disjoint working and a shared protection path available in the network.

Intuitively we would like to have a shorter working path, since the whole working path will be allocated in the network, while the protection route may be shared with other protection routes. Therefore the need for an algorithm, which derives a disjoint path-pair with the working path as short as possible. In [32] a problem called the asymmetrically weighted disjoint path-pair problem was defined in the following way:

\[
\text{minimize: } \alpha \cdot \text{cost}(W) + \text{cost}(P),
\]

where the cost of the working path is denoted by \(\text{cost}(W)\), the cost of the protection path is denoted by \(\text{cost}(P)\) and \(\alpha\) is a given parameter. This includes the above mentioned two
proposals (α = 1 and α → ∞) as well.

Consequently, a new two-step approach called the "alpha method" is introduced, where in the first step an asymmetrically weighted path-pair is calculated with a pre-defined α parameter. In the next step an SRG-disjoint protection route is selected to permit sharing of protection capacity on links of the protection route, if the working paths belonging to them has no SRGs in common. The new two-step approach successfully solves the trap topology dilemma and with a well adjusted alpha parameter it is proven to have good performance.

To calculate the cost of the connection as it was defined in Section 2.2.1, we need a matrix $R$ that has all entries set to $r_{i,j} = \frac{1}{\alpha}$. According to Eq. (1.2.23) we get the following cost for protection route:

$$C(P) = b(P) \cdot \|C^T \cdot P^T \cdot R \cdot W\| = b(P) \cdot \sum_{i \in P} c_{i,i} \cdot \max_{j \in W} r_{i,j} = b(P) \cdot \sum_{i \in P} c_{i,i} \cdot \frac{1}{\alpha} \quad (1.4.1)$$

This is the same as using alpha method. However, in order to reduce the capacity need of the connection in the alpha method in the second step, a new protection route is calculated based on the working route. Therefore, a parameter $\alpha$ is chosen as the mean value of the entries of matrix $R$ ($\alpha = \sum_{i} \sum_{l} \frac{|BSS| \cdot |L|}{r_{i,l}}$). The mean value of the entries of matrix $R$ [based on Eq. (1.2.15)] can be expressed as:

$$\frac{1}{\alpha} = \sum_{i} \sum_{l} \frac{r_{i,l}}{|BSS| \cdot |L|} = \sum_{i} \sum_{l} \frac{1}{|BSS| \cdot |L|} \cdot \max \left\{ 1 - \frac{1}{b(W)} \cdot h_{j,l}, 0 \right\} \quad (1.4.2)$$

1.4.1 Integer Linear Programming Formulation

In this subsection, an Integer Linear Programming formulation is presented for solving the optimal diverse routing problem, where a commercially available ILP solver (i.e., CPLEX 7.5 in this case) is adopted to jointly determine the optimal SRG-disjoint working and protection path-pair upon a connection request according to the current link-state. This ILP formulation is more general than the one published in [48], since it allows us to address all type of SRG constraints. The other reason to include this formulation is to make the understanding of the ILP formulations of the next sections easier.

The following symbols are adopted to develop the ILP formulation. The input of the problem with notations was described in Section 1.3. Two residual graphs are defined to facilitate the solving of this problem, each of which carries one or a few variables for the identification of the working and protection path-pairs. The graph for solving the working path is denoted as $G_w(V, A_w)$ and is composed of arcs with $f_j \geq b(W)$ for $j \in A_w$ (variable $x$ in the following formulas are assigned to this graph). The second residual graph is denoted as $G_p(V, A_p)$, which is to facilitate solving the protection path. We need this graph to record the spare link-state because the working and the protection paths take different suites of link-state with shared protection. This graph is composed of the links where the amount of free capacity $f_j$ plus that of the spare capacity $v_j$ is larger than or equal to $b(W)$ i.e., $b(W) \leq f_j + v_j$ for $j \in A_p$. Let $x_a$ (or $y_a$) be the binary variable which is 1 if the flow for the working (or protection) path passes through the arc $a$. Let $z_a$ be a non-negative real variable assigned to the spare link-state. It
is used to derive the cost for the protection path to take link \( a \). Note that \( z_a \) is dependent on the location of working flows (i.e., \( x_a \)). A small non-zero additional cost \( \epsilon \) is imposed when the protection path consumes sharable spare capacity to minimize its length, which was defined in Section 1.2.7.

The integer programming formulation is as follows:

Object function:

\[
\text{minimize } \sum_{a \in A} (x_a + z_a)e_a + \epsilon \cdot y_a \quad (1.4.3)
\]

which is subject to the following constraints

1. The flow conservation constraints for all nodes \( i \in V \):

\[
\sum_{(i,j) \in A} x_{i,j} - \sum_{(j,i) \in A} x_{j,i} = \begin{cases} 
1 & \text{if } i \text{ is the source node } s \\
-1 & \text{if } i \text{ is the destination node } d \\
0 & \text{otherwise}
\end{cases} 
\]

\[
\sum_{(i,j) \in A} y_{i,j} - \sum_{(j,i) \in A} y_{j,i} = \begin{cases} 
1 & \text{if } i \text{ is the source node } s \\
-1 & \text{if } i \text{ is the destination node } d \\
0 & \text{otherwise}
\end{cases} 
\quad (1.4.4)
\]

2. The constraint that working and protection paths should be SRG-disjoint:

\[
n \cdot x_a + \sum_{\forall b \in A \text{ in the same } \text{SRG} \notin W} y_b \leq n \text{ for } \forall a \in A
\]

(1.4.5)

The value of the constant \( n \) is larger than the maximum number of arcs in the SRGs (e.g., it can be set to \( |V| \)). In the summation all SRGs are considered except all \( \text{SRG} \notin W \), which failure separates \( s \) and \( d \), since the failure of those SRGs cannot be protected.

3. The constraint that both flows for working and protection paths should be integer (the flow integrity constraint):

\[
x_a \in \{0, 1\}, y_a \in \{0, 1\}, z_a \geq 0 \text{ for } \forall a \in A
\]

(1.4.6)

4. The SRG constraints:

\[
x_a + y_e - 1 - \frac{h_{e,b}}{b(W)} \leq z_e \quad \forall e \in A_P, \forall a \in A_w
\]

\[
a \in b \in \text{BSS} \text{ and } h_{e,b} + f_e \geq b(W) \quad (1.4.7)
\]

\[
\sum_{\forall a \in b, a \in A_w} x_a + n \cdot y_e \leq n \quad \forall e \in A_P, \forall b \in \text{BSS} \text{ and } h_{e,b} + f_e < b(W) \quad (1.4.8)
\]

In the above formulation, Eq. (1.4.4) is for the flow conservation constraint, which ensure the path requirement for both the working and protection paths. Eq. (1.4.5) and (1.4.6) are to ensure the SRG-disjointness of the working and protection paths, and the integrity of the traffic flows, respectively. Eq. (1.4.7) states that the cost for the protection path taking arc \( e \) has a lower bound of \( x_a + y_e - 1 - h_{e,b} \) for all arcs \( e \) and \( a \), such that arc \( a \) is involved in BSS \( b \), where \( H \) is the sharable spare capacity matrix defined in Section 1.2.8. Eq. (1.4.8) states that if \( h_{e,b} + f_e < b(W) \), link \( a \) and \( e \) cannot be used at the same time for \( W \) and \( P \) since there would be insufficient free and spare capacity for \( P \) along \( e \) given that \( W \) takes \( b \).
1.4.2 Performance Evaluation

The numerical tests have been carried out on a 950 MHz AMD Athlon processor, with 512 Mbytes of RAM running operating system suse Linux 6.3 with kernel version 2.2.17. The program has been written in C++. Four different test networks were adopted: N16, N22, N30 and N61. The networks are named after the number of nodes. Networks N16, N30, N61 are designed by a network designer program (see Figure 1.12), while network N22 is a European backbone network (as shown in Figure 1.11). See also Table 1.1 for their properties. Their costs used for load balancing are written next to the edges.

<table>
<thead>
<tr>
<th>Name</th>
<th>N16</th>
<th>N22</th>
<th>N30</th>
<th>N61</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. nodes</td>
<td>16</td>
<td>22</td>
<td>30</td>
<td>61</td>
</tr>
<tr>
<td>No. of edges</td>
<td>27</td>
<td>43</td>
<td>63</td>
<td>114</td>
</tr>
<tr>
<td>Avg. hop counts</td>
<td>2.3</td>
<td>2.9</td>
<td>3.6</td>
<td>4.5</td>
</tr>
<tr>
<td>Avg. nodal degree</td>
<td>3.4</td>
<td>3.9</td>
<td>4.2</td>
<td>3.7</td>
</tr>
</tbody>
</table>

Table 1.1: The properties of the networks.

The algorithms are compared by better overall performance what is measured by the average and the maximal waiting time of establishing connections. If a connection request cannot be established immediately, it has to wait until some other connections are taken down and the required spare capacity is available. Note that it does not necessary mean that other connections are not established during this waiting period. We have also evaluated the average of network utilization and the average of shared capacity allocation for all routing algorithms. On the charts the network load (the average of working and protection capacity utilization) and working load (the average of working capacity utilization) were normalized by the total network capacity.

Dynamic traffic pattern was generated in advance; therefore, all algorithms got exactly the same connection request at the same time. The capacity of the network was set to be able to cope with the traffic. The waiting times for each connection were set unlimited, therefore no connection were blocked.

Four different algorithms, altogether with nine different settings were compared. The first algorithm optimizes the working and the protection path jointly (in one step). All the others are two-step approaches. The first algorithm is based on ILP solver (it is called ILP-opt on the charts). It was introduced in Section 1.4.1 and CPLEX 7.5 was used for solving the ILP formulation. Unfortunately, it was not able to cope with our networks consisting of more than 22 nodes. The second algorithm is based on OSPF routing and the shortest path is assigned as a working path (2D). The third algorithm is based on Suurballe's algorithm by keeping the shorter route as the working path (it is called SUURB on the charts). If it fails to find a disjoint path-pair, the shortest path is taken as a working path. The next one is based on the algorithm, which solves the asymmetrically weighted node-disjoint path-pair problem, giving five different values to the parameter of $\alpha$: $\alpha = 2$, $\alpha = 3$, $\alpha = 4$, $\alpha = 5$, $\alpha = 100$. Similarly to SUURB, if it fails the shortest path is taken as a working path. The last one is an improvement of 2D: if establishing the working path fails, then it is weighted with $\alpha = 100$ of the
asymmetrically weighted node-disjoint path-pair is kept as a working path (it is called "2D + α = 100" on the charts).

The difference of routing algorithms is not significant on small networks, because the number of different routes is limited (see simulation results of N30 at Figure 1.7 and N61 at Figure 1.8).

Simulation results of network N61 are showed in Figure 1.8. Due to the good load balancing function the network utilization reached an average of ~ 75%, 2D was able to reach 88% total load of the network, which became a disadvantage, because its waiting time increased significantly.

---

Figure 1.7: Simulation results of the 30 node network where 7847 connections were established

Figure 1.8: Simulation results of the 61 node network where 2599 connections were established

More simulation results were conducted, where the value of α was set according to Eq. (1.4.2). The obtained results were similar for all topologies and N30 was chosen for further analysis.
Fig. 1.9 shows the evaluated value of $\alpha$ along the simulation. The $x$-axis of the chart is the time step of the simulation. There are three lines drawn on the charts; a thick line, which represents the obtained $\alpha$ value, and two thin lines for the network load and work load. The simulation started with an empty network and by the end of the simulation the network load reached 65% and the work load 45%. The evaluated $\alpha$ was usually between 4 and 5 - as accepted based on the simulation results of 1.7. Fig. 1.10 shows the simulation results of three methods; the 2D (OSPF on the chart), the alpha method and ILP-opt, which calculates the optimal solution according the the current network state. On four different network loads, four different aspects were compared; the network load, the work load, the average waiting time and the runtime. In this simulation ILP was used for implementing the alpha method, in order increase the accuracy in the performance of any heuristic approach for the asymmetrically weighted disjoint path-pair problem. As a consequence the runtime of the alpha method becomes significantly higher when compared to 2D (OSPF). For the average waiting time the ILP-opt out-performed the other two methods, however due to the enormously high computational time, it is not preferred for realtime routing. On the other hand, the alpha method is a good compromise of runtime versus performance.

![Graph showing network load, alpha, and work load over time.](image)

Figure 1.9: The value of $\alpha$ for N30 compared with the network load.

### 1.4.3 Conclusion

In this section several routing algorithms were compared for establishing both working and their shared protection paths in mesh telecommunication networks. If there are insufficient network resources to set up the connection, it waits until some capacities of the network become free. Traffic pattern was generated in advance to provide exactly the same traffic conditions for each algorithm. In our simulations four different routing algorithms were examined:

2D is based on OSPF routing, where the shortest path is chosen as a working path, and in the next step an optimal shared protection path is calculated. The first step of 2D is a "greedy" method, and in the simulations it had the highest network load. Unfortunately, it often fails establishing a disjoint shared protection path. As a consequence 2D was poor in fairness and had a longer connection waiting time.

A possible improvement of 2D is $2D + \alpha = 100$, which tries to reduce the fail rate of 2D by
Figure 1.10: Simulation results of the 30 node network where 1598 connections were established running $\alpha = 100$ when 2D fails. It improves the performance of 2D a little bit, but $\alpha = 100$ alone is even better although 2D is approximately three times faster than $\alpha = 100$.

Deriving working and protection paths jointly was implemented with ILP, which was able to achieve the highest sharing and capacity efficiency, however its runtime is more than 100 times larger than 2D.

Another possible improvement of 2D to calculate a minimal cost disjoint path-pair and choose the shorter path as working path (SUURB). It can be longer than the shortest path, but it never fails to find the shared protection path. In the simulations the ratio of the allocated capacity to working paths and the total allocated capacity was higher compared to other methods. It means, we allocate less capacity for shared protection. The reason for this is that SUURB derives shorter protection routes than other routing algorithms.

The method of alpha is based on finding asymmetrically weighted node-disjoint path-pairs. This method is a generalization of the greedy 2D and the generous SUURB in terms of the length of the working path. Compared to 2D it never fails to find the shared protection path. A novel method was introduce to tune the value of $\alpha$, with which it slightly outperformed the traditional methods.
Figure 1.11: The 22-nodes European backbone network.
Figure 1.12: The networks generated by a network designer program (N16, N30 and N61).
1.5 Claim 1.2: Two-Step-Approach with Maximum Likelihood Relaxation (MLR)

1.5.1 Introduction

Due to the dependency of the protection path on its working path, the ILP formulation of Section 1.4.1 is not scalable with the network size. Therefore, a heuristic algorithm called Maximum Likelihood Relaxation (MLR) is introduced, which aims to explore the approximating optimal solutions with less computation time. We evaluate the performance of the scheme and make a comparison with some reported counterparts. The simulation results show that MLR yields a faster path selection process, which initiates a compromise between computation efficiency and performance.

1.5.2 Maximum Likelihood Relaxation (MLR)

This subsection introduces the MLR scheme, which is a modified Dijkstra’s algorithm carrying/handling some additional information during Dijkstra’s relaxation process. The main idea for MLR is that the working path is selected such that the number of links without enough sharable spare capacity and working link cost are jointly considered. The links in the network with enough sharable spare capacity for the protection path of a working path segment, \( \hat{w} \), are called as “Easy Links” of \( \hat{w} \). During Dijkstra’s relaxation process, when vertex \( n \) is given a temporary label through arc \((x, n)\), by vertex \( x \), a new working path segment from the source vertex \( s \) to vertex \( n \) by way of vertex \( x \) (denoted as \( \pi(s, x) \cup (x, n) \)) is formed. We have the following relationship:

\[
s_j^{\pi(s, x) \cup (x, n)} \geq s_j^{\pi(s, x)} \quad \forall j \in A \quad \text{and} \quad \forall n \in V \quad \text{is not on the path segment} \pi(s, x) \quad (1.5.1)
\]

We refer to Eq. (1.2.8) for the definition of \( s_j^{\pi(s, x) \cup (x, n)} \) and \( s_j^{\pi(s, x)} \). Eq. (1.5.1) holds due to the reason that when vertex \( x \) gives a temporary label to vertex \( n \) and the resultant path segment \( \pi(s, x) \cup (x, n) \), the working paths passing through arc \((x, n)\) are newly included into the SRG, which yields a fact that some sharable spare capacity in the network may become non-sharable along some Easy Links for \( \pi(s, x) \).

Based on the above discussion, it is clear that during the relaxation process, the amount of sharable spare capacity and the number of Easy Links decreases. Therefore, one of the objectives in the proposed Dijkstra’s relaxation process is to find a working path maximizing the number of Easy Links. In addition, we need to consider the cost of each link, \( c_a \), for \( a \in A \), such that having a long working path is discouraged. In this study, the label (denoted as \( l(n) \) for vertex \( n \)) given by vertex \( x \) in Dijkstra’s relaxation process is defined as the link cost \( c_{x,n} \) divided by the log of the number of Easy Links for \( \pi(s, n) \). The label replacement at vertex \( n \) by vertex \( x \) will be conducted in such a way that \( l(n) \) is minimal; i.e.,

\[
l(n) = \min\{(l(n), l(x) + \frac{c_{x,n}}{\log(\text{offset}(x, n) + 1)})\}, \quad (1.5.2)
\]

where \( \text{offset}(x, n) \) is the reduction on the number of Easy Links for \( \pi(s, x) \cup (x, n) \). We have an expression for \( \text{offset}(x, n) \) as follows:
\[
\text{offset}(x, n) = \left\{ \sum_{j \in A} \text{stp}(\varphi_j^{W'}) : \varphi_j^{W'} \leftarrow v_j - \max_{l \in W'} s_{i,j} - b(W), \text{ where } W' = \pi(s, x) \right\} - \\
\left\{ \sum_{j \in A} \text{stp}(\varphi_j^{W'}) : \varphi_j^{W'} \leftarrow v_j - \max_{l \in W'} s_{i,j} - b(W), \text{ where } W' = \pi(s, x) \cup (x, n) \right\}
\]

(1.5.3)

where the function \( \text{stp}(x) \) returns 1 if \( x \geq 0 \), and 0 otherwise; \( v_j \) is the spare capacity along \( j \) and can be derived in the link-state database; \( s_{i,j} \), as defined in Section 1.2.8, is the amount of protection capacity routed on \( j \) which protect the working capacity along \( l \) before the arrival of the current connection request, and is supposed to be non-sharable if \( W \) traverses in the same SRG as \( l \). Both \( v_j \) and \( s_{i,j} \) can be derived off-line (i.e., before the connection request is launched).

The algorithm proceeds as follows: at the beginning, \( l(n) = 0 \) for \( n = s \) and \( l(n) = \infty \) otherwise. The extra labels required to be recorded for Dijkstra's relaxation process is an \( |A| \times |\pi(s, x)| \) array storing \( s_{i,j} \), where \( l \in \pi(s, x) \) and \( j \in A \). When the relaxation process attempts to replace the label of vertex \( n \) by that of vertex \( x \), the “MAX” operation for the array storing \( s_{i,j} \) will have to be performed for \( l \in \pi(s, x) \) and \( l \in \pi(s, x) \cup (x, n) \) so that Eq. (1.5.2) can hold. In addition, when a node is relaxed, the array storing \( s_{i,j} \) is also updated. After the derivation of the working path, the corresponding protection path can be derived by Two-Step-Approach defined in the previous section.

The computational complexity in implementing the MLR algorithm is \( O(|A| \cdot |V|^2 \cdot \log |V|) \). To see its detail, the complexity for performing the regular Dijkstra's algorithm yields \( |V| \cdot \log |V| \). An extra computational effort to scan and update the array of \( s_{i,j} \) is required each time when a temporary label is sent, which yields computational complexity \( O(|A| \cdot |V|) \). Therefore, the computational complexity for the MLR scheme is \( O(|A| \cdot |V|^2 \cdot \log |V|) \) in the worst case.

The MLR method cannot guarantee the derivation of the best working and shared protection path-pair. However, the computational efficiency can be tremendously improved compared with both of the other two proposed schemes. We will verify the schemes with simulation in the subsequent section.

### 1.5.3 Performance Evaluation

Experiments are conducted to verify the ILP formulation (denoted as ILP in the following context) and MLR algorithms on four networks with 22, 30, 79, and 100 nodes with Ultra-80 SUN workstations. The network topologies are shown in Fig. 1.13. Assume each directional link in the network supports 32 independent connections. We first examine the capacity efficiency in terms of blocking probability for the dynamically arrived connection requests following the Poisson model and a holding time with an exponential distribution function. Let node pair \((i, j)\) be subject to traffic load \( \rho_{i,j} = \eta \cdot (\lambda_{i,j}/\mu_{i,j}) \), where \( \lambda_{i,j} \) and \( \mu_{i,j} \) are arrival and departure rate upon the node pair \((i, j)\), respectively. \( \mu_{i,j} \) is set to 1, while \( \lambda_{i,j} \) is a predefined number between 0.5 - 1.5 stored in a traffic matrix. The scaling parameter \( \eta \) represents the level of traffic load in the network with the unit of Erlang. Therefore, each node pair yields different bandwidth demand. Without loss of generality, every connection request is a single lightpath that occupies a wavelength channel as traversing through the corresponding link.
An other method, called Iterative Two-Step-Approach (ITSA) [3], was also developed, which iteratively takes $k$-shortest path for inspection, and can guarantee the derivation of the best solution. For the ITSA scheme, in order to limit the computation time, the maximum number of allowable iterations for solving a connection request is 50 for each case.

For comparison, we also implement the scheme provided in [6] (denoted as APF-PBC) and the scheme without any resource sharing (denoted as NS). With APF-PBC, the cost function for the working path is $c(W) = f_1 \cdot (1 + \frac{1}{\max r_{i,j}})$, where $f_1$ is the residual bandwidth of link $l$, $s_{i,j}$ is the total spare capacity required along link $j$ to protect the working capacity along link $i$. With NS, Suurballe’s algorithm [24] is adopted to find the least-cost working and protection path-pair in the network, where resource sharing is not allowed.

![Figure 1.13: 79-node and 100-node network topologies.](image)

<table>
<thead>
<tr>
<th>Network</th>
<th>N10</th>
<th>N22</th>
<th>N30</th>
<th>N79</th>
<th>N100</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of nodes</td>
<td>10</td>
<td>22</td>
<td>30</td>
<td>79</td>
<td>100</td>
</tr>
<tr>
<td>No. of links</td>
<td>28</td>
<td>88</td>
<td>126</td>
<td>216</td>
<td>358</td>
</tr>
<tr>
<td>Nodal degree</td>
<td>2.8</td>
<td>4.0</td>
<td>4.2</td>
<td>2.73</td>
<td>3.58</td>
</tr>
</tbody>
</table>

Table 1.2: Topology of the networks adopted in the simulation. All the topologies are two-connected mesh networks.

Fig. 1.12(a) and (b) and Fig. 1.13 (a) and (b) show the simulation results for all the schemes in terms of blocking probability on the four network topologies. The properties of the networks are summarized in Table 1.2. Since solving the ILP takes computation time much longer than the cases of the other heuristics, we only verify ILP with the 22-node and 30-node networks as shown in Fig. 1.12(a) and (b). The computation time for solving the ILP formulation on the 22-node and the 30-node networks is around several seconds and 10~20 seconds by using CPLEX 7.5 on an Ultra 80 SUN Workstation with 4 GB memory. However, it may take more than half an hour to allocate a working and protection path-pair for a single connection request upon the 100-node network.

It is clear from Fig. 1.14 that ITSA yields the best performance while NS yields the worst one under the provisioned traffic load and the given network topologies. It is worth noting that
although ILP can provide the optimum allocation of working and shared protection paths for each connection request, the overall performance is outperformed by ITSA, MLR, and APF-PBC when the traffic load is high. The reason for this unexpected and non-intuitive result is probably as follows: The protection paths in ILP can be very long, since the calculation of sharable spare capacity is not adapted with the cost of the working paths. It is extremely hard to find a suitable scale between the cost of working and protection paths. As mentioned earlier in the paper, a long protection path consuming a large amount of sharable spare capacity may increase the potential non-sharable spare capacity for the subsequent connection requests. As a result, the overall performance can be significantly impaired. This effect deteriorates the performance especially when the traffic load is high, in which case the algorithm tries its best to find links with sufficient sharable spare capacity in the network to yield a very long protection path. However, if an attempt is made to impose an adaptive link cost on the protection path when it takes sharable spare capacity, the formulation becomes non-linear and can hardly be solved by any Linear Programming solver.

From the experimental results the ITSA scheme can find a solution very close to the optimal one with 50 iterations allowed before terminating the algorithm, which, however, differs from the ILP formulation in that it adopts the adaptive spare link-state (shown in Eq. (1.2.3)). As a result, a proper weighting on taking a spare link by a protection path is imposed, which yields the best performance among the five. It is observed from the experiment that the average length (in hops) of the path-pairs in the ITSA scheme is significantly less than is the case derived in the ILP formulation.

It is also worth noting that MLR slightly outperforms APF-PBC because the former provides a specific path-based approach during the selection of a working path such that the corresponding protection path can find maximum number of links with enough sharable spare capacity in the network. The latter scheme, on the other hand, encourages the selection of the working path minimizing the sum of the maximum non-sharable spare capacity along each link, which is nonetheless not quite straightforward and may be far from being an effective approach to improve the resource sharing.

The average computation time spent for allocating each connection request on each of the experiments is shown in Fig. 10. We do not show the case of ILP because it yields too long a computation time to have any meaning by including it into the comparison. From the results, the NS consumes the least amount of time since no resource sharing needs to be investigated. Both MLR and APF-PBC only invoke Dijkstra's algorithm twice, which are scalable to the network size. MLR needs an extra amount of time to inspect the number of Easy Links in the network during Dijkstra's relaxation process. The construction of spare link-state takes a worst-case complexity of $O(|A| \cdot |V|)$, where the factor $|V|$ comes from the length of the working path, while $|E|$ is the complexity taken to mark each arc in the graph with non-sharable spare capacity for every arc traversed by the working path. However, the practical implementation can be much faster, because neither a working path can be as long as $|V|$ hops, nor can the number of links that need to be re-marked with a new sharable spare capacity in each iteration of relaxation be as large as $|A|$. The NS case is even faster with a complexity $O(|A| \cdot \log(1+|A|/|V|) \cdot |V|)$ in implementing Suurballe's algorithm [25]. The complexity in performing ITSA is much larger than the other heuristics due to its nature of iterative trial-and-error. The number of iterations
Figure 1.14: Simulation results on the four network topologies with different traffic load.
required to find and verify the optimality in each case is strongly determined by the network size and the length of the working path. It is observed to be not practical to pursue the optimal solution in large-sized networks since the number of iterations required to verify the optimality can be too time-consuming for an on-line algorithm.

![Average Computation Time for Each Case](chart.png)

Figure 1.15: Comparison of computation time between each case.

1.5.4 Conclusion

In this chapter we studied the diverse routing problem for shared protection dealing with dynamic connection requests arriving at networks one after the other without any prior knowledge of future arrivals. Several algorithms are presented and verified with extensive simulation. We first define Two-Step-Approach in shared protection, which can explore the maximum extent of resource sharing for a protection path if the working path is given. To jointly consider the working and protection paths, we first formulate the diverse routing problem into an Integer Linear Programming (ILP) process, in which the working and shared protection paths corresponding to a connection request are solved in a single step. Due to its time-consuming solving process and bad scalability with the network size induced by the ILP formulation, we propose a heuristic algorithms, called Maximum Likelihood Relaxation (MLR), aiming at the improvement of the computational efficiency without losing much performance. The MLR scheme is a modification of Dijkstra’s algorithm jointly taking the link cost and the number of links with sufficient sharable spare capacity into consideration. To verify the proposed approaches, simulation is performed on four networks launched with dynamic traffic following the Poisson traffic model. The simulation results show that MLR can provide an ultra-fast path selection process, which behaves as a good tradeoff between computation efficiency and performance. It is worth noting that although the ILP formulation can achieve the quickest path selection process, the overall performance in terms of blocking probability is outperformed by the other heuristic counterparts due to the non-adaptation in the cost imposed on the sharable spare capacity.
Chapter 2

Shared Segment Protection (SSP) in Mesh Communication Networks with Bandwidth Guaranteed Tunnels
2.1 Introduction

Survivability has emerged as one of the most important issues in the design of modern communications networks with bandwidth guaranteed tunnels. Survivable routing is recognized as one of the best strategies to equip the networks with service continuity by pre-planning link-disjoint or node-disjoint protection paths for working capacity. With a diversely routed working-protection path-pair, once the working path is subject to any unexpected interruption, the corresponding service can be restored by switching over the working traffic to the protection path such that the failure is circumvented. With the emergence of some commercial applications and delay-sensitive services addressing stringent requirements on data integrity and service continuity, the design of survivable routing algorithms should not only be both capacity- and computation-efficient, but also minimize the possible restoration time for a specific connection, such that the maximum benefits can be gained in the operation of carrier networks.

Segment shared protection (SSP) is one of the best approaches to meet the above design requirements, where a connection is provisioned by concatenating a series of protection domains, each of which contains a working and protection segment-pair behaving as a self-healing unit for performing local restoration when the working segment is subject to any unexpected interruption. As shown in Fig. 2.1, when the working path segment of protection domain 2 is impaired unexpectedly (e.g., either link E-F, F-G, G-H, or H-I is cut), the restoration is performed locally within protection domain 2 such that the affected flow switches over to the backup segment at node E (called switching node of the protection domain) and merges back to the original working path at node J (called merging node of the protection domain).

![Diagram of SSP](image)

Figure 2.1: An illustration of SSP.

Comparing with its counterpart — path shared protection [35, 49, 4, 50, 5, 6], segment shared protection has been reported to achieve a better throughput by maximizing the extent of spare capacity resource sharing [2, 8, 51]. It can also impose a stringent limitation on the restoration time for a specific application by constraining the length/hop-count of the working and protection segment in each protection domain. The major difficulties of implementing SSP lie in the dependency between the working and spare capacities as well as the exponentially enlarged design space with the network size in identifying a set of switching/merging node-pairs to form protection domains for a connection request. Thus, most studies focus on heuristic approaches to solve the problem. In [2], a framework known as Short Leap Shared Protection
(SLSP) is proposed, which implements SSP by pre-assigning a series of switching/merging node-pairs along a given working path. The capacity-efficiency with SLSP is expected to be further improved if the working and backup segments can be jointly determined. In [34], an algorithm is developed to find the working path first followed by its backup path segments. The study is characterized by the fact that the spare capacity sharing is not considered until the physical routes of the backup path segments are defined. In [33] and [30], the authors propose two similar dynamic algorithms to switch over working traffic for each link from its immediate upstream node and merge back to the original path at any of the downstream nodes. Both studies do not impose any limitation on the lengths of the backup paths, and may not be able to guarantee the restoration time when a failure occurs. Note that the unnecessarily lengthy backup paths have been witnessed to impair the overall performance even if they share spare capacity with the other backup paths [3, 6].

The study of [52] provides an algorithm for computing QoS paths with restoration, which is characterized by considering multiple link metrics in searching the working and protection segments. This study does not consider resource sharing and adopts exhaustive search for protection segments for the working path. The study in [8] proposes an Integer Linear Programming (ILP) formulation for SSP to determine the switching/merging node-pairs along the working path given in advance. The algorithm is characterized by the efforts of inspecting all possible number of protection domains for a connection request and all possible combinations of allocation under each given number of protection domains. Since the formulation needs to have the working path first before the protection segments can be derived, the solution may be far from the optimal if the working path is not well selected. The study in [53] takes a very similar approach to that in [8]. The algorithm finds a protection segment for each link along the working path given in advance, in which “backtrack” by d hops is allowed, where d can be an arbitrary positive integer or infinity. In the study of [54], a heuristic algorithm is proposed for segment shared protection, called Optimal Protection Domain Allocation (OPDA), where a graph transformation algorithm, called transferred graph of cycles, is devised. Since the algorithm needs to enumerate and align simple cycles in the network topology as candidate cycles for a given working path, it may impair the capacity efficiency if only a limited number of cycles can be inspected within the allowed computation time. It is notable that all of the above schemes deal with work and protection paths separately, which are also known as Active Path First (APF) [6].

To jointly determine a working and protection path-pair, some studies focus on developing a programming-based solution [35, 33, 7, 3]. One of the key issues in formulating the problem under the complete routing information scenario [48] (i.e., the algorithm is aware of all the working and protection paths along each link) is that the routing of the working path and the dependency between working and spare capacity along each link must be handled in a single step. Besides, the consideration of multiple states for the protection path to consume spare capacity along each link may easily leave the formulation nonlinear [48].

The study in [48] provides an ILP formulation in the complete routing information scenario for solving the end-to-end working and shared protection path-pair according to the current link-state. The formulation is nonetheless nonlinear and cannot be solved by most of the commercially available LP solvers. The study in [7] provides a heuristic-based ILP formulation
to deal with the problem, where two scaling parameters are introduced to avoid the nonlinearity possibly induced when multiple states of spare capacity along each link are considered. The study in [3, 35] focuses on the complete routing information case and provides an ILP formulation that can solve the problem with two states of spare capacity along each link for the protection path - either sharable or non-sharable, where the link bandwidth constraint can never be addressed. It is clear that an ILP formulation for segment shared protection considering all possible states of spare capacity consumption along each link in the complete routing information scenario has never been seen before.

This paper contributes in formulating the problem into an ILP working under the complete routing information scenario, in which the location of the switching/merging nodes and the corresponding working and protection segment-pairs for a connection request can be derived in a single step according to the current link-state. This is the first linear formulation that can handle the dependency between working and spare capacity for segment shared protection, where a novel method of arc-reversal transformation is devised to deal with the situation that working segments of two neighbour protection domains may overlap with each other by more than a single node (as shown in Fig. 2.1). To avoid the nonlinearity possibly incurred when dealing with the multiple states for a protection segment to take spare capacity, a graph transformation technique is devised to facilitate the formulation.

ILP yields the least-cost (or optimal) working and protection segment-pairs for a connection request, which gives a chance to examine the offset of optimality of reported heuristic algorithms: Cascaded Diverse Routing (CDR) [55], PROMISE [8], and OPDAW [54].

2.2 An Overview on SSP

The advantage of using SSP compared with path shared protection lie in not only the reduction of restoration time, but also the achievement of larger degree of sharing. The following is an example to show the resultant capacity saving and reduced restoration time in using SSP compared with path shared protection. In Fig. 2.2, the network contains working path $W_1$ (A-B-C-D-E-F-G) and its link-disjoint protection path along (A-H-C-J-E-K-G). Let $W_2$ (C-D-E) is being allocated with its protection path. In case path shared protection is adopted, as shown in Fig. 2.2(a), the spare capacity taken by the protection path of $W_1$ can never be shared by the protection path of $W_2$ since both $W_1$ and $W_2$ are involved in a common SRG. For SSP shown in Fig. 2.2(b), on the other hand, $W_1$ is segmented into multiple segments, each of which is assigned with a switching/merging node-pair and a protection segment. For example, for the second protection domain, each switching and merging node is C and E, respectively. Therefore, the protection path of $W_2$ can share the spare capacity taken by the protection segments in the first and the third protection domain. In this case, the number of working paths in an SRG is reduced such that the total amount of non-sharable spare capacity in the network for a specific working segment is reduced, which yields better throughput. Because the restoration can be performed locally in each protection domain, the propagation time of signalling messages can be largely reduced.

The overhead in using segment shared protection compared with path shared protection is as below. Firstly, larger computation complexity is taken to solve the problem due to the efforts
in determining the switching/merging node-pair of each protection domain. Secondly, a new suite of signalling mechanism must be defined.

The signalling effort with SSP is briefly described as follows. After a fault on the working path occurs, it is localized immediately by the downstream neighbour node, which notifies the switching node of the protection domain to activate a traffic switchover. For the example in Fig. 2.1, a fault on link C-D is localized by its immediate downstream node D. A fault on link E-F is localized by the immediate downstream node F. In the former case, node D sends notification indicator signal (NIS) to notify node A and F that a fault occurred in their protection domain. In the latter case, node F sends an NIS to node E and J for a fault notification. In the case that a failure occurs to the link or node covered by two neighbour protection domains (e.g., link E-F), the protection domain close to the source node is in charge of the failure. After receiving the NIS, the merging node (i.e., A or E) immediately sends a wake-up packet to activate the configuration of switching fabric in each node along the corresponding backup segment of the corresponding protection domain, and then the traffic can be switched over to the protection path.

With the above signalling mechanism, it is clear that every node must additionally keep track of the switching node of each working path traversing through the node in the corresponding protection domain. Fault localization is necessary such that the downstream node of a failure can activate the failure recovery process, in which a higher requirement on hardware responsiveness and control complexity is needed. The largely increased computation complexity in using SSP compared with path shared protection is also a non-trivial problem that should be solved before the scheme can be practically applied.

2.2.1 Definition of Cost Functions

Now we define the cost function of a connection, which represents the weighted capacity allocation and composed of the cost of the working path and the protection path of each segment. Most of the notations and definitions are similar to Section 1.2.7. Given a network with a set of links \( L \) and nodes \( N \). The task is to find a working path (denoted as \( W \)), which is divided into \( ( \)working segments\( ) \) (denoted as \( W_i \) and the corresponding backup segments (also called protection segments and denoted as \( P_i \) between the source \( s \) and destination node \( d \) with bandwidth \( b(W) \). For easier understanding this section defines the problem at the level of network. Note that in the graph model described in sections 1.2.4 and 1.2.5 each arc belongs
to a network element, thus using this relationship the definitions of this section can be easily transformed on graphs.

The cost function for finding working path segment in the $k^{th}$ protection domain (denoted as $W_k$) with bandwidth $b(W)$ is the same as Eq. (1.2.1). The cost function of each backup segments can be calculated in a similar way as it was described in Section 1.2.7 for shared path protection. The only difference that instead of the whole working path only the working segment should be used for he cost of the corresponding backup segments $P_k$.

Similarly to Section 1.2.7 the capacity along links $j$, $\forall j \in L$, can be categorized into the following three types: free capacity (denoted as $f_j$), working capacity (denoted as $v_j$), spare capacity (denoted as $v_j$). The spare capacity can be further categorized into the following two types: sharable spare capacity (denoted as $s_j^{W_k}$), non-sharable spare capacity (denoted as $s_j^{W_k}$).

The spare link-state for the backup segment of $W$ can be expressed as:

$$c_{j}^{P_k} = \begin{cases} 
  b(W) \cdot c_j \cdot r_j^{W_k} + \varepsilon & \text{if } h_j^{W_k} \geq b(W) \\
  b(W) \cdot c_j \cdot r_j^{W_k} + \varepsilon & \text{if } h_j^{W_k} + f_j \geq b(W) > h_j^{W_k} \\
  \infty & \text{if } h_j^{W_k} + f_j < b(W)
\end{cases}$$

(2.2.1)

where $r_j^{W_k}$ is defined as $r_j^{W_k} = 1 - \frac{h_j^{W_k}}{b(W)}$ for any link $j \in L$. See also Fig. 1.3 to understand the three situations defined in Eq. (2.2.1).

Our objective is to determine $c_{j}^{P_k}$ in Eq. (2.2.1) – the spare link-state that defines the cost of the backup segment of $W_k$ passing through link $j$, in which $h_j^{W_k}$ is the only variable that must be figured out (or equivalently, $s_j^{W_k}$ since $v_j = s_j^{W_k} + h_j^{W_k}$). Note that $h_j^{W_k}$ and $s_j^{W_k}$ are network-wide link-state specific to the presence of $W_k$.

### 2.2.2 Sharable Spare Capacity Matrix

The way of calculating sharable spare capacity matrix in case of shared path protection was described in Section 1.2.8. This section will give a brief summary on the differences of the matrix form of deriving matrix $S$ in case of shared segment protection.

We define $D^K$ as the set of already established working and protection path segments in the network. We need to define the working segment-SRG’ incidence matrices as $A^K$, which is a $|D^K| \times |SRG’|$ matrix. The $a_{ij}^{K}$ entry of $A^K$ is 1 if the $i^{th}$ working path segment of the array is involved in the $j^{th}$ SRG’ and the neither the source nor the destination node of the working path (or the connection) are involved in the SRG’. We also define the protection segment-link incidence matrices as $B^K$ with a size of $|D^K| \times |L|$ matrix. The $b_{ij}^{K}$ entry of $B^K$ is 1 if the $i^{th}$ protection path segment passes through the $j^{th}$ link of the network. We define $M^K$ as a diagonal matrix of size $|D^K| \times |D^K|$ with $m_{ii}^{K}$ standing for the bandwidth of the $i^{th}$ working segment, and $m_{ii}^{K} = 0$ if $i \neq j$.

We can formulate the spare provision matrix for $W$ as:

$$S = ((B^K)^T \cdot M^K \cdot A^K) \odot G^W$$

(2.2.2)

which is a $|L| \times |BSS|$ matrix. Matrix $G^W$ is called $BSS – SRG’$ incidence matrix and it was defined in Section 1.2.8.
Similarly to shared path protection we can define a vector $W_k$ of size $|1| \times |L|$ representing $W_k$, where $W_k = \sum_{l \in W_k} W_l^T$. We can also define $P_k$ for the protection path in the similar way. Let us define $W_k'$ of size $|1| \times |BSS|$ storing the $BSS$s of $W_k$ in a similar way.

Applying a $\max$ operation upon each column of $S$ corresponding to the $BSS$s of $W_k$ will yield a $1 \times |L|$ vector $\Sigma^{W_k}$, which keeps the amount of non-sharable spare capacity along each link provided with working segment $W_k$. It can be formulated as:

$$\Sigma^{W_k} = \Sigma \cap W_k'$$  \hspace{1cm} (2.23)

The $1 \times |L|$ vector $H^{W_k}$, which keeps the amount of sharable spare capacity along each link provided with $k^{th}$ segment of $W$, can thus be derived by referring to the relationship

$$H^{W_k} = V - \Sigma^{W_k}$$  \hspace{1cm} (2.24)

Finally we can define the sharable spare capacity matrix ($H$) size of $|L| \times |BSS|$. The $(i,j)$ entry of $H$ keeps the amount of sharable spare capacity along link $i$ provided for protection path if the corresponding working path involved in $j^{th}$ $BSS$ of $BSS$. It can be expressed as

$$h_{i,t} = v_i - s_{i,t}$$  \hspace{1cm} (2.25)

We also define a matrix called $P$. The entry $(l, j)$ (i.e., $r_j^l$) stands for the ratio of the bandwidth needs to be allocated as spare capacity along link $j$ if the working route pass through link $l$ (similarly to 1.2.2).

$$r_{j,l} = 1 - \frac{h_{j,l}}{b(W)}$$  \hspace{1cm} (2.26)

### 2.2.3 Cost Function in Matrix Form

With $\Sigma$ we can formulate the cost function in matrix form. This formulation does not use $\infty$ as a cost, thus it is only valid for feasible solutions. See Section 2.3 for feasibility conditions.

To formulate the cost of $W$ we need to sum up the cost along each link taken by $W$, which can be done like in Eq. (1.2.22):

$$c(W) = b(W) \cdot ||C^T \cdot W||$$  \hspace{1cm} (2.27)

Let us define a matrix $W^{K'}$ representing all segments of $W$, where $W^{K'}$ is a diagonal matrix of size $k_W \cdot |BSS| \times |BSS|$, $k_W$ (denotes the number of segments of $W$) such that $\text{diag}(W^{K'}) = \{(W^1)^T, \ldots, (W^{k_W})^T\}$, so the diagonal of $W^{K'}$ are the $W_k$ of each segment. Let us define a matrix $P_k$ for $P$ such that $\text{diag}(P_k) = P^k$ with size of $|L| \times |L|$. A matrix $P_k$ is also defined for all $P^k$ such that $(P^{K_k})^T = \{P^1, \ldots, P^{k_W}\}$ of size $|L| \times k_W \cdot |L|$.

We will use the Kronecker multiplication of matrices, which is the following [45]:

$$\mathbf{A} \otimes \mathbf{B} = [A \cdot b_{i,j}] = 
\begin{bmatrix}
A \cdot b_{1,1} & A \cdot b_{1,2} & \ldots & A \cdot b_{1,n} \\
A \cdot b_{2,1} & A \cdot b_{2,2} & \ldots & A \cdot b_{2,n} \\
\cdots & \cdots & \cdots & \cdots \\
A \cdot b_{m,1} & A \cdot b_{m,2} & \ldots & A \cdot b_{m,n}
\end{bmatrix}$$  \hspace{1cm} (2.28)

With Kronecker multiplication we can define the cost of $P$ as:
\begin{equation}
    c(P) = b(P) \cdot \left\| C^T \cdot (P^K)^T \cdot (E \otimes R) \cdot (W^K)' \right\| \tag{2.2.9}
\end{equation}

Figure 2.3: An illustration for the derivation of the cost of protection segments.

where $E_{kw}$ is a $kW \times kW$ unit matrix. It holds since $(P^K)^T \cdot (E \otimes R) \cdot W^K'$ is a $|L| \times kW \cdot |BSS|$ size matrix (see also Fig. 2.3 for illustration) made of $kW$ matrices of size $|L| \times |BSS|$. The $k$th row's $(i, j)$ entry is $\sum_{o=1}^{L} \sum_{l=1}^{BSS} (p^K)_{i,o} \cdot r_{o,l} \cdot (w^K)_j^l$, where $w^K$ is an entry of the diagonal matrix $W^k$, which can be expressed $diag(W^k) = W^k$. It is multiplied from left with a diagonal matrix $C$, which means each row of the matrix is multiplied with $c_{i,i}$, thus the $(i, j)$ entry is $\sum_{o=1}^{L} \sum_{l=1}^{BSS} c_{i,i} \cdot (p^K)_{i,o} \cdot r_{o,l} \cdot (w^K)_j^l$. It is a matrix with elements of $c_{i,i} \cdot r_{k,l}$ at the rows representing links involved in $P$ and columns representing $BSS$s involved in $W$. The norm of this matrix equals with

\begin{equation}
    \left\| C^T \cdot (P^K)^T \cdot (E \otimes R) \cdot W^K' \right\| = \tag{2.2.10}
\end{equation}

\begin{align*}
    &\sum_{i=1}^{L} \max_j \max_k \sum_{o=1}^{L} \sum_{l=1}^{BSS} c_{i,i} \cdot p^K_{i,o} \cdot r_{o,l} \cdot w^K_{l,j} = \\
    &\sum_{i=1}^{L} \max_j \max_k \sum_{l=1}^{BSS} c_{i,i} \cdot p^K_{i,i} \cdot r_{i,l} \cdot w^K_{l,j} = \\
    &\sum_{i=1}^{L} \max_k \sum_{j=1}^{W} c_{i,i} \cdot p^K_{i,i} \cdot r_{i,j} \cdot w^K_{j,j} = \\
    &\sum_{i \in P} \max_j \max_k c_{i,i} \cdot r_{i,j} = \\
    &\sum_{i \in P} \max_k \max_{j \in W} c_{i,i} \cdot r_{i,j} = \sum_{i \in P} c_{i,i} \cdot \max_{k,l} W^k
\end{align*}

where the second equality is true, due to the fact that $P$ is diagonal (i.e., $p_{i,k} = 0$ if $i \neq k$) and the third equality is true since $W$ is diagonal (i.e., $w_{i,j} = 0$ if $l \neq j$). Finally the sixth equality is due to Eq. (1.2.8). After multiplying with $b(W)$ we get the same cost function defined in Eq. (2.2.1).

Summarizing this section the total cost can be expressed as:
$$c_{total} = c(W) + c(P) = b(W) \cdot (\| \textbf{C} \cdot \textbf{W} \| + \| \textbf{C}^T \cdot (\textbf{P}^K)^T \cdot (\textbf{E} \otimes \textbf{R}) \cdot \textbf{W}^K \| )$$  \hspace{1cm} (2.2.11)

### 2.3 Problem Definition

Finally we define the problem of Shared Segment Protection. Given a network with some already established connections and with a new demand. With load balancing functions (see Section 1.2.6) a cost function is calculated for each link. A directed graph is built up representing the network as it was described in Section 1.2.4 or 1.2.5. If the SRGs are defined such that node-failures are considered, the protected nodes, should be split into two twin vertices. One of the vertex is called incoming vertex and the other is called outgoing vertex [56]. The incoming arcs of the node are connected to the incoming vertex, all the rest are connected to the outgoing vertex and an additional arc is inserted in the graph form the incoming vertex to the outgoing vertex. If the source node of the demand is split in the graph s must be assigned to the outgoing vertex, and in case of the destination node is split d must be assigned to the incoming vertex. Since each arc of the graph belongs to a network element the SRGs and all notations defined in sections 1.2.7, 1.2.8, 1.2.9 can be defined on the arcs of a graph as well.  

All arcs of the graph are erased where the free capacity plus the spare capacity is less than the bandwidth of the new demand (all arcs $j$ is erased if $f_j + v_j < b(W)$). The $\textbf{S}$ matrix is calculated as it was described in Section 1.2.8. The routing problem can be defined as:

**Given:**

- a directed graph $G(V, A)$, with $V$ and $A$ being the set of vertices and arcs, respectively,
- the free capacity ($f$) and the spare capacity ($v$) of each arc,
- the SRGs of $G$ (sets on the arcs of $G$),
- the $\textbf{S}$ matrix (it can be assigned to arcs as vectors of size $|\textbf{BS}|$)
- the source node $s$ and the destination $d$ and the bandwidth $b(W)$ of the new demand,

**Find:**

- working path $W$,
- the switching and merging nodes of each segment along $W$ (represented in matrix $\textbf{W}_0$),
- protection path $P$ of each segment (represented in matrix $\textbf{P}$),
- $W_k$ and $P_k$ should be SRG disjoint for $\forall k$,
- the feasible condition of the working path is $f_i \geq b(W)$ for $\forall i \in W$
- the feasible condition of the protection path is $f_i + v_i \geq b(W) + \max_{v_j \in W^k} s_{i,j}$ for $\forall i \in P_k$ and $\forall k$,
- such that

$$\text{Minimize: } b(W) \cdot (\| \textbf{C} \cdot \textbf{W} \| + \| \textbf{C}^T \cdot (\textbf{P}^K)^T \cdot (\textbf{E} \otimes \textbf{R}) \cdot \textbf{W}^K \| )$$
2.4 **Claim 2.1: ILP Formulation of Shared Segment Protection**

This section introduces a linear formulation for the segment shared protection problem. Our approach is to find a path \( Q \), called **mass protection path**, which is composed of all the backup segments and some links along the working path. A simple example is shown in Fig. 2.4, where \( Q \) is \((s\rightarrow a\rightarrow b\rightarrow c\rightarrow d)\). The first protection domain is formed by the working and protection segments \((s\rightarrow k\rightarrow b)\) and \((s\rightarrow a\rightarrow b)\), respectively; while the second is formed by \((c\rightarrow l\rightarrow d)\) and \((c\rightarrow n\rightarrow d)\), respectively. The allowance of overlapping between the working segments of two neighbour protection domains is to explore the largest design space so as to guarantee the optimality of the derived solution. Note that \( Q \) may contain loops to reflect the fact that spare capacity sharing can happen between two protection segments of different protection domains.

![Diagram of mass protection path Q](image)

**Figure 2.4: Design of mass protection path \( Q \).**

Three residual graphs are defined to facilitate the solving of this problem, each of which carries one or a few variables for the identification of the working and protection segment-pairs. The graph for solving the working segments is denoted as \( G_w(V, A_w) \) and is composed of arcs with \( f_j \geq b(W) \) for \( j \in A_w \) \((x, \bar{x} \text{ and } x^k \text{ variables in the following formulas are assigned to this graph})\). The second residual graph is denoted as \( G_p(V, A_p) \), which is to facilitate solving the protection segments. We need this graph to record the spare link-state because working and protection paths take different suites of link-state with shared protection. This graph is composed of the arcs where the amount of free capacity \( f_j \) plus that of the spare capacity \( v_j \) is larger than or equal to \( b(W) \) \((i.e., b(W) \leq f_j + v_j \text{ for } j \in A_p)\). The variables defined in this graph are \( y^k \) and \( y \), which will be discussed in details later.

The third residual graph \( G'_p(N, A'_p) \) is composed of all the arcs in \( A_p \) along with the arcs of \( A_w \) in a reverse direction. The inclusion of the arcs of \( A_w \) in a reverse direction into \( A'_p \) is called **arc-reversal transformation** similar to the graph technique adopted in Snurbelle’s algorithm [25]. With \( G'_p(N, A'_p) \), we will be able to handle the reverse arcs caused by the overlapping between \( Q \) and \( W \), such that the route of \( Q \) can be identified. The variables defined in this graph are \( y' \) and \( y' \), which will be discussed in details later. It is clear that we have the following relationship \( A'_p = A_p \cup \text{reversed}(A_w) \). Therefore, \( A'_p \) contains **forward arcs** denoted as \((\bar{a}, \bar{b})\), which are due to the arcs of \( A_p \), as well as the **reversed arcs** denoted as \((\bar{a}, \bar{b})\), which are due to the reversal of arc \((a, b)\) in \( A_w \). The transformation yields a fact that if there is a bi-directional link between nodes \( a \) and \( b \), and node \( a \) is split into two vertices \( a_i \) as the incoming vertex and \( a_o \) as the outgoing vertex, and node \( b \) is split into \( b_i \) and \( b_o \) (with the following arcs inserted...
\(\overline{a_i}, a_o\) as \((a_i \rightarrow a_o)\); \(\overline{a_i}, a_o\) as \((a_o \rightarrow a_i)\); \(\overline{b_i}, b_o\) as \((b_i \rightarrow b_o)\); \(\overline{b_i}, b_o\) as \((b_o \rightarrow b_i)\)), and the amount of free capacity larger than or equal to \(b(W)\), then \(G'_p\) will have four arcs between nodes \(a\) and \(b\) (i.e., \(\overline{a}, \overline{b}\) as \((a_o \rightarrow b_i)\); \(\overline{a}, b\) as \((b_i \rightarrow a_o)\); \(b, \overline{a}\) as \((b_o \rightarrow a_i)\); \(\overline{b}, a\) as \((a_i \rightarrow b_o)\)). If the amount of free capacity is less than or equal to \(b(W)\), and the amount of free plus spare capacity is larger than or equal to \(b(W)\) (i.e., \(f_j < b(W) \leq f_j + v_j\)), then \(G'_p\) will have two arcs between \(a\) and \(b\) (i.e., \((\overline{a}, \overline{b})\) as \((a_o \rightarrow b_i)\); \((\overline{b}, a)\) as \((b_i \rightarrow a_o)\)); otherwise, there is no arc between node \(a\) and \(b\) in \(G'_p\).

Please refer to Fig. 2.5 for an illustration of the graph transformation.

In the formulation, all the three graphs \((G_w, G_p, G'_p)\) are considered and indexed in an array during the implementation, and we have to keep track of the index of each arc on different graphs even though they are in the same direction and at the same physical location. For example, we must distinguish between \((\overline{a}, b) \in A'_p\) and \(a, b \in A_w\), and between \((\overline{a}, b) \in A'_p\) and \(a, b \in A_p\), and between \(a, b \in A_p\) and \(a, b \in A_w\), in the array keeping the indexes of the arcs.

![Graph transformations for the arcs in G, which yields the graphs G_w, G_p and G'_p, respectively.](image)

To introduce the target function of the proposed ILP formulation, the following two flow indicators are defined in the graphs \(G_w\) and \(G'_p\): \(x_{a,b}\) is a binary variable with size \(|A_w|\) defined in the graph \(G_w\), while \(y'_{u,v}\) is an integer variable ranged \([0,k_{max}]\) with size \(|A'_p|\) defined in the graph \(G'_p\), where \(k_{max}\) is the maximum number of protection domains that can be possibly handled in the problem. The target function is as follows:

\[
\text{Minimize: } \sum_{(a,b)\in A_w} b(W) \cdot x_{a,b} + \sum_{(u,v)\in A_p} (b(W) \cdot c_{u,v} \cdot z_{u,v} + \varepsilon) \cdot y'_{u,v} \tag{2.4.1}
\]

where \(c_{a,b}\) is the cost per unit of working bandwidth to reserve arc \((a, b)\), which was defined in Section 1.2.6. \(y'_{u,v}\) is a mirror binary variable of \(y'_{u,v}\) to facilitate the calculation of the total cost caused by the protection segments, and will be further discussed later.

Each of \(x_{a,b}\) and \(y'_{u,v}\) indicates the number of times the working and mass protection paths traverse \((a, b) \in A_w\) and \((u, v) \in A'_p\), respectively. The reason of setting \(y'_{u,v}\) an integer instead of a binary variable is that path \(Q\), which is composed of the protection segments of all the protection domains, may have loops in case the protection segments of two different protection domains traverse through the same link. Therefore, \(y'_{u,v}\) may be larger than 1. Therefore, the concatenation of all arcs with \(x_{a,b} = 1\) yields \(W\), while the concatenation of all the arcs with \(y'_{u,v} \geq 1\) yields \(Q\). It is clear that the overlapped links between \(W\) and \(Q\) should not be
considered when calculating the cost for the corresponding protection segments in the target function. Besides, each link should be counted only once even if $Q$ traverses any link by multiple times due to the possible spare capacity sharing between protection segments of any two protection domains for the connection. Therefore, a transformation is required from the integer variable $y'_{u,v}$ (defined in $G'_p$) into a new binary variable, denoted as $y_{u,v}$ (defined in $G_p$). This transformation can be simply done by filtering out those links taken by $Q$ which are defined in $G'_p$ but not in $G_p$. This filtering can be done with the following linear formula:

$$k_{\text{max}} \cdot y_{u,v} - y'_{u,v} \geq 0 \text{ for } \forall (u, v) \in A_p, \forall (u, v) \in A'_p$$ (2.4.2)

In the transformation, $y_{u,v}$ is zero if $y'_{u,v}$ is zero, and is 1 if $y'_{u,v} > 0$. As for $z_{u,v}$, it is for the protection segment taking per-unit of spare capacity along link $(u, v)$.

The target function is subject to the following constraints:

$$\sum_{\forall (a,b) \in A_w} x_{a,b} - \sum_{\forall (b,a) \in A_w} x_{b,a} = \begin{cases} 1 & \text{if } a \text{ is the source node } s \\ -1 & \text{if } a \text{ is the destination node } d \\ 0 & \text{otherwise} \end{cases}$$ (2.4.3)

$$\sum_{\forall (a,b) \in A'_p} y'_{a,b} - \sum_{\forall (b,a) \in A'_p} y'_{b,a} = \begin{cases} 1 & \text{if } a \text{ is the source node } s \\ -1 & \text{if } a \text{ is the destination node } d \\ 0 & \text{otherwise} \end{cases}$$ (2.4.3)

These equations are the flow conservation constraints for the working and mass protection paths, respectively.

It is important to note that $x_{a,b}$ and $y'_{a,b}$ will be exclusive in terms of the SRGs they take. However, an arc can be taken by $y'_{a,b}$ in a reversed direction only if $x_{a,b}$ pass through it. Besides, each reversed arc can be used only once since the algorithm only allows two working segments overlapped. The above statements can be formulated into the following two constraints:

$$n \cdot k_{\text{max}} \cdot x_a + \sum_{\forall b \text{ in the same } SRG \in Z^W \text{ as } a} y'_{a,b} \leq n \cdot k_{\text{max}} \text{ for } \forall a \in A_w$$

$$x_{a,b} \geq y'_{a,b} \text{ for } \forall (a, b) \in A_w, \forall (a, b) \in A'_p$$ (2.4.4)

where $y'_{a,b}$ represents the reversed link of $A'_p$. In the summation all SRGs are considered except the $SRG_{aut}$, which failure separates $s$ and $d$, since the failure of those SRGs cannot be protected. In case of node protection $Z^W$ is an empty set and node splitting is done on the graph as it was described in Section 2.3. The value of the constant $n$ is greater than the maximum number of arc in the SRGs (e.g. it can be set to $|V|$). Under the above two constraints, $Q$ has to be disjoint from $W$ except for those arcs of $W$ being reversed (see Fig. 2.4). These two constraints not only assert the SRG disjointness of the working and the corresponding protection segment, but also facilitate the indication of the switching/merging nodes for each protection domain along $W$.

A pair of variables, $x_{a,b}$ (with size $|A_w|$) and $y'_{a,b}$ (with size $|A'_p|$), is assigned to each link along $W$ and $Q$, respectively, such that the first link from the source has a label 1; and if a protection domain ends or starts at a node, the labels of the following arcs will be increased by 1. This labelling method is similar to that proposed in [8]. Let $k_{\text{max}}$ be the number of protection domains with an upper bound $|V| - 1$. In solving the ILP problem, the value of
$k_{\text{max}}$ should be set to the upper bound: $|V| - 1$ to guarantee the derivation of the optimal solution. If the value of $k_{\text{max}}$ is set smaller than $k_W$ (i.e., the number of protection domain in the optimal solution), the LP solver cannot return an optimal solution although we have a smaller problem size.

For $\hat{x}_{a,b}$, we have the following constraints:

$$(2k_{\text{max}} - 1) \cdot x_{a,b} \geq \hat{x}_{a,b} \geq 0 \quad \text{for } (a, b) \in A_w$$  \hspace{1cm} (2.45)

$$\sum_{(a,b) \in A_w, a \neq d} \hat{x}_{a,b} - \sum_{(b,a) \in A_w, a \neq s} \hat{x}_{b,a} \geq -2k_{\text{max}} + \sum_{(a,b) \in A_p, a \neq d} y'_{a,b} + \sum_{(b,a) \in A_p, a \neq s} y'_{b,a} \quad \forall a \in V$$  \hspace{1cm} (2.46)

$$\sum_{(a,b) \in A_w, a \neq d} \hat{x}_{a,b} - \sum_{(b,a) \in A_w, a \neq s} \hat{x}_{b,a} \leq \sum_{(a,b) \in A_p, a \neq d} y'_{a,b} + \sum_{(b,a) \in A_p, a \neq s} y'_{b,a} \quad \forall a \in V$$  \hspace{1cm} (2.47)

$$\sum_{(a,b) \in A_w} \hat{x}_{a,b} = 1 \quad \forall b \in V$$  \hspace{1cm} (2.48)

The constraint in Eq. (2.45) ensures that $\hat{x}_{a,b}$ is upper-bounded by $(2k_{\text{max}} - 1)$, and is nonzero only if $W$ passes through $(a, b)$. To verify Eq. (2.46) and Eq. (2.47), the following four situations are defined for a vertex (different from the source or destination) taken by $W$: (a) $Q$ merges back to $W$; (b) $Q$ switches out of $W$; (c) $Q$ merges back and switches out of $W$ (if the failure of the node is not protected); (d) otherwise. Eq. (2.46) and (2.47) behave as a special type of flow conservation constraint upon the net change of $\hat{x}_{a,b}$ for vertex $a$ in the network, which is denoted at the left-hand side of the equations. In Eq. (2.46), the value of $\hat{x}_{a,b}$ of vertex $a$ along $W$ increases by 1 in the case of situations (a) and (b), increases by 2 in case of situation (c), and is unchanged otherwise. The constraint on the net change in terms of $\hat{x}_{a,b}$ has a lower bound specified at the right-hand side of the equation, where the term $\sum_{(a,b) \in A_w, a \neq d} k_{\text{max}} \cdot x_{a,b} + \sum_{(b,a) \in A_w, a \neq s} k_{\text{max}} \cdot x_{b,a} - 2 \cdot k_{\text{max}}$ checks if vertex $a$ is taken by $W$. It is clear that the term is 0 if vertex $a$ is traversed by $W$, and is $-2k_{\text{max}}$ otherwise. Therefore, if vertex $a$ is not taken by $W$, Eq. (2.46) automatically holds.

The four cases specified in Eq. (2.46) are presented as follows. In case of (a), the increase of $\hat{x}_{a,b}$ is 1 since $\sum_{(a,b) \in A_p, a \neq d} y'_{a,b} = 0$ and $\sum_{(b,a) \in A_p, a \neq s} y'_{b,a} = 1$ (since $Q$ merges back to $W$ at vertex $a$). In case (b), the increase of $\hat{x}_{a,b}$ is still 1 because $\sum_{(b,a) \in A_p, a \neq s} y'_{b,a} = 0$ and $\sum_{(a,b) \in A_p, a \neq d} y'_{a,b} = 1$. In situation (c), increase of $\hat{x}_{a,b}$ is 2 since $\sum_{(a,b) \in A_p, a \neq d} y'_{a,b} = \sum_{(b,a) \in A_p, a \neq s} y'_{b,a} = 1$. If vertex $a$ does not correspond to a switching or a merging node (but it is taken by $W$), the right-hand-side of Eq. (2.46) becomes 0, in which no change upon $\hat{x}_{a,b}$ is required.

Eq. (2.47) basically has the same working principles as Eq. (2.46) except that when vertex $a$ is not taken by $W$, the right-hand side becomes 0 instead of $-2k_{\text{max}}$. Both Eq. (2.46) and (2.47) constrain the net change of the value of $\hat{x}_{a,b}$ to be either 0, 1, or 2, for any vertex $a$ taken by $W$, depending on how path $Q$ switches out and merges back to $W$. Eq. (2.48) sets $\hat{x}_{a,b}$ to 1 if node $a$ corresponds to the source node.

For $y'_{a,b}$, we have the following constraints:
\[ k_{\text{max}} \cdot (2k_{\text{max}} - 1) \cdot y'_{a,b} \geq y'_{a,b} \geq 0 \quad \text{for } \forall (a, b) \in A'_p \]  

(2.4.9)

\[ \sum_{(a,b) \in A'_p, a \neq d} y'_{a,b} - \sum_{(b,a) \in A'_p, a \neq s} y'_{b,a} \geq 0 \]  

(2.4.10)

\[ \sum_{(a,b) \in A'_p, a \neq d} (x_{a,b} - y'_{a,b}) + \sum_{(b,a) \in A'_p, a \neq s} (x_{b,a} - y'_{b,a}) + \]  

\[ k_{\text{max}} \cdot y'_{a,b} + \sum_{(b,a) \in A'_p, a \neq s} k_{\text{max}} \cdot y'_{b,a} - 2k_{\text{max}} + \]  

\[ \sum_{(a,b) \in A'_p, a \neq d} k_{\text{max}} x_{a,b} + \sum_{(b,a) \in A'_p, a \neq s} k_{\text{max}} x_{b,a} - 2k_{\text{max}}^2 \]  

(2.4.11)

\[ \sum_{(a,b) \in A'_p, a \neq d} (x_{a,b} - y'_{a,b}) + \sum_{(b,a) \in A'_p, a \neq s} (x_{b,a} - y'_{b,a}) \quad \forall a \in V \]  

(2.4.12)

\[ \sum_{(a,b) \in A'_p} y'_{a,b} = 1 \quad \forall b \in V \]  

(2.4.13)

The constraint in Eq. (2.4.9) ensures that \( y'_{a,b} \) is nonzero only if \( Q \) passes through arc \((a, b)\). The constraint in Eq. (2.4.11) and (2.4.12) ensures that the value of \( y'_{a,b} \) on path \( Q \) increases by 1 only if \( Q \) merges back to \( W \) or \( Q \) switches out of \( W \) at vertex \( a \). The idea behind Eq. (2.4.11) and Eq. (2.4.12) is similar to that of Eq. (2.4.6) and Eq. (2.4.7). The only difference is that instead of the forward arcs of \( Q \) (denoted as \( y'_{a,b} \)), the term \((x_{a,b} - y'_{a,b})\) is used, which is nonzero for link \((a, b)\) along \( W \) not taken by \( Q \). The constraint in Eq. (2.4.13) is to set \( y'_{a,b} \) as 1 if vertex \( a \) corresponds to the source node, and that there would be only a single protection link stretching out of the source node. Please refer to Fig. 2.6 for an illustration for the variables formulated above. It can be easily observed that the maximum of \( \hat{x}_{a,b} \) is \( 2k_{\text{max}} - 1 \); and the maximum of \( y'_{a,b} \) is less than \((2k_{\text{max}} - 1) \cdot y'_{a,b}\) even if path \( Q \) have loops.

With \( \hat{x} \) and \( \hat{y}' \) link labels, path \( W \) is divided into segments such that each link along it is covered by at least one protection segment. This effort introduces \( k_{\text{max}} \cdot |A_w| \) and \( k_{\text{max}} \cdot |A_p| \) arc-domain incidence binary variables denoted as \( x'_{a,b} \) and \( y'_{a,b} \), which is 1 if link \((a, b)\) is traversed by the working and protection segment of the \( k^{th} \) protection domain, respectively. Note that the only variable defined in graph \( E_p \) is the variable \( y'_{a,b} \) in the formulation. We can alternatively define the variable \( y'_{a,b} \) upon \( A'_p \) instead of having a new graph \( A'_p \), in which the formulation turns out to take only two residual graphs. Although it is a way more tractable to implement, there would be at most \( k_{\text{max}} \cdot |A_w| \) variables unnecessarily induced due to the fact that we do not need to define \( y'_{a,b} \) on the reversed links of \( A'_p \).

For \( y^k_{a,b} \), the following constraints are introduced:

\[ \sum_{k=1}^{k_{\text{max}}} (2k - 1) \cdot y^k_{a,b} = y'_{a,b} \quad \forall (a, b) \in A_p \]  

(2.4.14)

\[ \sum_{k=1}^{k_{\text{max}}} y^k_{a,b} = y'_{a,b} \quad \forall (a, b) \in A_p \]  

(2.4.15)

\[ 0 \leq y^k_{a,b} \quad \text{for } 1 \leq k \leq k_{\text{max}} \text{ and } \forall (a, b) \in A_p \]  

(2.4.16)
Figure 2.6: An example showing the variables \( x_{a,b}, y'_{a,b}, \tilde{x}_{a,b}, \text{ and } y'_{a,b} \). It is zero for the variables of the arcs not shown on the figure.

\[- \sum_{(a,b) \in E_w} (\tilde{x}_{a,b} + x_{a,b}) \leq \sum_{(a,b) \in A_p} (2\kappa_{\text{max}} - 1) \cdot y_{a,b}^k - \sum_{(b,a) \in A_p} (2\kappa_{\text{max}} - 1) \cdot y_{b,a}^k \quad \forall a \in V \]  

(24.17)

\[ \sum_{(a,b) \in A_p} (2\kappa_{\text{max}} - 1) \cdot y_{a,b}^k - \sum_{(b,a) \in A_p} (2\kappa_{\text{max}} - 1) \cdot y_{b,a}^k \leq \sum_{(b,a) \in A_w} (\tilde{x}_{b,a} + x_{b,a}) \quad \forall a \in V \]  

(24.18)

Eq. (24.14) can be easily verified by observing Fig. 2.6, where the value of \( y'_{a,b} \) on \( Q \) of the first protection domain is 1; and in the second protection domain \( y'_{a,b} \) is 3; and in the \( k \)th protection domain \( y'_{a,b} \) is \( 2k - 1 \). Eq. (24.15) ensures that the number of traversals of path \( Q \) upon each link is correctly counted. It is clear that Eq. (24.14), Eq. (24.15) and Eq. (24.16) set \( y_{a,b}^k = 1 \) only when \( y'_{a,b} = 2k - 1 \). Eq. (24.17) and Eq. (24.18) are flow conservation constraints for \( y_{a,b}^k \) since \( \sum_{(b,a) \in A_w} \tilde{x}_{b,a} \) is 0 for all nodes except for the ones along \( W \). This ensures that \( y_{a,b}^k \) to be the flows starting from a node along \( W \) with label \( \tilde{x} = 2k - 1 - 1 \) (an incoming arc has the label) and terminate at a node along \( W \) with label \( \tilde{x} = 2k - 1 - 1 \) (an outgoing arc has the label).

Reducing the number of integer variable significantly reduce the runtime of the solver [57, 58]. Note that variable \( y_{a,b}^k \) for \( k \) equals to 1 or 2 can be relax to real. To verify this state we need to observe the case when \( y_{a,b}^k \) is 0, 1, 3, or more. If \( y_{a,b}^k \) is 0 then \( y_{a,b}^k = 0 \) for 1 \( \leq \) \( k \leq \kappa_{\text{max}} \) since \( y_{a,b}^k \geq 0 \) and \( \sum_{k=1}^{\kappa_{\text{max}}} y_{a,b}^k = 0 \). If \( y_{a,b}^k \) is 1 Eq. (24.14) and Eq. (24.15) can be reduced to \( \sum_{k=1}^{\kappa_{\text{max}}} (2k - 2) y_{a,b}^k = 0 \) , which means \( y_{a,b}^k = 0 \) for 2 \( \leq \) \( k \leq \kappa_{\text{max}} \), thus \( y_{a,b}^1 = 1 \) due to Eq. (24.15). In case of \( y_{a,b}^k = 3 \) variable \( y_{a,b}^k \) must be 0 for \( k = 3 \) \( \kappa_{\text{max}} \) since they are binary variables and Eq. (24.14) must hold. Thus Eq. (24.14) and Eq. (24.15) can be reduced to \( y_{a,b}^1 + y_{a,b}^2 = 1 \) and \( y_{a,b}^1 + y_{a,b}^2 = 3 \), respectively, which holds only if \( y_{a,b}^1 = 0 \) and \( y_{a,b}^2 = 1 \). If \( y_{a,b}^1 \) is more than 3 one of the \( y_{a,b}^k \) variables for 3 \( \leq \) \( k \leq \kappa_{\text{max}} \) must be positive to satisfy Eq. (24.15) and since they are binary variable it must be 1, and if any of \( y_{a,b}^k = 1 \) all the other \( y_{a,b}^l = 0 \left( \forall l \neq k \right) \) due to Eq. (24.15). For \( x_{a,b}^k \) the following constraints are introduced:

\[ \sum_{k=1}^{\kappa_{\text{max}}} (2k - 1) \cdot x_{a,b}^k = \tilde{x}_{a,b} + y'_{a,b} \quad \forall (a,b) \in A_w \]  

(24.19)
\[
\sum_{k=1}^{k_{\text{max}}} x_{a,b}^k = x_{a,b} + y_{a,b}^r \quad \forall (a, b) \in A_w
\]  

\[
0 \leq x_{a,b}^k \leq 1 \quad \text{for } 1 \leq k \leq k_{\text{max}} \text{ and } \forall (a, b) \in A_w
\]  

Eq. (2.4.19) corresponds to Eq. (20) in [55]; however, Eq. (2.4.19) is much more tight than Eq. (20), and the runtime of the CPLEX solver can be significantly reduced since the gap between the relaxed problem and the integer solution is also reduced [57, 58]. Eq. (2.4.19) can be easily verified by the following argument. On the links of the \( k \)th protection domain, we have \( \hat{x}_{a,b} = 2k - 2 \) on the non-overlapped links and \( \hat{x}_{a,b} = 2k - 1 \) on the overlapped links of the \((k - 1)\)th and the \( k \)th protection domain; we have \( \hat{x}_{a,b} = 2k \) on the overlapped links of \( k \)th and \((k - 1)\)th protection domain. On the non-overlapped link(s) of the \( k \)th protection domain, \( y_{a,b}^r = 0 \) and Eq. (2.4.19) holds. For a link serving as the overlapped link of the \( k \)th and \((k + 1)\)th working segments, the left hand side of Eq. (2.4.19) is equal to \((2k - 1)\cdot x_{a,b}^k + (k+1)\cdot x_{a,b}^{k+1} = 4k \). Since \( y_{a,b}^r = 2k \), the equation naturally holds. It verified that Eq. (2.4.19) holds in all cases.

Since the overlapped link(s) of two neighbor working segments is counted twice in \( x_{a,b}^k \), while the corresponding reversed arcs of \( G'_p \) are taken by \( Q \). This fact is formulated as Eq. (2.4.20). Note that variables \( x_{a,b}^k \) for \( k = 1 \) or 2 can be relaxed to real due to the same reason as that of \( y_{a,b}^r \).

The constraint upon the variable \( z_{a,v} \) defined in the target function is as follows:

\[
x_a^k + y_e^k - 1 - \frac{h_{e,b}}{b(W)} \leq z_e \quad 1 \leq k \leq k_{\text{max}}, \forall e \in A_p, \forall a \in A_w, a \in b \in \text{BSS}, h_{e,b} + f_e \geq b(W)
\]  

\[
z_e \geq 0 \quad \forall e \in A_p
\]  

Here the SRG constraint is considered using a pre-defined \(|A_w| \times |\text{BSS}| \) matrix in Section 2.2.2 recording \( h_{e,b} \), where \( e \in A_p \) and \( b \in \text{BSS} \), which can be prepared off-line and behaves as an upper bound of spare capacity along arc \( e \) sharable by the backup segment if the corresponding working segment passes through arc \( a \) involved in \( b \)th \( \text{BSS} \) in \( \text{BSS} \). Eq. (2.4.22) ensures that when link \( a \) and \( e \) are taken by the working and protection segments in the \( k \)th protection domain, respectively, the resultant amount of scaling (i.e., \( z_e \)) is at least \( 1 - h_{e,b}/b(W) \) (since \( x_a^k + y_e^k = 2 \)). If \( h_{e,b} \geq b(W) \), it means that there is sufficient sharable spare capacity along link \( e \) that can be taken to protect any additional \( b(W) \) units of working capacity along link \( a \). In this case, \( z_e = 0 \), and the only cost imposed upon the consumption of the sharable spare capacity along link \( e \) is \( \varepsilon \), as shown in the target function.

The following constraint imposes a bandwidth limitation upon the consumption of spare capacity.

\[
\sum_{\forall a \in b, \forall a \in A_w} x_a^k + n \cdot y_e^k \leq n \quad \text{for } 1 \leq k \leq k_{\text{max}}, \forall b \in B
\]  

\[
\forall e \in A_p, \text{ and } h_{e,b} + f_e < b(W)
\]  

where the value of the constant \( n \) is larger than the maximum number of arcs in the SRGs (e.g., it can be set to \(|V|\)). Eq. (2.4.24) ensures that if \( h_{e,b} + f_e < b(W) \), arc \( a \in b \) and \( e \) cannot be used at the same time for a working and protection segment in the same protection domain.
Note that $z_e$ is automatically transformed from $A'_p$ to $A_p$ such that values of $z'_e$ at those reverse arcs in $A'_p$ are set to zero.

It is clear that the adoption of the second graph has successfully defined all the three states for the protection path (see also Fig. 1.3) to take spare capacity, which are the case of $h_{e,b} = b(W)$, the case of $h_{e,b} + f_{a,v} = b(W) > h_{e,b}$, and the case of $h_{e,b} + f_{a,v} < b(W)$. The former two cases are jointly defined by Eq. (2.4.22) and Eq. (2.4.23), where $z_e$ is constrained no smaller than $1 - h_{e,b}/b(W)$ and 0 in the two cases, respectively; while the latter case is defined by Eq. (2.4.24), which prohibits the traversal of any protection segment through $e$ if there is no sufficient capacity along the arc.

To sum up the above, the shared protection problem has been formulated with the same spare link-state defined in Eq. (2.2.1), in which $z_e$ is equivalent to $r_j^{W_k}$ for all arcs of $e$ of the backup segment (the arc is denoted by $j$ in Eq. (2.2.1)). With the ILP formulation, we claim that all the three states for the protection path to take spare capacity can be well defined. Readers are requested to compare the resultant cost function adopted in the ILP formulation with that defined in Section 2.2.3. It can also be observed that the use of the residual graphs $A_w$ and $A_p$ along with the constraints of Eq. (2.4.22), Eq. (2.4.23) and Eq. (2.4.24) has imposed a bandwidth limitation constraint along each link upon the selection of working and protection segment of each protection domain, respectively. Without such a design, the extra constraint on the feasibility of spare capacity resource sharing and the link bandwidth limitation for protection paths can never be defined at the same time by using a single graph.

The number of variables in an ILP formulation directly influences the computation time required to solve the formulation. In this formulation, the number of variables is $(k_{max} + 4) \cdot |A_w| + (k_{max} + 3) \cdot |A_p|$, and the number of rows in the constraint matrix (where the linear formulation can be expressed in a general form as $A \times z = b$ with a target to minimize $z \times e$) is: $8 \cdot |A_w| + 9 \cdot |A_p| + 11 \cdot |V|$ plus the SRG constraints shown in Eq. (2.4.22) and (2.4.23). Therefore, the number of rows in the matrix $A$ has an upper bound $k_{max} \cdot |A_w| \cdot |A_p| + 8 \cdot |A_w| + 9 \cdot |A_p| + 11 \cdot |V|$. The computation time and memory occupation for each network topology adopted in this study will be shown in the next section.

### 2.4.1 Performance Evaluation

We conduct experiments to verify the proposed ILP model and the CDR algorithm on the network topologies Fig. 1.12(a) and (b) and Fig. 1.13 (a) and (b). The properties of the networks are summarized in Table 1.2.

The experiment is arranged as follows. Each directional link in the networks contains 32 units of bandwidth. We consider the capacity efficiency in terms of blocking probability for the dynamically arrived connection requests (of a single bandwidth unit) following the Poisson model and with a holding time defined in an exponential distribution function. For more details see Section 1.5.3. Each data point in the figures is the blocking probability for 100,000 connection requests using a specific survivable routing algorithm. The confidence interval is within 0.1% if we take the result of 100 connection requests as a trial.

We solve the above ILP formulation using CPLEX 7.5 and a Sun Ultra 80 workstation. The average computation time (including time taken by the pre-solver of CPLEX), memory occupation, the number of rows, columns and the number of non-zero elements of the constraint
matrix (after the CPLEX pre-solver reduce the size of the problem) are shown for all three cases in Table 2.1.

It is clear that the complexity of solving the ILP grows very quickly as the network size increases, which leads to a fact that the approach can hardly be applied for any on-line purpose. Therefore, it is positioned as a benchmark to evaluate the optimality achievable by the other heuristic counterparts. Table 2.2 is a demonstration on the optimality achieved by each scheme in the simulation. The offset from the optimality is defined as \( Q = c_a / c_{opt} - 1 \), where \( c_a \) is the total cost of the working and protection segments for the connection request by using a specific scheme, while \( c_{opt} \) is the cost achieved by solving the ILP formulation. The index \( Q \) is called Offset of Optimality; in this particular case, \( Q \) is an evaluation on the extent of resource sharing plus the extra (or unnecessary) cost taken by the working path. The smaller the value of \( Q \) is the better optimality has the heuristic algorithm achieved. Due to the very lengthy computation process in the 79- and 100-node networks, we only conduct this experiment in the 10-, 22- and 30-node networks, in which the ILP formulation is solved for every one of 1000 connection requests while the simulation is running in each case. Note that since the optimality is focused in this experiment, we do not consider any connection request that is blocked. Table 2.2 shows the average \( Q \) value and the average cost taken by each connection in the experiment using N22 and N30.

<table>
<thead>
<tr>
<th></th>
<th>Time (sec)</th>
<th>Memory (MB)</th>
<th>Rows</th>
<th>Columns</th>
<th>Non-zero</th>
</tr>
</thead>
<tbody>
<tr>
<td>N10</td>
<td>5.6</td>
<td>5</td>
<td>1772</td>
<td>281</td>
<td>6012</td>
</tr>
<tr>
<td>N22</td>
<td>255</td>
<td>23</td>
<td>10996</td>
<td>993</td>
<td>53534</td>
</tr>
<tr>
<td>N30</td>
<td>1185</td>
<td>42</td>
<td>31840</td>
<td>1203</td>
<td>99012</td>
</tr>
</tbody>
</table>

Table 2.1: The computation time and the amount of memory occupied for CPLEX solving the ILP formulation.

<table>
<thead>
<tr>
<th></th>
<th>CDR</th>
<th>PROMISE</th>
<th>OPDA</th>
<th>ILP</th>
</tr>
</thead>
<tbody>
<tr>
<td>N22</td>
<td>9.3%</td>
<td>10.1%</td>
<td>13.8%</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>2.210</td>
<td>2.226</td>
<td>2.301</td>
<td>2.022</td>
</tr>
<tr>
<td>N30</td>
<td>12.4%</td>
<td>13.9%</td>
<td>16.4%</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>2.439</td>
<td>2.478</td>
<td>2.533</td>
<td>2.176</td>
</tr>
</tbody>
</table>

Table 2.2: Offset of optimality \( (Q \) value) and the average cost taken by each connection request for all the schemes.

### 2.4.2 Conclusion

In this section we have studied dynamic survivable routing for a special type of protection, called segment shared protection, in which a novel Integer Linear Programming (ILP) formulation is proposed. We first define the segment shared protection, and demonstrate the advantages in using segment shared protection, which includes the facts that the restoration time can be shortened/guaranteed and that a higher possibility of resource sharing can happen between
different protection segments. We also define the spare link-state taking the SRG constraint into consideration. The ILP formulation is thus presented, where the switching/merging nodes and the corresponding least-cost working and protection segment pair for a connection request are jointly determined in the programming process. A novel approach of arc reversal along with a graph transformation method is devised to keep the formulation linear and to deal with the situation that the working segments of two neighbouring protection domains may overlap with each other by more than a single node. The ILP formulation is verified and compared with three reported schemes for the segment shared protection problem, namely CDR, PROMISE, and OPDA. We conclude that although solving the proposed ILP takes a much longer computation time than that an on-line algorithm is supposed to yield, the ILP nonetheless provides a means of evaluating the other heuristic algorithms for segment shared protection.
2.5 Claim 2.2: Extending the ILP of SSP with Constraints on Restoration Time and with Further Network Architectures

SSP is a special type of survivable routing, which has been proved to yield better flexibility and degree of sharing, and to be able to guarantee the restoration time once the working traffic is subject to any unexpected failure. This section tackles the problem of dynamic survivable routing by extending the Integer Linear Program (ILP) formulation introduced in the previous section for mesh communication networks in the complete routing information scenario. The proposed formulation can flexibly address the following two constraints: (1) the restoration time constraint for each connection request, and (2) the switching/merging (S/M) capacity constraint at each node. A novel approach, called SSP algorithm, is developed to reduce the extremely high computational requirement of solving the ILP formulation. Basically, our approach is to derive a good approximation on the parameters in the ILP by referring to the result of solving the corresponding SPP problem. Thus, the design space can be significantly reduced by eliminating some edges in the graphs. We will show in the simulation that with our approach, the optimality can be achieved in most of the cases. To verify the proposed formulation and investigate the performance impairment in terms of average cost and success rate by the additional two constraints, extensive simulation work has been conducted on three network topologies, in which SPP and SLP are implemented for comparison. Two novel performance metrics, Average Increase in Blocking Probability (AIBP) and Average Increase of Total Cost (AITC), are devised to define the importance of being S/M capable for each node. A novel recursive function is developed to evaluate the proposed performance metrics, in which both a brute force method and an enhanced method for reducing computation requirement are discussed. Simulation is conducted on two network topologies to verify the proposed approaches in evaluating the two performance metrics. We will demonstrate that the proposed SSP algorithm can effectively and efficiently solve the survivable routing problem with constraints on restoration time and S/M capability of each node. The comparison among the three protection types further supports SSP can yield significant advantages over SPP and SLP without taking much computation time.

2.5.1 Motivation

In the previous section, an ILP was provided to solve dynamic survivable routing for SSP. Relaxation methods are widely used to reduce the computation time for deriving approximate ILP solutions. Herzberg et. al. [59] formulated a linear programming (LP) model for the spare capacity assignment problem and treat spare capacity as continuous variables. A rounding process is used to obtain the final integer spare capacity solution which might not be feasible. They use hop-limited restoration routes to scale their LP problem. This technique can also be extended to input ILP formulation when Branch and Bound (BB) method is employed for searching the optimal solution [60], [61]. Lagrangian relaxation with subgradient optimization are used by Medhi and Tipper [62]. The Lagrangian relaxation [63] usually simplifies a hard original problem by dualizing the constraints and decomposing it into several easier sub-problems. Subgradient optimization is used to iteratively find the dual variables in these subproblems.

To improve the computation efficiency of the ILP, a novel approach, called SSP algorithm,
is proposed to reduce the runtime in solving the proposed ILP. Basically, our approach is to
derive a good approximation on the parameters in the ILP by referring to the result of solving
the corresponding ILP for shared path protection (SPP) such that significant reduction on the
design space can be achieved by eliminating some edges in the graphs. We will show in the
simulation that the optimality of the derived solution can be achieved in most of the cases with
our approach.

Although the formulation can effectively provide optimal solution according to a specific con-
nection request and network state, the following two assumptions leave space for improvement:

1. The size of each protection domain is not constrained so that the restoration time for each
   connection is not considered. We claim that an approach for constraining the restoration
time must be developed such that the class of service provisioning of survivable bandwidth
   is possible.
2. Every node can switch and/or merge restoration traffic at the same time, which may not be
   the case for practical applications. The major concern for this assumption is that the nodes
   serving as switching/merging devices need to provide extra signalling efforts and hardware
   responsiveness, which may not be general to the whole network.

To improve the above two shortcomings, this section is committed to extending the ILP for-
mulation of the previous section, such that the constraint on restoration time in each protection
domain of a connection as well as the constraint on the ability for each node to switch/merge
restoration traffic are well addressed by classifying each node into four categories, each having
different switching/merging capability. It is important to note that the nodes serving as S/M
devices for SSP must provide extra signalling efforts and hardware responsiveness especially
in the transparent optical plane. Thus, equipping a network node with S/M capability in
the optical domain should be taken as a network resource instead of being taken as a general
assumption. The performance impact by the allocation of S/M capability in each node, there-
fore, turns out to be an interesting problem that is subject to further efforts of network-wide
planning according to the topology, traffic pattern, and the corresponding survivable routing
algorithm. This paper is committed to addressing the performance impact by removing the
S/M capability of a specific node in the network when SSP is implemented for each connection.
Two performance metrics are defined: the average increase in blocking probability caused by
removing the S/M capability at a specific node (denoted by AIBP), and the average increase of
cost in allocating a connection request if the S/M capability at a specific node is removed (de-
noted by AITC). Two realistic network topologies, European Reference Network (ERNET) and
North-American Reference Network (NARNET), are adopted, where a traffic matrix in year
2005 is estimated for each network according to [38]. To evaluate the two performance metrics,
two methods are discussed: the brute force method and the enhanced recursive method. In the
former, the SSP algorithm is applied for all the connection requests on each possible state of
S/M capability; while in the latter, a novel approach is devised to reduce the computational
requirement induced in the brute force method. For both of the approaches, a corresponding
recursive function is developed for this purpose.

To compare SSP with the other two types of protection, namely Shared Path Protection
(SPP) and Shared Link Protection (SLP) [64], modification is made upon the ILP formulation
of SSP to implement the two schemes. Although similar studies have been widely reported, all of them are based on heuristic approaches which may be somewhat specific to traffic patterns, network topologies, or availability of the network resources. Among the comparisons of the three type of protection, we claim that this is the first study using ILP formulation yielding optimal solution.

2.5.2 A Network Modeling

The input of SSP was defined in Section 2.3. With \( G(V, A) \), each node of the network is classified into the following four sets:

\( V_f \subset V \) the vertices of \( G \) assigned to nodes which can serve as both switching and merging node.

\( V_s \subset V \) the vertices of \( G \) assigned to nodes which can serve as a switching but not a merging node.

\( V_m \subset V \) the vertices of \( G \) assigned to nodes which can serve as a merging but not a switching node.

\( V_i \subset V \) the vertices of \( G \) assigned to nodes which can serve as neither a switching nor as a merging node.

Based on the modeling of restoration time in [2], a parameter denoted as \( s_{a,b} \) is defined for each link, which represents the propagation delay of the signalling on link \( a, b \). A global parameter \( s_{max} \) is taken to denote the limit of the restoration time for the connection request, and could be defined in the service level agreement. The restoration time of the connection request is defined as the maximum of the restoration time of all segments.

2.5.3 ILP Formulation for SSP with Constraints on Restoration Time and Adapting Further Network Architectures

This section presents the additional constraints of the ILP formulation presented in Section 2.4. The target function of the ILP remains as it was defined in Eq. 2.4.1, since the cost function is unchanged. Constraints ensuring the special structure of path \( W \) and \( Q \) are also kept (Eq. (2.4.2), (2.4.3), (2.4.4)), while the equations assigned to variables \( \hat{x} \) and \( \hat{y} \) need to be replaced (Eq. (2.4.5), (2.4.6), (2.4.7), (2.4.8)), since each node may not necessarily be able to serve as a switching and/or merging node of a protection domain. Indexes \( \gamma_a \) and \( \delta_a \) shown in Table 2.3 are introduced to depict whether or not vertex \( a \), where \( a \in V \), can switch and/or merge the corresponding affected traffic.

<table>
<thead>
<tr>
<th></th>
<th>( V_f )</th>
<th>( V_s )</th>
<th>( V_m )</th>
<th>( V_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \gamma_a )</td>
<td>+1</td>
<td>+1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>( \delta_a )</td>
<td>+1</td>
<td>-1</td>
<td>+1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2.3: The indexes for the attributes of node \( a \), where \( a \in V \). For example, if \( a \in V_f \), both \( \gamma_a \) and \( \delta_a \) are +1 in Eq. (2.5.2), Eq. (2.5.3), Eq. (2.5.6) and Eq. (2.5.7).
For \( \hat{x}_{a,b} \), we have the following constraints:

\[
(2k_{\text{max}} - 1) \cdot x_{a,b} \geq \hat{x}_{a,b} \geq 0 \quad \text{for } (a,b) \in A_w \tag{2.5.1}
\]

\[
\sum_{(a,b) \in A_w, a \neq d} \hat{x}_{a,b} - \sum_{(a,b) \in A_w, a \neq s} \hat{x}_{a,b} \geq -2k_{\text{max}} + \gamma_a \cdot \sum_{(a,b) \in A'_w, a \neq d} y'_{a,b} + \\
\delta_a \cdot \sum_{(a,b) \in A'_w, a \neq s} y'_{a,b} \cdot \hat{x}_{a,b} + \\
\sum_{(a,b) \in A_w, a \neq d} k_{\text{max}} \cdot x_{a,b} + \\
\sum_{(a,b) \in A_w, a \neq s} k_{\text{max}} \cdot x_{a,b} \quad \forall a \in V \ a \neq s, d \tag{2.5.2}
\]

\[
\sum_{(a,b) \in A_w, a \neq d} \hat{x}_{a,b} - \sum_{(a,b) \in A_w, a \neq s} \hat{x}_{a,b} \leq \gamma_a \cdot \sum_{(a,b) \in A'_w, a \neq d} y'_{a,b} + \\
\delta_a \cdot \sum_{(a,b) \in A'_w, a \neq s} y'_{a,b} \cdot \hat{x}_{a,b} + \\
\sum_{(a,b) \in A_w, a \neq d} k_{\text{max}} \cdot x_{a,b} + \\
\sum_{(a,b) \in A_w, a \neq s} k_{\text{max}} \cdot x_{a,b} \quad \forall a \in V \ a \neq s, d \tag{2.5.3}
\]

\[
\sum_{(s,b) \in A_w} \hat{x}_{s,b} = 1 \quad \forall b \in V \tag{2.5.4}
\]

The constraint in Eq. (2.5.1) ensures that \( \hat{x}_{a,b} \) is upper-bounded by \((2k_{\text{max}} - 1)\), and is nonzero only if \( W \) passes through link \((a, b)\). Eqs. (2.5.2) and (2.5.3) function as a special type of flow conservation constraint upon the net change of \( \hat{x} \) at vertex \( a \) in the network, which is expressed at the left-hand side of the equations. With different attributes of network nodes, however, the criterion of flow conservation on \( \hat{x} \) at each vertex is different. Without loss of generality, the two inequalities are explained in details only in the event that the corresponding nodes are assigned to \( V_f \). Four situations are defined for vertex \( a \) when it is taken by \( W \), where \( \forall a \in V_f, a \neq s, d \), and \( \gamma_a = \delta_a = 1 \): (a) \( Q \) merges back to \( W \) at \( a \); (b) \( Q \) switches out of \( W \) at \( a \); (c) \( Q \) merges back and switches out of \( W \) at \( a \); (d) otherwise.

<table>
<thead>
<tr>
<th>description</th>
<th>( V_f )</th>
<th>( V_s )</th>
<th>( V_m )</th>
<th>( V_l )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( W ) intersects ( P ) at ( a )</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( W ) exits ( P ) at ( a )</td>
<td>1</td>
<td>1</td>
<td>forbidden</td>
<td>forbidden</td>
</tr>
<tr>
<td>( W ) enters ( P ) at ( a )</td>
<td>1</td>
<td>forbidden</td>
<td>1</td>
<td>forbidden</td>
</tr>
<tr>
<td>otherwise</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2.4: The ‘flow increase’ at different type of vertices (i.e., \( \sum_{(a,b) \in A'_w, a \neq d} y'_{a,b} - \sum_{(a,b) \in A'_w, a \neq s} y'_{a,b} \))

or \( \sum_{(a,b) \in A_w, a \neq d} \hat{x}_{a,b} - \sum_{(a,b) \in A_w, a \neq s} \hat{x}_{a,b} \)

Now we consider the change of the value of \( \hat{x} \) from the source node (where \( \hat{x}_{s,b} = 1 \)) to the destination (where \( \hat{x}_{b,d} = 2k_{\text{max}} - 1 \)). Table 2.4 summarizes the four possible situations of increasing \( \hat{x} \) along \( W \) (as well as \( \hat{y} \) that will be seen later) at vertex \( a \), which are also shown at the right hand side of Eq. (2.5.2) and Eq. (2.5.3). In Eq (2.5.2) the value of \( \hat{x}_{a,b} \) of vertex \( a \in V_f \) along \( W \) increases by 1 at vertex \( a \) in the situations (a) and (b), and increases by 2 at vertex \( a \) in the situation (c), and is unchanged otherwise. The constraint on the net change in terms of \( \hat{x}_{a,b} \) has a lower bound specified at the right-hand side of the equation, where the term \( \sum_{(a,b) \in A_w, a \neq d} k_{\text{max}} \cdot x_{a,b} + \sum_{(a,b) \in A_w, a \neq s} k_{\text{max}} \cdot x_{a,b} - 2k_{\text{max}} \) checks if vertex \( a \) is taken by \( W \).
It is clear that the term is 0 if node \( a \) is traversed by \( W \), and is \(-2k_{\text{max}}\) otherwise. Therefore, even if node \( a \) is not taken by \( W \), Eq. (2.5.2) still holds.

To take a more detailed look at the right-hand side of Eq. (2.5.2), in the case of (a), the increase of \( \hat{x}_{a,b} \) is 1 since \( \sum_{(a,b) \in \mathcal{A}_{p}, a \neq d} y_{a,b} = 0 \) and \( \sum_{(b,a) \in \mathcal{A}_{p}, a \neq s} y_{a,b} = 1 \) (since \( Q \) merges back to \( W \) at vertex \( a \)). In case (b), the increase of \( \hat{x}_{a,b} \) is still 1 because \( \sum_{(a,b) \in \mathcal{A}_{p}, a \neq s} y_{a,b} = 0 \) and \( \sum_{(a,b) \in \mathcal{A}_{p}, a \neq d} y_{a,b} = 1 \). In situation (c), increase of \( \hat{x}_{a,b} \) is 2 since \( \sum_{(a,b) \in \mathcal{A}_{p}, a \neq d} y_{a,b} = \sum_{(a,b) \in \mathcal{A}_{p}, a \neq s} y_{a,b} = 1 \). If node \( a \) is neither a switching nor a merging node (but it is taken by \( W \)), the right-hand-side of Eq. (2.5.2) becomes 0, in which no change upon \( \hat{x}_{a,b} \) is required.

Basically, Eq. (2.5.3) is devised based on the same idea as Eq. (2.5.2) except that when node \( a \) is not taken by \( W \), the right-hand side becomes 0 instead of \(-2k_{\text{max}}\). Both Eq. (2.5.2) and (2.5.3) constrain the net change of the value of \( \hat{x}_{a,b} \) to be either 0, 1, or 2, at any vertex \( a \) taken by \( W \), depending on if path \( Q \) switches over and/or merges back to \( W \) at vertex \( a \). Eq. (2.5.4) sets \( \hat{x}_{a,b} \) to 1 if vertex \( a \) is assigned to the source node.

For \( \hat{y}_{a,b} \), we have the following constraints:

\[
k_{\text{max}} \cdot (2k_{\text{max}} - 1) \cdot \hat{y}_{a,b} \geq \hat{y}_{a,b} \geq 0 \quad \text{for } (a,b) \in A_{p}'
\]

\[
\sum_{(a,b) \in A_{p}', a \neq d} \hat{y}_{a,b} - \sum_{(b,a) \in A_{p}, a \neq s} \hat{y}_{a,b} \geq \sum_{(a,b) \in A_{p}', a \neq d} k_{\text{max}} \cdot \hat{y}_{a,b} + \sum_{(b,a) \in A_{p}, a \neq s} k_{\text{max}} \cdot \hat{y}_{a,b} + \sum_{(a,b) \in A_{p}, a \neq s} \gamma_{a} \cdot (x_{a,b} - y_{a,b}^{-}) + \sum_{(b,a) \in A_{p}, a \neq s} \delta_{a} \cdot (x_{a,b} - y_{a,b}^{-}) - 2k_{\text{max}}
\]

\[
\sum_{(a,b) \in A_{p}', a \neq d} \hat{y}_{a,b} - \sum_{(b,a) \in A_{p}, a \neq s} \hat{y}_{a,b} \leq \sum_{(a,b) \in A_{p}', a \neq d} \gamma_{a} \cdot (x_{a,b} - y_{a,b}^{+}) + \sum_{(b,a) \in A_{p}, a \neq s} \delta_{a} \cdot (x_{a,b} - y_{a,b}^{+}) \quad \forall a \in V \ a \neq s, d
\]

\[
\sum_{(b,a) \in A_{p}, a \neq s} \hat{y}_{a,b} = 1 \quad \forall b \in V
\]

The constraint in Eq. (2.5.5) ensures that \( \hat{y}_{a,b} \) is non-zero only if \( Q \) passes through arc \((a,b)\).

The idea behind Eq. (2.5.6) and Eq. (2.5.7) is similar to that of Eq. (2.5.2) and Eq. (2.5.3). The only difference is that instead of considering the forward arcs of \( Q \) (denoted as \( y_{a,b}^{-} \)), the term \((x_{a,b} - y_{a,b})\) is used, which is non-zero for link \((a,b)\) along \( W \) while not being taken by \( Q \). Therefore, Eq. (2.5.6) and (2.5.7) ensures that the value of \( \hat{y}_{a,b} \) on path \( Q \) increases by 1 at node \( a \) only if \( Q \) merges back to \( W \) or \( Q \) switches out of \( W \) at vertex \( a \). The constraint in Eq. (2.5.8) is to set \( \hat{y}_{a,b} \) as 1 if vertex \( a \) is assigned to the source node, and that there would be only a single protection link stretching out of the source node. Please refer to Fig. 2.6 for an explicit illustration for the variables formulated above. It can be easily observed that the maximum of \( \hat{x}_{a,b} \) is \( 2k_{\text{max}} - 1 \); while the maximum of \( \hat{y}_{a,b} \) is less than \( k_{\text{max}} \cdot (2k_{\text{max}} - 1) \) because path \( Q \) may traverse the same link at most \( k_{\text{max}} \) times.

In addition to the modification on the constraints upon the variables \( \hat{x} \) and \( \hat{y} \), an extra constraint is required on the routing of path \( Q \) for the node set \( S_{s} \), \( V_{m} \), and \( V_{t} \), respectively. At any vertex in \( S_{s} \), \( Q \) should not merge back to path \( W \), thus we have:
\[ \sum_{(a,b) \in A_w, a \neq d} y_{a,b} \geq \sum_{(b,a) \in A_w, a \neq s} y_{a,b} \quad \forall a \in V_s \quad a \neq s, d \] (2.5.9)

At any vertex in \( V_m \), \( Q \) should not switch out of \( W \), we have:

\[ \sum_{(a,b) \in A_w, a \neq d} y_{a,b} \leq \sum_{(b,a) \in A_w, a \neq s} y_{a,b} \quad \forall a \in V_m \quad a \neq s, d \] (2.5.10)

At any vertex in \( V_l \), \( Q \) should neither merge back nor switch out of path \( W \), thus we have:

\[ \sum_{(a,b) \in A_w, a \neq d} y_{a,b} = \sum_{(b,a) \in A_w, a \neq s} y_{a,b} \quad \forall a \in V_l \quad a \neq s, d \] (2.5.11)

With \( \hat{x} \) and \( \hat{y} \) link labels, \( W \) is divided into segments such that each link along it is covered by at least one protection segment. Similarly to Section 2.4 path \( W \) and \( Q \) are divided into segments, for that Eqs. (2.4.14), (2.4.15), (2.4.16), (2.4.19), (2.4.20), (2.4.21) are kept in the formulation.

In order to address the constraint on the restoration time for a connection, the size of each protection domain for the connection must be upper-bounded. Therefore, the following constraint is appended:

\[ \sum_{\forall (a,b) \in A_w} \varsigma_{a,b} \cdot x_{a,b}^k + \sum_{\forall (a,b) \in A_p} \varsigma_{a,b} \cdot y_{a,b}^k \leq \varsigma_{\text{max}} \quad \text{for} \quad 1 \leq k \leq k_{\text{max}}, \] (2.5.12)

where \( \varsigma_{a,b} \) is the delay of link \((a,b)\), which is modeled as the length of the link divided by the light speed. Eq. (2.5.12) brings forth \( k_{\text{max}} \) constraints, which guarantees the sum of the physical length of the working and the corresponding protection segment for each protection domain being constrained. Finally the SRG constraints (Eqs. (2.4.22), (2.4.23), (2.4.24)) should be kept in order to formulate the problem.

The number of variables in an ILP formulation have not changed, however the number of constraints slightly increased.

### 2.5.4 A Heuristic of Improving the Runtime of Solving ILP: The SSP Algorithm

It is clear that the computational requirement for solving the above ILP is huge and the runtime strongly depends on the parameter \( k_{\text{max}} \). Thus, selecting a proper value of \( k_{\text{max}} \) will significantly improve the computation efficiency. Furthermore, some links in the network are very unlikely to be taken by any working or protection segment, and can be simply excluded for achieving better computation efficiency without losing the optimality. We are motivated by the above observation in the design of the SSP algorithm that is committed to speed up solving the ILP.

In the SSP algorithm, a pre-calculation mechanism is devised to estimate the value of \( k_{\text{max}} \) and to exclude those edges with little chance of being taken by the working and protection segments in \( A_w \) and \( A_p \). The first step of the pre-calculation is to derive a feasible solution, which can be done either by a heuristic algorithm (like Maximum Likelihood Relaxation see Section 1.5.2) or by solving the ILP problem for SPP, see Section 1.4.1. In this section we use the latter, with the following equation being added, which addresses the restoration time constraint:
\[
\sum_{\forall (a,b) \in A} \xi_{a,b} \cdot x_{a,b} + \sum_{\forall (a,b) \in A} \xi_{a,b} \cdot y_{a,b} \leq \xi_{\max}
\] (2.5.13)

where \(x_{a,b}\) and \(y_{a,b}\) are binary flow indicators of the working and protection path, respectively.

Fig. 2.7 shows the flowchart of the SSP algorithm. At the beginning, the algorithm tries to find a feasible solution in the pre-calculation; if successful, it can serve as an upper bound on the problem since SPP is a special case of SSP. In (2) of the flowchart, if the working path of SPP is one-hop, the searching process is terminated in (3) because the derived solution must be the optimal solution for the SSP case, and it goes to (4) otherwise. If the working path consists of more hops and it intersects with the protection path, the nodes of intersection can be treated as switching and merging nodes, by which a feasible solution with a better upper bound is identified.

Let us define a detour factor of each edge assigned to the connection and network state, denoted as \(\xi_{a,b}^{w}\), and can be calculated with the following formula:

\[
\xi_{a,b}^{w} = sp(G_w, s, a) + c_{a,b} + sp(G_w, b, d)
\] (2.5.14)

where \(sp(G_w, s, a)\) represents the cost of the shortest path in \(G_w\) between \(s\) and \(a\); \(c_{a,b}\) represents the cost of link \((a, b)\), and \(sp(G_w, b, d)\) is the cost of the shortest path between \(b\) and \(d\). The detour factor shows the minimal detour compared with that of the shortest path if the working path passes through link \((a, b)\). Let us define \(c_{\text{pre}(W)}\) and \(c_{\text{pre}(P)}\) as the cost of the working and protection path derived in the pre-calculation, respectively. Let us define \(\xi_{\max}\), which gives an upper bound on the detour of the optimal solution. In the first step, \(\xi_{\max}\) can be set to \(c_{\text{pre}(W)} - sp(G_w, s, d) + c_{\text{pre}(P)}\), since the pre-calculated solution was feasible. As a consequence, arcs with \(\xi_{a,b}^{w} > \xi_{\max}\) can be removed from \(A_w\). As a result, in (5) of the flowchart, \(A_w\) will have all the edges with \(\xi_{a,b}^{w} \leq \xi_{\max}\) and \(b(W) \leq f_j\).

After \(|A_w|\) is reduced, a much better upper bound on the available restoration capacity, \(m_j\), can be derived, compared with using \(f_j + v_j\), as shown on (6) and (7) of the flowchart. Obviously, the working path will take at least one edge of any cut between \(s\) and \(d\) in \(A_w\). We can get an upper bound on the restoration capacity by selecting a cut, analyzing the failure of each arc of the cut, and taking the minimum on the sharable capacity plus the free capacity; in other words,

\[
m_j = \max_{C \text{ is a cut of } A_w} \min_{j \in C} h_j + f_j
\] (2.5.15)

With more cuts being analyzed, a better upper bound can be derived, with which \(A_p\) can be defined such that all the edges with \(b(W) \leq m_j\) are contained. Thus, in (8) of the flowchart, we can derive a lower bound on the cost of each edge taken by the protection path as:

\[
\xi_{a,b}^{p} = \max \left\{ \frac{f_{a,b} - m_{a,b} + b(W)}{b(W)}, 0 \right\} \cdot c_{a,b}
\] (2.5.16)

The above relationship holds since \(f_{a,b} - m_{a,b}\) is the lower bound of the sharable capacity and \(\max \left\{ \frac{f_{a,b} - m_{a,b} + b(W)}{b(W)}, 0 \right\} \cdot c_{a,b}\) gives a lower bound on the ratio of the sharable capacity. With this, we can derive the detour factor of each edge in \(A_p\) (denoted as \(\xi_{a,b}^{p}\)), which is specific to the connection request and the current link-state.
\[ \xi_{a,b}^p = sp(G_p, s, a) + c_{a,b}^p + sp(G_p, b, d) \]  \hspace{1cm} (2.5.17)

where \( sp(G_p, s, a) \) represents the total cost of the shortest path in \( G_p \) between the source node and node \( a \), where \( c_{a,b}^p \) is the cost of \( (a, b) \) in \( G_p \) and \( sp(G_p, b, d) \) is the distance between node \( b \) and the destination node of the demand. With \( G_p, b, d \) we can derive a better \( \xi_{\text{max}} \), such that \( \xi_{\text{max}} = c_{\text{pre}}(W) - sp(G_w, s, d) + c_{\text{pre}}(P) - sp(G_p, b, d) \). Arrows with \( \xi_{a,b}^p > \xi_{\text{max}} \) are not included in \( A_p \) since the optimal protection path will never pass through it due to its large amount of detour. As a result as it is shown in (9) of the flowchart \( A_p \) will have all the edges where \( \xi_{a,b}^p \leq \xi_{\text{max}} \) and \( b(W) \leq m_j \). In (10) of the flowchart, if \( sp(G_p, s, d) > 0 \), the algorithm switches back to further reduce \( \xi_{\text{max}} \).

If no feasible solution of SPP is derived in (1), a heuristic approach is developed to set \( \xi_{\text{max}} \) described as follows. In (11) of the flowchart, we solve the SPP again with the restoration time constraint relaxed. If succeed, we go to (12) and remove the edges of the working path according to our rule of thumb such that \( \xi_{\text{max}} = 0.7 \cdot c_{\text{pre}}(W) \), and \( \xi_{\text{max}} = c_{\text{pre}}(W) \) if the restoration constraint is very tight. If the solving of SPP with the restoration constraint being relaxed still fails, we go to (13) and calculate the shortest path, and in (14) we remove the edges for the working path similarly with \( \xi_{\text{max}} = 1.2 \cdot c_{\text{pre}}(W) \), \( \xi_{\text{max}} = 2 \cdot c_{\text{pre}}(W) \), and \( c_{\text{pre}}(P) = 0 \). Obviously, if the shortest-path-first algorithm fails to find any solution, we go to (15) to halt the algorithm.

Finally in (17) we set the value of \( k_{\text{max}} \) such that the following two quantities are considered: (a) the hop count of the shortest path between the source and destination (denoted as \( |W| \)), and (b) the number of intersections of the working and protection paths at nodes capable to switching and merging restoration traffic, denoted as \( |\text{feasible segments}| \):

\[ k_{\text{max}} = \left( |W| - |\text{feasible segments}| \right) / 2 \]  \hspace{1cm} (2.5.18)

With the SSP algorithm, the runtime of solving the ILP of SSP can be significantly reduced while the quality of the result is almost always guaranteed or very close to the optimal one. We will further support the heuristic in the following section.

### 2.5.5 Evaluation of the S/M Capability of the Network

Since each node can be either S/M capable or S/M non-capable, a Boolean array \( \psi \) with a size \( |N| \) is defined to represent the current network configuration, where \( \psi[i] \) is "1" if node \( i \) can switch and merge restoration traffic, and "0" otherwise. In general, the above mentioned performance metrics stand for nothing but the difference made by removing the S/M capability at a specific node based on the pre-defined traffic pattern (or a set of connection requests). To evaluate the two metrics, we can simply use a brute force method in which the SSP algorithm is solved for \( |\psi| \times k_t \) times, where \( |\psi| = 2^{|N|} \) is the number of all possible states of \( \psi \) array and \( k_t \) is the number of connection requests to be launched.
Figure 2.7: The flowchart of pre-calculation process, which reduces $A_p$ and $A_w$ while the optimality of the result is still guaranteed.

The Proposed Performance Metrics: AIBP and AITC

To precisely define the two performance metrics, let the solution for the $k^{th}$ connection request under a specific array $\psi$ be termed a trial, which is determined by the network state $G_k$ (when the $k^{th}$ connection request is launched), the source node $s$ and destination node $d$, and the bandwidth $b(W)$ of the connection request. In other words, $(G_k, s, d, b(W), \psi)$ can explicitly identify a trial. We also define that a trial is a counterpart of another if the two trials are performed based on absolutely the same conditions except that $\psi[i]$ is flipped (between "1" and "0").

To evaluate the two metrics, totally $k_t$ connection requests are launched for a single trial such that $k_t$ counterparts are examined, in which the solving of the SSP algorithm on each trial of a counterpart returns two data: a Boolean parameter indicating whether the request is blocked due to the flip of $\psi[i]$ from "1" to "0" as well as a non-negative number representing the cost increase due to the flip of $\psi[i]$ from "1" to "0". Based on the returned data, we can derive the increase of blocking probability and the increase of total cost at node $i$ by arithmetically averaging the results based on all the possible $\psi$ arrays. For the increase of blocking probability at node $i$ with the $k^{th}$ connection request, the increase of blocking probability ($IBP(i)$) is:
\[ \text{IBP}(i,k) = \sum_{\forall \psi, \psi'[i]=0} \text{SSP\_block}(G_k, s, d, b(W), \psi) - \sum_{\forall \psi, \psi'[i]=1} \text{SSP\_block}(G_k, s, d, b(W), \psi) \]
\[ = \sum_{\forall \psi} \text{SSP\_block}(G_k, s, d, b(W), \psi) - \text{SSP\_block}(G_k, s, d, b(W), \psi)_{[\psi'[i]=1]} \]  
\[ \text{(2.5.19)} \]

where \( \text{SSP\_block}(G_k, s, d, b(W), \psi) \) is a Boolean function that returns "1" when the \( k \)th connection request is blocked under a specific state of \( \psi \) array and traffic distribution \( G_k \), and "0" otherwise. \( \text{IBP}(i,k) \) may return zero when both terms in the right side of the equation are 0 or 1. The average increase of blocking probability (\( AIBP \)) at node \( i \) is simply the average of the results by launching all the connection requests:
\[ AIBP(i) = \frac{\sum_{\forall k} \text{IBP}(i,k)}{k_i} \]  
\[ \text{(2.5.20)} \]

For the increase of total cost (\( ITC \)) at node \( i \) with the \( k \)th connection request \( (G_k, s, d, b(W)) \) can be formulated using a similar concept as:
\[ ITC(i,k) = \sum_{\forall \psi, \psi'[i]=0} \text{SSP\_cost}(G_k, s, d, b(W), \psi) - \sum_{\forall \psi, \psi'[i]=1} \text{SSP\_cost}(G_k, s, d, b(W), \psi) \]
\[ = \sum_{\forall \psi} \text{SSP\_cost}(G_k, s, d, b(W), \psi) - \text{SSP\_cost}(G_k, s, d, b(W), \psi)_{[\psi'[i]=1]} \]  
\[ \text{(2.5.21)} \]

where \( \text{SSP\_cost}(G_k, s, d, b(W), \psi) \) returns the cost of the derived solution by the SSP algorithm regarding to the \( k \)th connection request, \( G_k \), and a specific state of array \( \psi \). The average increase of cost (\( AITC \)) at node \( i \) is simply the average of the results by launching all the connection requests:
\[ AITC(i) = \frac{\sum_{\forall k} ITC(i,k)}{k_i} \]  
\[ \text{(2.5.22)} \]

**Evaluation of the Two Performance Metrics**

With the brute force method, all the possible states of array \( \psi \) are examined, which can be implemented with the following recursive function

**Set** \( \psi[j] = 1 \) for all \( j \)

last\_cost:= undefined

last\_node:= undefined

Analyze(trial\( (G,s,d,b(W),\psi)\),int last\_cost,node last\_node)

if (there was last\_node)

if SSP\_block\( (G,s,d,b(W),\psi) \)

IBP(last\_node)=IBP(last\_node)+1/2\( ^{|N|} \)

int cost:=undefined

else

int cost:=SSP\_cost\( (G,s,d,b(W),\psi) \)

ITC(last\_node)=ITC(last\_node)+(cost-last\_cost)/2\( ^{|N|} \)

// call recursively the function to discover all states of \( \psi \) array for all node \( x \) in \( G \)

if \( \psi[x] == \"" \)

\( \psi' := \psi \); \( \psi'[x] := \"0" \)

Analyze(trial\( (G,s,d,b(W),\psi')\),cost,x)

\( \text{(a)} \)

\( \text{(b)} \)

\( \text{(c)} \)
At the very beginning of the recursion, the trial \((G_k, s, d, b(W), \psi[j] = 1 |_{j=1..N})\) is taken as the root, and an SSP solution is derived on the trial. The result of solving each trial is taken to derive the cumulative \(ITC\) and \(IBP\) at line (a) and line (b), respectively. The recursive function \(\text{Analyze}\) is called with three parameters: the trial, the \text{last_node} and the \text{last_cost}. The \text{last_node} is the node whose S/M capability is just removed, and the \text{last_cost} is the cost before removing the S/M capability. These values are undefined at the root. The function \((\text{there was last_node})\) is true if \text{last_node} is defined, thus it is false only at the root of the recursion. If \(\text{SSP\_block}(G,s,d,b(W),\psi)\) returns a failure after flipping \(\psi[\text{last_node}]\) from \(\text{"1"}\) to \(\text{"0"}\), \(IBP(\text{last_node})\) is incremented by the number of states of \(\psi\) array that yield the same solution. Similarly, if \(\text{SSP\_block}(G,s,d,b(W),\psi)\) returns a success, we need to add to \(ITC(\text{last_node})\) the cost increase after flipping \text{last_node}. In order to consider all the possible states of \(\psi\) array; it recursively calls the function \(\text{Analyze}\) such that one more node is set to S/M incapable than that in the previous recursion. Since the recursion analyzes all possible states of array \(\psi\), we need to divide \(IBP\) and \(ITC\) with \(|\psi| = 2^{|N|}\) to get the increment for the mean values.

An instance is defined as a state of \(\psi\) array, which is nothing but a trial. It is clear that at the root, the recursion is at the first level with a single instance, and has \(|N|\) instances in the second layer, each of which is a state of \(\psi\) array with \((|N| - 1)\) elements to be \(\text{"1"}\) and one element to be \(\text{"0"}\). In the \(s^{th}\) level, generally, we have all the possible states of \(\psi\) array with \((|N| - s)\) elements to be \(\text{"1"}\) and \(s\) elements to be \(\text{"0"}\), where \(\binom{|N|}{s}\) instances are included. This recursion guarantees to discover all possible states of \(\psi\) array.

Although the brute force method can sufficiently define the two performance metrics for the whole network, it will certainly induce intolerably long computation time. In order to reduce the calculation time, the recursive method is improved by preventing from inspecting trials with an identical result. AIPB and AITC can be evaluated with much simpler calculation. The following lemma serves as a basis for the calculation reduction.

**Lemma 1:** The cost of a trial is never smaller than its \(i\)-counterpart if the former has \(\psi[i] = 0\).

**Proof.** After flipping \(\psi[i]\) from \(\text{"1"}\) to \(\text{"0"}\), an additional constraint is added to the solving the SSP algorithm, thus the solution cannot be smaller. \(\square\)

The fact that employing the brute force method may redundantly calculate a huge number of trials with no significance to the result, can be easily observed through the following example. Let a trial \((G_k, s, d, b(W), \psi[j] = 1 |_{j=1..N})\) return that the \(l^{th}\), \(m^{th}\) and \(n^{th}\) nodes serve as the S/M nodes. It is clear that flipping any node(s) other than the three from \(\text{"1"}\) to \(\text{"0"}\) will not affect the result since only when any of the three elements in the \(\psi\) array is flipped can a new result come up. In this case, calculation efforts of \(2^{|N|-3}\) trials are saved compared with the case by the brute force method, and these trials are called “don’t care trials”. The “don’t care” nodes of the don’t care trial are the nodes which can be \(\text{"1"}\) or \(\text{"0"}\) \((l, m, n\ in \ the \ example)\) \(2^{|N|-3}\ (G_k, s, d, b(W), \psi)\). Another important observation in this example is that only when \(i\) equals to \(l\), \(m\), or \(n\) could it be possible that the results of solving the \(i\)-counterparts are different. As consequence the recursion needs to be called only with \(\psi[l] = \text{"0"}\) and \(\psi[m] = \text{"0"}\) and \(\psi[n] = \text{"0"}\) at line (c) to discover all possible \(\psi\) arrays.
According to the above observations, the enhanced recursion identifies the „don’t care nodes” and never calls the recursion with them to prevent any redundant calculation. With the enhanced recursion all the different solutions on trials \((G_k, s, d, b(W))\) are derived. Let \(\phi_k\) denote the set of \(\psi\)-arrays that are solved in the enhanced recursion by the SSF algorithm for the \(k^{th}\) connection request. Let the \(\psi\)-array yielding the \(j^{th}\) feasible solution at the \(r^{th}\) layer of the recursion for the \(k^{th}\) connection request be denoted as \(\phi_{j,r,k}^{feas}\) and the \(j^{th}\) \(\psi\)-array blocked at the \(r^{th}\) layer of the recursion for the \(k^{th}\) connection request denoted as \(\phi_{j,r,k}^{block}\). Trial \(\phi_{j,r,k}^{feas}\) (and \(\phi_{j,r,k}^{block}\)) is classified in \(|N|\) long arrays \(\phi_{j,r,k}^{feas}(i) (\phi_{j,r,k}^{block}(i))\), such that \(\phi_{j,r,k}^{feas}(i) (\phi_{j,r,k}^{block}(i))\) is part of \(\phi_{j,r,k}^{feas}(i) (\phi_{j,r,k}^{block}(i))\) if node \(i\) was not a don’t care node of \(\phi_{j,r,k}^{feas} (\phi_{j,r,k}^{block})\). The proposed recursive function is to identify \(\phi_{j,r,k}^{feas}(i) (\phi_{j,r,k}^{block}(i))\), for \(k = 1\) to \(k_t\), \(r = 1\) to \(|N|\), and \(j = 1\) to \(|\phi_{r,k}^{feas}(i)|\) or \(|\phi_{r,k}^{block}(i)|\), respectively. Let \(SM_{j,r,k}\) be the set of \(S/M\) nodes in the \(j^{th}\) feasible solution at the \(r^{th}\) layer of the recursion of the \(k^{th}\) connection request. The don’t care nodes of \(\phi_{j,r,k}^{block}\) are the \(S/M\) capable nodes (nodes with \(\psi[i] =\) “1”). The don’t care nodes of \(\phi_{j,r,k}^{feas}\) are the \(S/M\) capable nodes (nodes with \(\psi[i] =\) “0” except the nodes of \(SM_{j,r,k}\). The number of “don’t care” trials corresponding to all \(\phi_{j,r,k}^{block}\) equals to \(2^{|N|-r}\), and the number of “don’t care” trials corresponding to all \(\phi_{j,r,k}^{feas}\) equals to \(2^{|N|-|SM_{j,r,k}|}-r\). Eq. (2.5.19) and Eq. (2.5.21) can be thus modified as follows:

\[
IBP(i, k) = \frac{|N|}{\sum_{r=1}^{k_t} \sum_{j=1}^{|N|} SSP_{\text{blocking}}(G_k, s, d, b(W), \phi_{j,r,k}^{block}(i)|\phi_{j,r,k}^{block}(i)=0) \times 2^{|N|-r}}
\]  

(25.23)

\[
ITC(i, k) = \frac{|N|}{\sum_{r=1}^{k_t} \sum_{j=1}^{|N|} SSP_{\text{cost}}(G_k, s, d, b(W), \phi_{j,r,k}^{cost}(i)|\phi_{j,r,k}^{cost}(i)=0)} \times 2^{|N|-|SM_{j,r,k}|}-r
\]  

(25.24)

Similarly to the brute force method, the enhanced recursion starts in the trial \((G_k, s, d, b(W), \psi[j] = 1)_{j=1..N}\). The recursion stops when the following two cases are encountered:

1. The connection request is blocked in all instances of a level. According to Lemma 1, having more “0” when going to the next level can never help getting a feasible solution.
2. The solution consists of a single segment. It means that only one solution can possibly be derived no matter how we alter the \(\psi\) array. If the solution is derived in the root, it is the exact solution for the SPP problem, as well.

When the recursion stops, we can claim that all the possible solutions for a specific connection request have been derived, even if the recursion is called only for nodes of \(SM_{j,r,k}\) for each solution; and the proposed two performance metrics can also be evaluated with Eq. (2.5.23) and Eq. (2.5.24). The following is the pseudo code for the enhanced recursion.

1. Set \(\psi[j] = 1\) for all \(j\)
2. \(r := 0, \text{last}_\text{cost} := \text{undefined}, \text{last}_\text{node} := \text{undefined}\)
3. \(\text{set}_\text{of}_\text{already}_\text{examined}_\text{psi} = \text{empty}\)
4. \(\text{Analyze}((\text{trial}(G, s, d, b(W), \psi), \text{integer} r, \text{int last}_\text{cost}, \text{node last}_\text{node}, \text{set}_\text{of}_\text{already}_\text{examined}_\text{psi})
5. \(\text{if} (\psi \text{ is in the set}_\text{of}_\text{already}_\text{examined}_\text{psi}) \)

load(SSP_blocked,cost, |SM|) results stored with ψ
else
    // derive the SSP of the trial
    int cost=SSP_cost(G,s,d,b(ψ),ψ)
    int SM= switching or merging nodes of the result of SSP
    boolean SSP_blocked := SSP_block(G,s,d,b(ψ),ψ) // see Eq(2.5.19)
    put (ψ; (SSP_blocked,cost, |SM|)) in _set_of_already_examined
    if SSP_blocked
        for all x in SM
            ψ[x] := "0"
        Analyze(trial(G,s,d,b(ψ'),r+1,cost,x,set_of_already_examined_ψ)
    // record the obtained statistics
    if (there was last_node)
        if SSP_blocked
            IBP(last_node)=IBP(last_node)+1/2^r
        else
            ITC(last_node)=ITC(last_node)+(cost–last_cost)/2^|SM|+r

    All results obtained by the SSP algorithm are saved with a triplet (a Boolean variable whether it is succeeded or blocked, the cost of the result, the number of switching and merging nodes in the result). A set structure called _set_of_already_examined_ψ (with the standard template library of C++) is used to assign a triplet to a trial. A red-black tree is formulated in the searching process (with a complexity O(log N)) to check whether a specific array ψ has already been in the set structure in order to reduce the redundant computation.

In the above, a variable r represents the depth of the recursion, what is equal to the number of "0" elements in the array ψ of the trial. If the SSP algorithm fails (while its (last_node)-counterpart succeed), the number of “don’t care” trials corresponding to the blocked array ψ is added to IBP(last_node, k). Since the number of “don’t care” trials according to Eq. (2.5.23) is 2^|N|–r, we need to add 2^|N|–r/2^|N| = 1/2^r to IBP(last_node, k). For the average cost increase, it is done in a similar way. According to Eq. (2.5.24) there are 2^|N|+|SM|–r “don’t care” trials, where |SM| is the number of S/M nodes in the solution of the trial (since there are r S/M incapable nodes and |SM| S/M capable nodes and all the other are “don’t care” nodes). Thus we need to increment ITC(last_node, k) with (cost_increase) ∗ 2^|N|+|SM|–r/2^|SM| = (cost_increase)/2^|SM|+r = (cost–last_cost)/2^|SM|+r, where the cost_increase is the difference between the cost of the trial and its (last_node)-counterpart.

2.5.6 Simulation Results

Experiments are conducted to verify the ILP formulation using CPLEX 8.0 on Sun Ultra 80 workstation with 2GB memory and several Linux workstations. To verify the impact of having different assignments of SM capability of each node, the simulation is conducted on two realistic network topologies. The first one is the pan-European fiber-optic network resulted by IST project LION and COST action 266 as [65]. It has 28 nodes and 57 bi-directional links as shown on Fig. 2.8. The second one is based on the US NSF Network [66] with 26 nodes and 43 bi-directional links as shown in Fig. 2.9. For both networks a traffic matrix in year 2005 is estimated according to [38], which is a slightly improved model than that provided in [30]. A dynamic traffic pattern is generated according to the traffic matrix such that an Interrupted
Poisson Process and Pareto inter-arrival times [37] are integrated together with exponential holding time.

Fig. 2.10. and Fig. 2.11. show the simulation results of the European Reference Network with 726 connection requests and of the North-American Reference Network with 1432 connection requests. The networks are lightly loaded with 5-10% link utilization. In NARNet, the recursion function invokes the SSP algorithm for 61120 times and 3265 for ERNet. In order to speed up the solving process such that more connection requests with different states of $\psi$ array can be examined, each connection arrival is first analyzed with the above mentioned recursive function and then is routed with the proposed SSP algorithm assuming that all nodes are S/M capable without any delay constraint.

For the ERNet, the S/M capability for restoration traffic in Paris, Lyon, Amsterdam, Berlin, Rome and London are crucial for reducing the average blocking probability. Removing the S/M capability of each of these nodes would increase the blocking probability of the network by 1-1.5% in average. It is notable that these nodes are part of the main cuts of the network. For example, London and Amsterdam switch all traffic between the European Continent and Great Britain, and removing S/M capability of these two nodes increases the blocking probability by 2.5%. In terms of cost increase, London, Berlin, Amsterdam, Paris, Munich Frankfurt and Rome are the crucial nodes. Removing S/M capability of any of these nodes would increase the average (weighted) capacity by 0.5-1%. It is notable that these nodes are in the center of the network with a high nodal degree, and are almost identical with the crucial nodes in the case for blocking probability except Munich and Frankfurt, which are in the center of Europe but not a significant part of any cut separating the network. From the experimental results we find that equipping nodes with S/M capability in the edge of the network is not effective, while it is much more crucial to equip the nodes in the center of the network with a higher nodal degree than the others for both cases of blocking probability and cost.

![European Reference Network (ERNet)](image)

Figure 2.8: European Reference Network (ERNet)
Performance Impact by the Size of Protection Domains

Simulation is arranged as follows to investigate the impact by addressing the restoration constraint on the connection requests. Without loss of generality, unit delay is assigned to each link in the simulation, which yields a fact that \( \zeta_j = 1 \) for all link \( j \). Therefore, the restoration constraint parameter \( s_{max} \) is nothing but the hop count of the working and protection segment-pair of each protection domain. For each connection request, the ILP is solved with different values of \( s_{max} \) (i.e., the restoration time constraint). Fig. 2.12 gives a flowchart for the simulation process. For each connection request, a series of SSP problems is solved with different values of \( s_{max} \), where two performance metrics are evaluated: one is the number of successful sets-up and the other is the average cost using the target function of the ILP.

Fig. 2.13 shows some illustrative examples, where simulation is conducted on the network N16 (with 16 node and 27 bi-directional links) with heavy load (at average of 73% total network utilization). The graphs with the selected \( s - d \) pairs are drawn on the right-bottom corner of the chart. The \( y \) axis represents the cost of the connection yielded by the target function of the ILP. It is clear that as the restoration time constraint is getting more relaxed, the performance impairment (in terms of the cost) for each connection request is reduced.

As a comparison, the optimal SPP is evaluated using the ILP formulation of Section 1.4.1, where the derived result for each connection is marked by triangles on the charts. Note that the cost of solving the SPP is not smaller than that of the SSP case since SPP is a special case of SSP (with \( k_{max} = 1 \)). Shared Link Protection (SLP) is also a special case of SSP where each working segment consists of only one link. The ILP for SLP is derived by adding an additional constraint Eq. (2.5.25) to the ILP of SSP:

\[
\sum_{j \in \alpha_k} x_{j}^{k} \leq 1 \quad \text{for} \quad 1 \leq k \leq k_{max}
\]  

(2.5.25)
Figure 2.10: The average increase of blocking probability in case of removing the S/M capability of the corresponding node. Each value is multiplied with $10^4$. 
Figure 2.11: The average increase of cost in case of removing the S/M capability of the corresponding node. Each value is multiplied with $10^3$ (they are in $\%$).
Figure 2.12: The simulation process for the performance impact by addressing a restoration time constraint.

A detailed overview on SLP and SPP was proposed in [2]. The simulation results of SLP are marked with boxes on the charts, in which the length limitation on the protection path can be addressed using Eq. (2.5.13). The selected node-pairs are selected far from each other such that it illustrates the increase of the cost of SSP if we gradually sharpen the restoration time constraint.

Figure 2.13: An example of the restoration time versus cost.

Fig. 2.14 shows the results using a 61-node network (shown in Fig. 1.12(c)) with a light traffic load for comparing the three types of protection in terms of average cost and the success rate under different restoration time constraints. Results of 100 random connection requests are averaged for each data.

It is shown that an average of approximately 10% reduction in the cost can be achieved with SSP over the case of SPP if the restoration time constraint is relaxed to 13 hops. The average cost in the SSP and SLP cases increase when the restoration time constraint is sharpened, as it was expected. However, the average cost drops dramatically when the restoration time constraint is very tight since at this moment most of the long connections (with large cost) are blocked, and only the short connections (with small cost) can be allocated. The results may serve as important basis in the effort of setting up the pricing policy for connections with different lengths and restoration time requirements.

It can also be observed that SSP can yield much higher success rate for those connection requests under a tight restoration time constraint than that with SPP at the expense of taking extra cost, as shown in Fig. 2.14. SLP yields the highest average cost in all cases with a close success rate with SSP, and is seen less competitive with the other two types of protection.
We further extend the simulation study on ERNet and NARNet with the same traffic pattern and simulation environment. In order to solve a large number of connection requests, we set $\xi_{\text{max}} := 0.1 \cdot c_{\text{pre}}(W)$ at function blocks (12) and (14) in the flowchart of Fig. 2.7, and the maximum number of segments are set to $k_{\text{max}} = |\text{feasible\_segments}| + 2$ at function block (16) in the flowchart. The above two steps can significantly reduce the runtime of solving the ILP for each connection at the expense of taking a little bit higher cost as well as having a larger blocking probability when the restoration time constraint is tight.

![Graph](image1)

Figure 2.14: Performance impairment by addressing the restoration time constraint using the 61-node network at light load (19%).

The simulation results of NARNet with the high traffic load are shown in Fig. 2.15, where the advantage demonstrated in using SSP is that the success rate outperforms that of the SPP case by 2 times or more at the expense of taking a little bit higher cost (as shown in Fig. 2.15(b)). For SLP, the overall performance is not comparable to the other two cases, although it can guarantee the shortest restoration time. In short, SSP gives the best compromise in terms of cost versus restoration time, while SLP requires an average of 10-20% additional capacity allocation compared to SSP or SPP. As it can be seen, the performance of SSP is not significantly degraded with the precalculation.

![Graph](image2)

Figure 2.15: Performance impairment by addressing the restoration time constraint with NARNet at high load.
Figure 2.16: Runtime in solving the SPP, SLP with pre-calculation, SLP, SSP with pre-calculation, and SSP on N16 and ERNet.

Figure 2.17: Average cost and success rate of SSP-optimal and SSP with pre-calculation step. For SLP there is an average of 3.3 gap.

2.5.7 Conclusion

This section studies dynamic survivable routing for segment shared protection (SSP), in which an ILP is formulated such that the S/M node-pairs in each W-P segment and the corresponding least-cost W-P segment-pair for a connection request can be jointly determined in a single step. A novel approach called arc reduction pre-calculation is devised to initiate a graceful compromise between the optimality and the computation time for solving the ILP. Different from the ILP of previous section, this formulation considers the SM capability of each node to serve as an S/M node for each W-P segment during the routing process, which meets the practical requirement of network control with limitation on the hardware and signaling precision. In particular, the performance impact by equipping/removing S/M capability for a specific node is investigated. For this purpose, two performance metrics are defined, called Average Increase in Blocking Probability (AIBP) and Average Increase in Total Cost (AITC), which are jointly determined by the network topology, traffic pattern, and the survivable routing algorithm adopted. To evaluate the two performance metrics, both a brute force method and an enhanced recursive method are introduced. With the former, the SSP algorithm is solved for every possible state of S/M capability distribution; while in the latter, any redundant calculation can be avoided by using a novel data structure in the recursion. Extensive simulation efforts are conducted on NARNet and ERNet based on the estimated traffic pattern in year 2005 to investigate the benefits of equipping each node to be capable of switching and merging
restoration traffic. The simulation results show that equipping only a small number of selective nodes with SM capability can be solidly beneficial to the network performance. We conclude that the proposed two metrics can effectively define the impact by equipping/removing S/M capability of a specific node. The enhanced recursive method can efficiently evaluate the two metrics without any redundant effort in solving the SSP algorithm. The study also investigates the performance impairment when different restoration time constraints are addressed (or having different sizes of protection domain). A comparison is made among Shared Link Protection (SLP), Shared Path Protection (SPP) and Segment Shared Protection (SSP) in terms of average cost and success rate of setting up connections. We observe that SSP can initiate a graceful compromise between average cost and network throughput under a wide range of restoration time constraints. For SLP, the overall performance is not distinguished compared with the other two types of protection although an ultra-fast restoration process can be guaranteed. With the result and analysis methodology, the modeling of the whole survivable routing process can further facilitate the deployment, and dimensioning of the network switching capacity, and can serve as a reference for setting up the pricing policy.
2.6 Claim 2.3: Complexity Analysis of Shared Segment Protection Problem

Finding two SRG disjoint paths between $s$ and $d$ was proved to be NP-complete in [67]. Most of the papers focus on the case that each fiber in the network topology is an SRG. This simplification comes from the assumption that the probability of each physical conduit to be subject to a failure is independent. In this case the sets of SRGs are the links of the network. In this section we will prove the NP-hardness of the problem in this special case.

2.6.1 Problem Formulation

The decision problem of Shared Segment Protection (SSP) in Mesh Telecommunication networks defined as follows:

Given:

- a undirected graph $G(V, A)$, with $V$ and $A$ being the set of vertices and arcs, respectively,
- the free capacity ($f$) and the spare capacity ($v$) of each arc,
- the SRGs of $G$ (the arcs of $G$ in this case),
- the $S$ matrix.
- the source node $s$, the destination $d$ and the bandwidth $b(W) = 1$ of the new demand,
- parameter $k_{\text{max}} (\geq 1)$ that is an upper bound on the number of segments.

Decide whether there exists:

- a working path $W$, with the switching and merging nodes of each segment along $W$ (represented in matrix $\mathcal{W}$), and the protection path of each segment (represented in matrix $\mathcal{P}$), such that
  - the number of segments (or the number of protection domains) along $W$ is $\leq k_{\text{max}}$,
  - $W_k$ and $P_k$ should be arc (SRG) disjoint for $\forall k$,
  - the feasible condition of the working path is $f_i \geq b(W)$ for $\forall i \in W$,
  - the feasible condition of the protection path is $f_i + v_i \geq b(W) + \max_{\forall j \in W_k} s_{i,j}$ for $\forall i \in P_k$ and $\forall k$.

In the optimization version of SSP a cost function $c$ assigned to each edge is given as well, representing the cost of allocating one unit of capacity. The task is to find the connection with minimal total capacity allocation.

Note that Shared Path Protection (SPP) is a special case of the above defined problem, where $k_{\text{max}} = 1$. This proof is a correction and generalization of the proof of SPP published in [67].

2.6.2 The proof of NP-completeness of SSP

Obviously, this problem belongs to the class NP. We reduce the 3SAT to this one. The 3SAT has variables $x_1, x_2, \ldots, x_n$, with each variable $x_i$ giving rise to literals $x_i$ and $\overline{x_i}$ (where $\overline{x_i}$ indicates the negation of literal $x_i$). A 3SAT Boolean expression consists of the conjunction of a set of clauses $C_1 \wedge C_2 \wedge \cdots \wedge C_3$, where each clause is a disjunction of 3 literals, e.g., $C_1 = \{x_1 \lor x_2 \lor x_3\}$. The truth assignment is a function $\tau : \{x\} \rightarrow \{\text{TRUE, FALSE}\}$. The instance asks if there is a truth assignment that satisfies the 3SAT expression. 3SAT is known to be NP-complete.
Given an instance of 3SAT we derive a network topology as illustrated in Fig. 2.18. To each variable $x_i$ we associate two edges between nodes $u_i-1$ and $u_i$. One of the edges is labeled $x_i$ and one labeled $\bar{x}_i$. To each clause $C_i$ we associate three edges between nodes $l_i-1$ and $l_i$. The edges are labeled $C_i : x_a, C_i : x_b, C_i : x_c$. A set of already provisioned connections is defined: for each clause $C_i$, there are three connections so that each working path has one edge $x_a$, and the shared protection path has three edges, the middle one is $C_i : x_a$ (in this example $P_1 = \{u_{a-1}, \bar{x}_a, u_{a-1}l_{i-1}, C_i : x_a, l_iu_a\}$ and $P_2 = \{u_{b-1}, \bar{x}_b, u_{b-1}l_{i-1}, C_i : x_b, l_iu_b\}$ and $P_3 = \{u_{c-1}, \bar{x}_c, u_{c-1}l_{i-1}, C_i : x_c, l_iu_c\}$). The capacity on the links are set so that only edges on the path $su_0 \ldots u_i \ldots u_n$ have (one unit of) free capacity so the working path can use the upper part of the graph (drawn with broken line on Fig. 2.18). Some other connections are defined such that their working segment uses $su_0$, which is a common link with the first working segment of the new demand. The protection paths of these connections should use all the edges between nodes $l_i$ and $u_j$ (the dotted lines on Fig. 2.18), such that the sharable spare capacity for the connection between $s$ and $t$ is set to zero on dotted edges. As consequence the protection path can use some of the edges between nodes $s, l_1, \ldots, l_i, \ldots, l_m, d$ (drawn with the solid line on Fig. 2.18). The capacity of the edges $sl_0$ and $l_m d$ should be set so that they have zero free capacity and one unit of sharable shared capacity. Variable $k_{max}$ can be set to any value ($\geq 1$).

![Figure 2.18: The network corresponding to \((x_1 \lor \bar{x}_2 \lor \bar{x}_3) \land (\bar{x}_1 \lor x_2 \lor \bar{x}_3)\). The capacity of edges $su_0$ is 10, $u_0 l_0$ is 6, $sl_0$ is 5, $u_0 l_1$ is 3, $u_1 l_0, u_2 l_2, u_3 l_2, u_3 d, u_0 \bar{x}_1 u_1, u_0 \bar{x}_1 u_1, u_1 x_2 u_2, u_1 \bar{x}_2 u_2, u_2 \bar{x}_3 u_3, u_2 \bar{x}_3 u_3$ are 2 and all the others are 1. Twelve connection are already provisioned, $P_1 = \{u_0 \bar{x}_1 u_1; u_0 l_0 C_i : x_1, l_1 u_1\}, P_2 = \{u_1 x_2 u_2; u_1 l_0 C_i : x_1, l_2 u_2\}, P_3 = \{u_2 x_3 u_3; u_2 l_0 C_i : x_1, l_3 u_3\}, P_4 = \{u_0 \bar{x}_1 u_1; u_0 l_1 C_i : x_2, l_2 u_1\}, P_5 = u_1 \bar{x}_2 u_2; u_1 l_1 C_i : x_2, l_2 u_2\}, P_6 = \{u_2 \bar{x}_3 u_3; u_2 l_1 C_i : x_2, l_3 u_3\}$, $P_7 = \{u_0 s; sl_0 u_0 l_0 u_0 l_0\}, P_8 = \{u_0 s; sl_0 u_0 l_0 u_0 l_0\}, P_{10} = \{u_0 s; sl_0 u_0 l_0 u_0 l_0\}, P_{11} = \{u_0 s; sl_0 u_0 l_0 u_0 l_0\}, P_{12} = \{u_3 l_2; u_3 d l_2\}$. The task is to establish a connection with one unit of bandwidth between $s$ and $d$. As the free (unreserved) capacity on edges with dashed line is 1, while on the others it is 0, the working path passes through $su_0 u_1 u_2 u_3 d$. The sharable spare capacity for protection path is zero on edges with dotted lines and one on edges drawn with solid line. Thus there is only one segment and the protection path passes through $sl_0 l_1 l_2 d$.]

It is easy to see that $G$ can be constructed in polynomial time. Therefore it is sufficient to show that there exists a truth assignment that simultaneously satisfies all the $m$ clauses if and
only if there exist a working path and a shared segment protection path between $s$ and $d$.

⇒

If there exists a truth assignment $\tau$ that simultaneously satisfies all $m$ clauses, then path $W$ passes through the edge $x_i$ if $\tau(x_i) = \text{FALSE}$ and passes through $\overline{x}_i$ if $\tau(x_i) = \text{TRUE}$. Each satisfied clause $C_j$ contains either a literal $x_i$ such that $\tau(x_i) = \text{TRUE}$ or literal $\overline{x}_i$ such that $\tau(x_i) = \text{FALSE}$, which implies that there exists a protection path on edges $C_i : x_a$.

⇐

Conversely, suppose there exist a working path and a protection path for each segment.

Only the edges drawn with broken lines have free capacity $(su_0 \ldots u_i \ldots u_n d)$ for the working path. We set $\tau(x_i) = \text{TRUE}$ if $W$ passes through the edge of $x_i$ and $\tau(x_i) = \text{FALSE}$ otherwise. The first segment of the working path will use edge $su_0$, so the protection path can take only edges drawn with solid line, as the edges of dotted lines have no sharable capacity. Thus the route of the protection path is $sl_0 \ldots l_i \ldots l_m d$ and there is only one segment. According to this truth assignment, clause $C_j$ is satisfied since $P$ can take edge $C_i : x_a$ only if their working segments are arc disjoint so the working segment passes through $x_a$, which means $\tau(x_a) = \text{TRUE}$. Therefore all $m$ clauses are satisfied.

\[\square\]

Note that this proof is valid both in the vertex and in the arc disjoint cases. The proof is valid if $k_{max} = 1$, so it is a proof of the NP-hardness of SPP as well.
Chapter 3

Distributed Control Architecture for Shared Path Protection in Mesh Communication Networks with Bandwidth Guaranteed Tunnels
This chapter proposes a novel distributed control architecture for shared protection with reduced complete routing information, which aims to initiate a graceful compromise between the amount of link-state dissemination and the performance impairment due to the incompleteness of routing information. We will first give explicit and comprehensible descriptions on a number of reported routing information dissemination scenarios for shared protection. Then a novel framework of link-state dissemination for facilitating shared protection, called Sharing with Reduced Information (SRI), will be introduced, in which the Singular Value Decomposition (SVD) transformation is adopted to deal with the information reduction. To implement the survivable routing for each scheme, a novel Integer Linear Programming (ILP) formulation is given, which is characterized by the ability of handling each scheme with a uniform formulation. We will show through simulation that using SRI can achieve a higher throughput and a better estimation in reconstruction of the spare provision matrix than the other schemes with the same complexity of link-state dissemination.

3.1 Introduction

Studies on the dynamic survivable routing problem for shared protection have been extensively reported in the past several years. Most of them focused on the schemes under the complete routing information scenario [4, 5, 2, 3] (also called Sharing with Complete Information, SCI [48]). With SCI, a survivable routing algorithm calculates the least-cost working and shared protection path-pair provided with per-flow information such that the maximum degree of sharing of spare capacity is achieved. However, in the practical operation of the Internet with a distributed control environment, the scenario may be subject to significant overhead in terms of link-state dissemination and yields a serious scalability problem. Therefore, some studies turned to solving the problem in a Scenario of Partial Routing Information (SPI) [68, 49, 50, 48, 27], in which a survivable routing protocol does not require per-flow information to make a routing decision such that both the amount of dissemination for enabling distributed control and the complexity of the routing process can be reduced at the expense of lower degree of sharing [68, 49, 48].

In [68], a scheme called Reduced Complete Information (RCI) is proposed, which aims to estimate the link cost of a protection path according to the shareability of each link. Here shareability of a link is defined as the number of links of working paths protected by the spare capacity of the link. The maximum extent of resource sharing is explored after the physical route of the protection path is determined. In [27], a common factor \( \alpha \) is adopted to estimate the impact induced by resource sharing in determining the working and protection path-pair. The study in [49] proposed a more advanced scheme than SPI, called Distributed Partial Information Management with Sufficient and Aggregated Information (DPIM-SAM), that can improve the scalability of link-state dissemination in SCI and the degree of sharing in SPI.

This chapter tackles the problem of shared protection under a distributed control environment. We first give a comprehensive discussion on the signalling and link-state dissemination mechanisms for the most recently reported proposals of the related topic, including Sharing with Complete Routing Information (SCI) [48], Distributed Partial Information Management with Minimum Bandwidth Allocation (DPIM-M-A) [49], Sharing with Partial Information (SPI)
[48], and Distributed Partial Information Management with Sufficient and Aggregated Information (DPIM-SAM) [49]. To improve the degree of sharing without increasing the complexity of link-state dissemination, a novel scheme called Sharing with Reduced Information scenario (SRI) is proposed. SRI is based on Singular Value Decomposition (SVD) transformation which extracts the most informative data from the conventional network state. Compared with DPIM-SAM, SRI requires the same dissemination and signalling complexity while yielding a better estimation on the dependency between working and spare capacity of each pair of links. An ILP formulation is given for SCI, DPIM-SAM and SRI to solve the problem of diverse routing for shared protection. We will verify the proposed scheme through extensive simulation and prove that SRI will cause the least amount of offset in reconstructing the dependency matrix (or called spare provision matrix defined in Section 1.2.8) compared to the other partial information schemes.

3.1.1 Link-State Dissemination and Spare Provision Matrix Reconstruction

In Section 1.2.8 we discussed the effort of exploring the maximum extent of resource sharing as \( W \) is given in advance. This section is committed to investigating more general situations in which a distributed control environment is considered. The following design issues will be emphasized: what information should be disseminated based on the OSPF framework and how an arbitrary ingress node reconstructs the matrix concerning the dependency of working and spare capacity between each pair of links, etc.

The following four scenarios are considered and discussed in terms of link-state dissemination and spare provision matrix reconstruction: (1) Sharing with Complete Routing Information (SCI) [48], (2) Distributed Partial Information Management with Minimum Bandwidth Allocation (DPIM-M-A) [49], (3) Sharing with Partial Information (SPI) [48], and (3) Distributed Partial Information Management with Sufficient and Aggregated Information (DPIM-SAM) [49]. With different routing information scenarios, the spare provisioning matrix \( S \) (see also Fig. 3.1) is determined through different approaches. The algorithm running at ingress nodes for explicit survivable routing is required to reconstruct \( S \) if the link-state received is not complete.

![Figure 3.1: An illustration of the spare provision matrix \( S \)](image-url)
Sharing with Complete Information (SCI)

Let us define $D_l$ as the set of those connections, whose working paths are involved in the $l^{th}$ $BSS$ of the network. Thus $D = \bigcup_{l \in L} D_l$. It is clear that the calculation defined in Section 1.2.8 is achievable if the ingress node has full knowledge of per-flow information to acquire the parameters $a_{q,l}$ and $b_{q,j}$, $\forall j \in L$, $\forall l \in W$, $q \in D_W = \bigcup_{l \in W} D_l$ in Eq. (1.2.5), in order to investigate the dependence of working and spare capacities between each pair of links. Most of the telecommunication networks are protected against cable cut and maybe node failure as well, thus $BSS$, as the basic set of SRGs, are the physical links of the network. To cooperate with the OSPF framework, the node corresponding to the $l^{th}$ physical link (or $BSS$) calculates the vector $\bar{S}^l$ for dissemination, which is nothing but a column-vector derived by using the equation

$$\bar{S}^l = ((B^l)^T \cdot M^l \cdot A^l) \odot G^W$$  \hspace{1cm} (3.1.1)$$

where $A^l$ is the working path-$BSS$ incidence matrix and $B^l$ is the backup path-link incidence matrix concerning all the connections with their working paths passing through the $l^{th}$ $BSS$; and $M^l$ is a $|D_l| \times |D_l|$ diagonal matrix concerning the bandwidth of the connections passing through any link assigned to the $l^{th}$ $BSS$. In [1] $A^l$ and $B^l$ is described in details. Note that $((B^l)^T \cdot M^l \cdot A^l) \odot G^W$ yields a $|L| \times |BSS|$ matrix (see also Fig. 3.2).

Let us define column vector $S^l$ as the $l^{th}$ column of $\bar{S}$

$$S^l = S \cdot e_l$$  \hspace{1cm} (3.1.2)$$

Using Eq. (1.2.5) it is easy to verify that $S^l = S \cdot e_l$ as well. Thus, at any remote ingress node, the receiving of $S^l$ from all the other nodes will enable the reconstruction of the matrix $S$ for the whole network. This job can be done by simply putting all the $S^l$ column-vectors together, for all $l \in BSS$. With the knowledge of $S$, the ingress node will have sufficient information to perform explicit survivable routing with complete routing information, in which resource sharing along each link can be completely explored according Eq. (1.2.8) once $W$ is given. A cost function can be designed based on specific purposes such that the least-cost shared protection path is derived, as it was described in Section 1.2.7.
The maximum extent of resource sharing is achieved at the expense of the fact that a node must have full knowledge of per-flow information of its corresponding link $l$ (i.e., $\mathcal{A}_l^i$ and $\mathcal{B}_l^i$) along with the computation effort addressed by Eq. (1.2.5). The node can derive $\mathcal{A}_l^i$ and $\mathcal{B}_l^i$ when those working paths defined in $\mathcal{A}_l^i$ are set up, where the signaling messages are sent back-and-forward along the working and protection paths. In addition, the link-state dissemination for vector $S_l^i$ is issued by each node flooding the networks, which yields a complexity in the amount of dissemination $O(|L| \cdot |BSS|) \sim O(|L|^2)$, leaving the SCI scheme not feasible in a distributed control environment. Fig. 3.3(a) illustrates the link-state dissemination. The corresponding node of $l$ will calculate $S_l^i$ according to local information for dissemination. To overcome this scalability problem, some studies proposed approaches for reducing the amount of dissemination without losing much performance/degree of sharing, which will be discussed in the following paragraphs.

**Distributed Partial Information Management with Minimum Bandwidth Allocation (DPITM-M-A)**

In this case no information about dependency of working and spare capacity between each pair of links will be disseminated. The physical routes of the working and protection paths will be determined first. In this case, a very fast algorithm (e.g., Surballe’s algorithm or Two-Step-Approach) can be adopted on the residual graph with all the links having insufficient bandwidth being excluded to solve the disjointly-routed problem path-pair according hop count. After the physical route of the protection path is determined, each node along the protection path considers the spare capacity sharing for its corresponding link. This is also referred to as *post-exploration of sharing*. With DPITM-M-A, each node is assumed to have per-flow information such that the maximum degree of sharing can be achieved along each link at the post-exploration stage. Fig. 3.3(b) illustrates link-state dissemination in this scheme. With DPITM-M-A, 100% feasibility can be guaranteed for the derived solution if the routing algorithm considers those links with $f_j \geq b(W)$ when $P$ is being determined.

**Sharing with Partial Information (SPI)**

In this scenario, sharing is partially allowed during the routing stage in the sense that the survivable routing algorithm performed at an ingress node has insufficient information to explore all the sharable spare capacity along each link. In this scenario a node does not have the knowledge of the parameters $b_{j,q}$ defined in Eq. (1.2.6), where each Q passes through the $i$th BSS; therefore, it simply sets $b_{j,q} = 1$ for all links $j$ and working path $Q$. In other words, the node assumes that the working paths passing through the corresponding link $l$ have all their protection paths passing $j$ for $j \in L$, which leads to a fact that $s_{j,l}$ is over-estimated for all $j \in L$ and $l \in BSS$ at any remote ingress. By over-estimating $s_{j,l}$, however, the feasibility of the routing solution any remote ingress node is guaranteed. Fig. 3.3(c) illustrates the SPI scheme, where the corresponding node simply sends $q_l$ to the remote ingress since $s_{j,l} \leq q_l$ for $\forall j \in L$.

With the partial information required in the survivable routing protocol, no dissemination concerning the dependency of working and spare capacities between each pair of links needs to be disseminated. This great advantage is, however, at the expense of only partially explored
resource sharing.

**Distributed Partial Information Management with Sufficient and Aggregated Information (DPIM-SAM)**

DPIM-SAM serves as a framework of link-state dissemination in-between the SCI and SPI scenarios in terms of computational complexity and the amount of link-state dissemination. The basic idea of link-state dissemination in DPIM-SAM is that each node exchanges the maximum element in $\mathbf{s}_l^i$ instead of the whole $\mathbf{s}_l^i$ vector as done in SCI. Therefore, the amount of dissemination can be greatly decreased and the complexity of this information exchanging mechanism becomes $O(|\mathcal{BSS}|)$. This can also guarantee the feasibility of the routing result at any remote ingress by over-estimating $s_{j,l}$.

Compared with SCI, except for the reduction in link-state dissemination, DPIM-SAM requires the same complexity in the propagation of per-flow information when a connection is set up and the same effort in calculating $\mathbf{S}_l$ at each corresponding node of a link. Fig. 3.3(d) illustrates the case.

### 3.2 Claim 3.1: Reduced Information Scenario for Shared Path Protection

We present in this section a new scheme for implementing the shared protection in a distributed control environment, called *Sharing with Reduced Information (SRI)*, which is an enhancement of the DPIM-SAM scheme. Compared with DPIM-SAM, in SRI each node disseminates additional two scalars for its corresponding link to further improve the accuracy in the reconstruction of the sparse provision matrix $\mathbf{S}_l$ at any remote ingress.

The task is to find a feasible SRG-disjoint working and protection path-pair such that the total cost (Eq. 1.2.24) is minimal:

\[
\text{Minimize: } b(W) \cdot \left( \| \mathbf{C}_l^T \cdot \mathbf{W}_l \| + \| \mathbf{C}_l^T \cdot \mathbf{D}_l^T \cdot \mathbf{R}_l \cdot \mathbf{W}_l' \| \right) \tag{3.2.1}
\]

Note that, the conversion between $\mathbf{R}_l$, $\mathbf{S}_l$, and $\mathbf{H}_l$ can be easily done by Eq. (1.2.15) and Eq. (1.2.14). Our task is to find a feasible path-pair such that the total cost is minimal.

An additional working variable $s_{j,w}^u$ is introduced for each link and another one $s_{j,p}^p$ for each BSS, which are handled by the corresponding node of the link or BSS respectively, and disseminated to all the other ingress nodes to facilitate the reconstruction of $\mathbf{R}_l$ at each ingress node. We will describe in detail how these variables function under the distributed control environment with incomplete routing information. Based on the framework, a novel survivable routing algorithm will be presented to solve a SRG-disjoint working and shared protection path-pair.

The basic idea of our scheme is to use SVD transformation to estimate the reconstruction of $\mathbf{R}_l$ in each ingress node. The SVD transformation can be performed with polynomial time complexity [21] and is shortly stated as below. Let $\mathbf{R}$ be a real $m$-by-$m$ matrix. There exist orthogonal matrices $\mathbf{U} = [U_1, \ldots, U_m] \in \mathbb{R}^{m \times m}$ and $\mathbf{V} = [V_1, \ldots, V_m] \in \mathbb{R}^{m \times m}$ such that

\[
\mathbf{U}^T \cdot \mathbf{R} \cdot \mathbf{V} = \text{diag}(\sigma_1, \ldots, \sigma_r) \in \mathbb{R}^{m \times m}, \text{ where the eigenvalues of } \mathbf{R} \text{ are given as } \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r.
\]
Figure 3.3: An illustration of the amount of dissemination for the four schemes discussed in this section: (a) SCI, which achieves the maximum extent of resource sharing (b) DPIM-M-A, which does not disseminate any dependency information while performing post-exploration of sharing after the physical route of the protection path is determined; (c) SPI, which disseminates $q_i$; and (d) DPIM-SAM, which disseminates $\max_{j \in L} s_{i,j}$.

$\sigma_r \geq 0$ and $\text{rank } (R) = r$. It is also proved that given any $R' = \sigma_1 U_1 V_1^T$, where $\text{rank } (R') = 1$, then

$$\min_{\forall R, \text{rank } (R) = 1} \| R - R' \|_2 = \| R - R' \|_2 = \sigma_2 \quad (3.2.2)$$

However, each node in the network is assumed to know $R'$ instead of $R$, which yields a fact that we need to take the SVD transformation on $R'$ to approximate the effect of using the matrix $R$. Thus, two column-vectors $U_1$ and $V_1$ can be derived at the corresponding node of $l$ such that the term $\sigma_1 \cdot U_1 \cdot V_1^T$ yields the best approximation of $R'$ out of all possible column vector pairs. Let us denote $\Sigma^p = \sqrt{\sigma_1} \cdot U_1$, and $(\Sigma^w)^T = \sqrt{\sigma_1} \cdot V_1^T$, which are the two column-vectors. The node in charge of the $l$th link will take the $l$th entry of each vector as part of the link-state dissemination, i.e., $s_1^p$ and $s_1^w$ in $\Sigma^p$ and $(\Sigma^w)^T$, respectively. In this case the cost function in Eq. (3.2.1) will be revised as:

$$c_{\text{total}} = b(W) \cdot (\| C^T \cdot W \| + \| C^T \cdot P^T \cdot R \cdot W' \|) \approx b(W) \cdot (\| C^T \cdot W' \| + \| C^T \cdot P^T \cdot s^p \cdot (\Sigma^w)^T \cdot W' \|) \quad (3.2.3)$$

where $P^T \cdot s^p$ is a column-vector and $(\Sigma^w)^T \cdot W'$ is a row-vector. Thus, Eq. (3.2.3) can be expressed as:

$$c_{\text{total}} \approx b(W) \cdot (\| C^T \cdot W' \| + \| C^T \cdot P^T \cdot s^p \| \cdot \| (\Sigma^w)^T \cdot W' \|) \quad (3.2.4)$$

since the matrix is a diad, its rank is 1, and in this case
$$\|U \cdot V^T\| = \sum_{i=1}^{n} \max_j |u_i \cdot v^j| =$$

$$\sum_{i=1}^{n} \max_j |u_i| \cdot |v^j| =$$

$$\sum_{i=1}^{n} |u_i| \cdot \max_j |v^j| = \|U\| \cdot \|V^T\| \quad (3.25)$$

By Eq. (3.2.4), the cost of the working path is the summation of the cost for each link taken by \( W \) as usually, while the cost of the protection path is:

$$\max_{j \in W} (s^w_j) \cdot \sum_{i \in P} s^p_i \cdot c_i \quad (3.2.6)$$

For the routing task, the cost of each link is determined by three variables: \( s^w_i \) and \( c_i \), for the cost of the protection route, and \( \max_{j \in W} (s^w_j) \cdot s^p_i \cdot c_j \) for the cost of the protection route. Thus, we have formulated the problem as a diverse routing problem with different cost functions for working and protection paths, in which each working and protection path takes independent cost function \( c_j \) and \( \max_{j \in W} (s^w_j) \cdot s^p_i \cdot c_j \). Note that the term \( \max_{j \in W} (s^w_j) \cdot s^p_i \) is a scalar possibly with different values at different links.

Although the diverse routing problem with independent and different cost functions for working and protection paths can be solved with some very fast heuristics [69], in this paper, an ILP is formulated to solve the survivable routing problem, which will be introduced in the next section.

### 3.2.1 The Survivable Routing

An ILP is formulated to solve the survivable routing problem for SCI, DPIM-SAM, SPI and SRL. Note that DPIM-M-A is solved with Sunrhalke’s algorithm along with the post-exploitation of spare capacity sharing. The ILP formulation is the same in each scheme; however, the matrix \( S \) is different because an ingress node will estimate/reconstruct the matrix \( S \) in different approaches, which yields a fact that the degree of sharing and throughput are different in each scheme.

Let the ingress and the degree of the connection request be denoted as \( s \) and \( d \). The graph for solving \( W \) is denoted as \( G_w(V, A_w) \), which is composed of the arcs with \( b(W) \leq f_j \). An additional residual graph is defined to facilitate solving \( p \) and is denoted as \( G_p(V, A_p) \). We need this graph to record the spare link-state, which is composed of the links with the amount of free capacity \( f_j \) plus that of the spare capacity \( v_j \) larger than or equal to \( b(W) \) (i.e., \( b(W) \leq f_j + v_j \) for every arc \( j \)). Please refer to Fig. 3.4 for an illustration of the graph transformation.

The objective function for the ILP is as follows:

$$\sum_{(a,b) \in A_w} b(W) \cdot c_{a,b} \cdot x_{a,b} + \sum_{(u,v) \in A_p} (b(W) \cdot c_{a,v} \cdot z_{a,v} + \varepsilon) \cdot y_{a,v} \quad (3.2.7)$$

where \( c_{a,b} \) is the cost of link \( (a, b) \), \( \varepsilon \) is a small constant devised for addressing the cost for \( P \) when it traverses through a link with sufficient sharable spare capacity; \( x_{a,b} \) (with size \( |E_w| \)) and \( y_{a,b} \) (with size \( |E_p| \)) are \( 0-1 \) binary variable indicating flows along the \( W \) and \( P \) on link
(a, b), respectively, which is 1 if the corresponding flow passes (a, b). The concatenation of all the links with \( y_{a,b} = 1 \) yields \( P \), while the concatenation of all links with \( x_{a,b} = 1 \) yields \( W \). \( z_{a,b} \) is a variable for scaling the cost of the protection path taking spare capacity along link (a, b).

\[
\begin{align*}
G_w & \quad \text{Free capacity} \\
G_p & \quad \text{Spare capacity}
\end{align*}
\]

Figure 3.4: Link transformations for the edges in \( G \), which yield the graphs \( G_w \) and \( G_p \) respectively.

The objective function is subject to the following constraints:

\[
\begin{align*}
\sum_{\forall (a,b) \in A_w} x_{a,b} - \sum_{\forall (b,a) \in A_w} x_{b,a} &= \begin{cases} 
1 & \text{if } a \text{ is the source node } s \\
-1 & \text{if } a \text{ is the destination node } d \\
0 & \text{otherwise}
\end{cases} \quad (3.2.8) \\
\sum_{\forall (a,b) \in A_p} y_{a,b} - \sum_{\forall (b,a) \in A_p} y_{b,a} &= \begin{cases} 
1 & \text{if } a \text{ is the source node } s \\
-1 & \text{if } a \text{ is the destination node } d \\
0 & \text{otherwise}
\end{cases} \quad (3.2.9)
\end{align*}
\]

Eq. (3.2.8) and (3.2.9) are flow conservation constraints for the working and protection paths. The disjointness of the working and protection flows can be asserted in Eq. 3.2.10.

\[
\sum_{\forall b \text{ in the same } \text{SRG} \notin Z^W, \text{as } a} y_b \leq n \quad \text{for } \forall a \in A_w
\]

where the value of the constant \( n \) is larger than the maximum number of arcs are in the SRGs, which are not for node failure (e.g., it can be set to \(|V|\)). In the summation all SRGs are considered except the \( \text{SRG}_{\text{null}} \), which failure separates \( s \) and \( d \), since the failure of those SRGs cannot be protected. The constraint upon the variable \( z_e \) is as follows:

\[
x_a + y_e - 1 - \frac{h_{e,b}}{b(W)} \leq z_e \quad \forall e \in A_p, \forall a \in A_w, \forall b \in BSS, h_{e,b} + f_e \geq b(W)
\]

(3.2.11)

By using different link-state dissemination scenarios, the reconstruction of the matrix \( Z \) will be through different approaches and information, in which different matrices \( H \) containing each entry of \( h_{e,b} \), for \( b \in BSS, e \in A_p \), will be yielded. Eq. (3.2.11) states that when links \( a \) and \( e \) are taken by \( W \) and \( P \), respectively, the resultant cost \( z_e \) is at least \( 1 - h_{e,b}/b(W) \). In this case, if \( h_{e,b} \geq b(W) \), there is sufficient sharable spare capacity along \( e \) that can be taken to protect any additional \( b(W) \) units of working capacity along link \( a \). Therefore, \( z_e \) can be as small as 0.

The following constraint imposes the link bandwidth limitation upon the consumption of spare capacity.

\[
\sum_{\forall a \in b, \forall a \in A_w} x_a + y_e \leq n \quad \forall a \in A_w, \forall b \in BSS \text{ and } h_{e,b} + f_e < b(W)
\]

(3.2.12)
Eq. (3.2.12) ensures that if \( h_{e,b} + f_e < b(W) \), links \( a \) and \( e \) cannot be used at the same time for \( W \) and \( P \) since there would be insufficient free and spare capacity for \( P \) along \( e \) given that \( W \) takes \( a \). At last, the variables \( z_e \) are required to be positive, which results in the following constraint:

\[
z_e \geq 0 \quad \forall e \in A_p
\]  

(3.2.13)

It is clear that the adoption of the second graph has successfully addressed the following three states for a protection path to take spare capacity: the case of \( h_{e,b} \geq b(W) \), the case of \( h_{e,b} + f_e \geq b(W) > h_{e,b} \), and the case of \( h_{e,b} + f_e < b(W) \). The former two cases are jointly defined by Eq. (3.2.11) and Eq. (3.2.13), where \( z_e \) is constrained no smaller than 0 and \( 1 - h_{e,b}/b(W) \) in the two cases, respectively; while the latter case is defined by Eq. (3.2.12), in which the traversal of \( P \) through \( e \) is not allowed if there is no sufficient capacity along the link. It can be observed that the use of the two residual graphs along with the constraints of Eq. (3.2.11), Eq. (3.2.12) and Eq. (3.2.13) impose a link bandwidth limitation constraint separately upon the selection of \( W \) and \( P \).

### 3.2.2 Performance Evaluation

The ILP is solved via CPLEX 7.1 for each case, where a low-end SUN Ultra 10 workstation with 384 MB memory is adopted. Three network topologies with 16, 22, and 30 nodes are taken, each of which contains 27, 44 and 63 arcs, respectively. The same performance metric as that adopted by D. Xu in [49], called bandwidth saving ratio (BSR), is used. BSR is characterized by the following facts: (1) the capacity along each link is infinite (and hence all the connection requests will be accepted); (2) the traffic is random and incremental; (3) after a fixed number of connection requests allocated, the total amount of bandwidth including both working and spare capacity (denoted as TBW) is calculated; (4) the TBW of No-Sharing (NS) is taken as the reference, and the value of BSR for each scheme is the difference between the corresponding TBW and the TBW of NS. Note that BSR is upper-bounded by 50%. For NS and DPIM-M-A, Sturbaile’s algorithm is adopted to allocate the physical routes of the working and protection path-pairs. DPIM-M-A will further conduct post-exploration of spare capacity sharing along the protection path while NS will not.

The computation time for allocating a connection with the ILP is less than 10, 2, and 0.3 seconds in the 30-, 22- and 16-node networks, respectively. For DPIM-SAM, DPIM-M-A and SRI, maximum resource sharing is explored on each link along the protection path after the physical route is determined since each node has the per-flow information. With SPI, nodes do not have per-flow information; therefore, the spare capacity may be over-reserved.

The simulation results are shown in Table 3.1 for all the cases in the three network topologies. It is clear that SCI yields the best BSR referring to the NS case, while the SPI scheme yields the worst. To our expectation, the SRI scheme outperforms DPIM-SAM, DPIM-M-A and SPI due to the two additional parameters adopted in the process of the reconstruction of matrix \( S \).

Table 3.2 shows the spectral 2-norm on the difference between the spare provision matrix \( \overline{S} \) in SCI and that by each of the others. It is clear that SRI can achieve the best estimation in the reconstruction of the matrix \( \overline{S} \) among all the approximating schemes.
| $|V|$ | $|A|$ | SCI | SPI | SRI | DPIM-SAM | DPIM-M-A |
|-----|-----|-----|-----|-----|----------|----------|
| 16  | 27  | 25.63 | -14.54 | 23.08 | 20.71 | 9.09 |
| 22  | 44  | 28.60 | 11.60 | 24.10 | 23.24 | 12.51 |
| 30  | 63  | 29.62 | 16.66 | 25.92 | 20.37 | 11.11 |

Table 3.1: Bandwidth saving ratio (BSR in %) for each case using the three network topologies.

| $|V|$ | $|A|$ | SCI | SPI | SRI | DPIM-SAM |
|-----|-----|-----|-----|-----|----------|
| 16  | 27  | 0   | 22.3 | 6.9  | 18.0     |
| 22  | 44  | 0   | 25.3 | 8.8  | 21.7     |
| 30  | 63  | 0   | 47.4 | 12.78| 42.8     |

Table 3.2: Spectral 2-norm of the matrix $S - S'$ (also denoted as $\|S - S'\|_2$), where $S$ and $S'$ are the spare provision matrix for SCI and that by any other scheme, respectively.

3.2.3 Conclusion

In this section the distributed control architectures were introduced for implementing shared protection in survivable networks with bandwidth guaranteed connections, where a novel scheme is proposed, called Sharing with Reduced Information (SRI). SRI is an enhancement of the DPIM-SAM scheme, and is committed to initiate a compromise between the amount of link-state dissemination and the ability of exploring resource sharing while spare capacity is allocated. With SRI, the node corresponding to each link additionally disseminates two parameters calculated through Singular Value Decomposition (SVD) transformation to improve the accuracy in the reconstruction of the spare provision matrix at the remote ingresses. Thus, resource sharing can be improved while the complexity of link-state remains $O(L)$, which is the same as that by DPIM-SAM. A novel Integer Linear Programming model is proposed, which can optimally find the least-cost disjoint path-pair according to a specific cost function and the spare provision matrix defined in each scenario. To verify the proposed scheme, simulation is conducted on three networks with 16, 22, and 30 nodes, respectively. The simulation results show that SRI outperforms DPIM-SAM, DPIM-M-A and SPI in terms of Bandwidth Saving Ratio (BSR). The accuracy in the reconstruction of the spare provision matrix in each scheme is also investigated, in which the SRI scheme is witnessed to yield the superb result compared to the other approximation schemes.
3.3 Claim 3.2: Reduced Information Scenario for Shared Segment Protection

This section introduces a new implementation of Shared Segment Protection (SSP) in mesh communication networks, in which a novel Integer Linear Program (ILP) is proposed under a reduced amount of routing information. In particular, the scheme of Distributed Partial Information Management with Sufficient and Aggregated Information (DPIM-SAM) [4] will be adopted for trading the optimality with the reduction on the computation time. We will verify the ILP formulation by comparing its result with the cases of SSP and Path Shared Protection under the complete routing information scenario (SCI) in terms of performance and computation time.

3.3.1 Introduction

Shared Segment Protection (SSP) has been proved to be an effective solution for dynamic survivable routing in mesh communication networks with bandwidth guaranteed tunnels [35]. It is also one of the best candidates for achieving survivability in mesh optical networks with Wavelength Division Multiplexing (WDM) as the core technology due to the better extent of resource sharing and the constrained restoration time compared with the conventional end-to-end path shared protection.

It is clear that the ILP formulated under the complete routing information scenario (SCI) (called SSP-SCI) in Section 2.4 can optimally find working and protection segments in each protection domain for a connection request. However, the extremely lengthy computation in solving the ILP discourages the adoption of SSP for most of the practical applications. The huge computation complexity is mainly caused by the large design space of protection domain allocation that exponentially grows with the network size, as well as the effort in correlating the inter-dependency of working and spare capacity of each pair of links (termed the Shared Risk Group (SRG) constraint). By observing the ILP formulation in Section 2.4, the two tasks greatly increase the complexity of the problem. Thus, it is expected that the problem size can be largely reduced in the case that the two tasks can be done in sequence at the expense of optimality.

In this section, a novel ILP formulation is proposed by adopting a reduced amount of routing information called SSP-DPIM-SAM, in which estimation is made for the sharable spare capacity along each link. In particular, the scheme of Distributed Partial Information Management with Sufficient and Aggregated Information (DPIM-SAM) [49] is taken in the ILP for SSP in order to trade the optimality with the reduction on the computation time in solving the SSP problem. With DPIM-SAM, the estimated spare capacity along each link to be allocated for the protection path does not depend on the route of working path. Hence, the complexity of the survivable routing problem significantly decreases. We will show in the simulation that the performance of SSP-DPIM-SAM is much better than PSP-SCI, and is comparable with that of SSP-SCI while the computation time can be reduced to its one thousandth for the network topology adopted.
3.3.2 DPIM-SAM

DPIM-SAM is a scheme devised for the implementation of shared protection in a distributed control environment, where the dissemination of dependency information from a node to all the other ingress nodes may be too expensive. In particular, DPIM-SAM allows link \( l \) to disseminate a scalar instead of a \( 1 \times |L| \) vector (denoted by \( \mathbf{s}^l \) and introduced in Section 3.1.1). The scalar to be disseminated is nothing but the maximum entry of \( \mathbf{s}^l \).

At any remote ingress node, the receipt of \( \mathbf{s}^l \) from all the other nodes will enable the reconstruction of the spare provision matrix for the whole network (denoted by \( \mathbf{S} \)) in the SCI scenario. This job can be done by simply putting all the \( \mathbf{s}^l \) column-vectors together, where \( l \in L \). With the knowledge of \( \mathbf{S} \), the remote ingress node will have sufficient information to perform explicit survivable routing with complete routing information, in which resource sharing along each link can be completely explored.

The above scheme will yield a dissemination complexity of \( O(|L| \times |BSS|) \), which may not be tolerable in some cases. The basic idea of link-state dissemination in DPIM-SAM is that each node exchanges the maximum entry in \( \mathbf{s}^l \) instead of the whole vector \( \mathbf{s}^l \) as done in the SCI scenario. Thus, the amount of dissemination can be greatly decreased and the complexity of this information exchanging mechanism becomes \( O(|BSS|) \). This can also guarantee the feasibility of the routing result at any remote ingress by over-estimating \( s_{j,l} \).

Compared with SCI, except for the reduction in link-state dissemination, DPIM-SAM requires the same complexity in the propagation of per-flow information when a connection is set up and the same effort in calculating \( \mathbf{s}^l \) at each corresponding node of a link. Fig. 3.3(d) illustrates the case.

3.3.3 ILP Formulation for Shared Segment Protection at DPIM-SAM Scheme

This section introduces a linear formulation for the segment shared protection problem in DPIM-SAM scenario, called SSP-DPIM-SAM. Our approach is to find a path \( P \), called mass protection path, which is composed of all the backup segments and some links along the working path. A simple example is shown in Fig. 2.4, where \( P \) is \((s-a-b-c-e-d)\). The first protection domain is formed by the working and protection segments \((s-c-b)\) and \((s-a-b)\), respectively; while the second is formed by \((a-b-d)\) and \((c-e-d)\), respectively. The allowance of overlapping between the working segments of two adjacent protection domains is to explore the largest design space so as to guarantee the optimality of the derived solution. Spare capacity sharing can happen between two protection segments of different protection domains, which results loops in \( P \). Though in the DPIM-SAM scenario \( P \) will be loopless this scenario is not able to benefit from sharing the spare capacity between two protection segments.

Two residual graphs are defined to facilitate the solving of this problem, each of which carries one or a few variables for the identification of the working and protection segment-pairs. The first is the graph for solving the working segments denoted as \( G_w(V, A_w) \), which is composed of links with \( f_j \geq b(W) \) for \( j \in A_w \).

The second residual graph \( G_p'(V, A_p') \) is to facilitate solving the mass protection path. This graph is composed of two types of links. The first one is the forward arcs, defined as the links where the amount of free capacity \( f_j \) plus that of the spare capacity \( v_j \) is larger than or equal
to $b(W)$ (i.e., $b(W) \leq f_j + v_j$ for $j \in A_p$). The second one is the reverse arcs, and are defined as the links of $E_w$ in a reversed direction. For easier understanding please see Fig. 3.5 and a more detailed description of the graphs can be found at the beginning of Section 2.4.

![Graph transformations for the links in $G$, which yields the two graphs $G_w$ and $G_p'$.](image)

Figure 3.5: Graph transformations for the links in $G$, which yields the two graphs $G_w$ and $G_p'$.

To introduce the target function of the proposed ILP formulation, the following two flow indicators are defined in the graphs $G_w$ and $G_p'$: $x_{a,b}$ is a binary variable with size $|A_w|$ defined in graph $G_w$, while $y'_{u,v}$ is a binary variable with size $|A'_p|$ defined in graph $G_p'$. The objective function is as follows:

$$\text{Minimize} \quad \sum_{(a,b) \in A_w} b(W) \cdot c_{a,b} \cdot x_{a,b} + \sum_{(u,v) \in A'_p} b(W) \cdot c^p_{u,v} \cdot y_{u,v}$$

(3.3.1)

where $c_{a,b}$ is the cost per unit of working bandwidth to reserve link $(a, b)$, which was introduced in Section 1.2.6. The cost for $Q$ to take $e \in G_p'$ is denoted as $c^p_e$ and can be expressed as follow:

$$c^p_e = \begin{cases} \max \left\{ 0, 1 - \frac{\max_j \left\{ s_{e,j} \right\}}{b(W)} \right\} & \text{if } (e') \text{ is forward arc} \\ 0 & \text{if } (e') \text{ is reversed arc} \end{cases}$$

(3.3.2)

where $\max_j \left\{ s_{e,j} \right\}$ is disseminated by the corresponding node of link $e$ to all the remote ingress nodes, and the spare provision matrix reconstructed at those remote ingress nodes will take all the entries along the column assigned to link $e$ of the matrix as $\max_j \left\{ s_{e,j} \right\}$. Hence the link cost $c^p_e = \max \left\{ 0, (1 - \max_j \left\{ s_{e,j} / b(W) \right\}) \right\}$ is 0 if the link can be shared and 1 if no spare capacity is available.

Each of $x_{a,b}$ and $y'_{u,v}$ is a binary number indicating whether $W$ and $Q$ traverses $(a, b) \in A_w$ and $(u, v) \in A'_p$, respectively.

The target function is subject to the following constraints:

$$\sum_{(a,b) \in A_w} x_{a,b} - \sum_{(b,a) \in A_w} x_{b,a} = \begin{cases} 1 & \text{if } a \text{ is the source node } s \\ -1 & \text{if } a \text{ is the destination node } d \\ 0 & \text{otherwise} \end{cases}$$

(3.3.3)

$$\sum_{(a,b) \in A'_p} y'_{a,b} - \sum_{(b,a) \in A'_p} y'_{b,a} = \begin{cases} 1 & \text{if } a \text{ is the source node } s \\ -1 & \text{if } a \text{ is the destination node } d \\ 0 & \text{otherwise} \end{cases}$$

(3.3.4)

Each Eq. (3.3.3) and (3.3.4) is the flow conservation constraint for $W$ and $Q$, respectively.
It is important to note that $x_{a,b}$ and $y'_{a,b}$ have to be SRG disjoint. However, a link can be taken by $y'_{a,b}$ in a reversed direction only if $x_{a,b}$ passes through it. Besides, each reversed arc can be used only once since the algorithm only allows two working segments overlapped. The above statements can be formulated into the following two constraints:

$$
\sum_{y'_{a,b} \text{ in the same } \text{SRG} \notin W \text{ as } a} y'_{a,b} \leq n \quad \text{for } \forall (b) \in A_w
$$

$$
x_{a,b} \geq y'_{a,b} \quad \text{for } \forall (a, b) \in A_w, \forall (a, b) \in A'_p
$$

(3.3.5)

where $y'_{a,b}$ represents the reversed link of $A'_p$. Under the above two constraints, $Q$ has to be SRG disjoint from $W$ except for those arcs of $W$ being reversed (see Fig. 2.4). These two constraints not only assert the SRG disjointness of the working and the corresponding backup segment, but also facilitate the indication of the switching/merging nodes for each protection domain along $W$.

The number of variables in an ILP formulation directly influences the computation time required to solve the formulation. In this formulation, the number of variables is $|A_w| + |A'_p|$, and the number of rows in the constraint matrix (where the linear formulation can be expressed in a general form as $A \cdot x = b$ with a target to minimize $c \cdot x^T$) is: $|A_w| + 2 \cdot |V|$.

### 3.3.4 Performance Evaluation

The ILP is solved via CPLEX 7.1 for each case, where a low-end SUN Ultra 10 workstation with 384 MB memory is adopted. Three network topologies with 16, 22, and 30 nodes are taken, containing 27, 44 and 63 arcs, respectively. The same performance metric as that adopted by D. Xu in [49], called bandwidth saving ratio (BSR), is used. BSR is characterized by the following facts: (1) the bandwidth along each link is infinite (and hence all the connection requests will be accepted); (2) the traffic is random and incremental; (3) after a fixed number of connection requests allocated, the total amount of bandwidth including both working and spare capacity (denoted as TBW) is calculated; (4) the TBW of No-Sharing (NS) is taken as the reference, and the value of BSR for each scheme is the difference between the corresponding TBW and the TBW of NS. Note that BSR is upper-bounded by 50%.

Four different scenarios were compared: shared path protection with complete routing information (SPP-SCI), shared segment protection with distributed partial information management with sufficient and aggregated information (SSP-DPIM-SAM), shared path protection with distributed partial information management with sufficient and aggregated information (SSP-DPIM-SAM), shared segment protection with complete routing information (SSP-SCI).

The simulation results are shown in Table 3.1 for all the cases in the three network topologies. It is clear that SSP-SCI yields the best BSR referring to the NS case, while the SSP-DPIM-SAM scheme yields the worst. To our expectation, the SPP-SCI and SSP-DPIM-SAM schemes have very similar performance in terms of BSR.

Table 3.2 shows the computation time for allocating a connection with ILP. It is clear that SSP-DPIM-SAM with the above described effective ILP implementation is the quickest. An effective implementation of SPP-DPIM-SAM with ILP formulation was not the scope of this study, though its calculation time can be significantly improved with similar ideas as it was described in this section. In our experience the computation time for allocating a connection
with the ILP with SSP-DPIM-SAM is 100 times quicker than SPP-SCI, and 1000 times than SSP-SCI.

<table>
<thead>
<tr>
<th></th>
<th>SPP-SCI</th>
<th>SSP-DPIM-SAM</th>
<th>SPP-DPIM-SAM</th>
<th>SSP-SCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>N16</td>
<td>45.8</td>
<td>43.4</td>
<td>40.9</td>
<td>45.8</td>
</tr>
<tr>
<td>N22</td>
<td>68.9</td>
<td>46.6</td>
<td>29.1</td>
<td>68.9</td>
</tr>
<tr>
<td>N30</td>
<td>63.3</td>
<td>58.3</td>
<td>51.7</td>
<td>65</td>
</tr>
</tbody>
</table>

Table 3.3: Bandwidth saving ratio (BSR in %) for each case using the three network topologies.

<table>
<thead>
<tr>
<th></th>
<th>SPP-SCI</th>
<th>SSP-DPIM-SAM</th>
<th>SPP-DPIM-SAM</th>
<th>SSP-SCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>N16</td>
<td>0.045</td>
<td>0.022</td>
<td>0.227</td>
<td>454.7</td>
</tr>
<tr>
<td>N22</td>
<td>0.8146</td>
<td>0.1349</td>
<td>1.0952</td>
<td>2030.8</td>
</tr>
<tr>
<td>N30</td>
<td>0.4567</td>
<td>0.1433</td>
<td>1.1957</td>
<td>2171.6</td>
</tr>
</tbody>
</table>

Table 3.4: The runtime for each case using the three network topologies.
Chapter 4

Conclusion

The aim of the dissertation was to present novel and efficient routing algorithms in survivable mesh communication networks that meet several Quality-of-Service requirements to cope with any possible limitation of hardware and of the network control. Not only algorithms, but also well adapted models are used to find optimal solution. The obtained results have shown that both the modelling and algorithmic approaches contain novelties and their joint development has potentials.

The dissertation at first investigates shared protection in mesh telecommunication networks. Using linear algebra the problem is divided into sub-problems, which are solved with scalable heuristics.

Shared Segment Protection (SSP) is the second topic investigated in the dissertation. The Integer Linear formulation (ILP) for SSP used to be an open problem, and the dissertation provides a solution that can derive the optimal answer. The ILP was expanded to explore the maximum design dimensions that have been covered by past studies. In our approach all the design criteria are considered and optimized in a single step. Although the lengthy computation in solving the ILP may prevent it from serving as an on-line solution, the ILP can nonetheless provide a benchmark to evaluate any other heuristic counterpart.

The third claim focuses onto the distributed routing algorithms. A new algebraic way to investigate the performance of different distributed control architectures for shared protection is introduced. Furthermore, new methods are proposed that yield a compromise between (1) the amount of link-state dissemination, (2) the ability of exploring resource sharing and (3) calculation time.

Shared path protection and shared segment protection are recognized as two of the most promising resilience strategies to equip the networks with. Performance evaluation with simulation was carried out to demonstrate the benefits of all the algorithms and solutions.
Bibliography


Index

c ⊙: an operator defined on matrices, 21
∥ ∥∥: matrix norm defined for shared protection, 23

A: working path-SRG’ incidence matrix, 20
active path, 9
adaptive routing, 4
AIBP: Average Increase in Blocking Probability, 60, 68
AITC: Average Increase of Total Cost, 60, 68
arc-reversal transformation, 50
ASON: Automatic Switched Optical Network, 2
ASTN: Automatic Switched Transport Network, 2
asymmetrically weighted disjoint path-pair problem, 26
ATM: Asynchronous Transfer Mode, 2
average waiting time, 13

B: backup path-link incidence matrix, 20
b(): bandwidth of the connection or path, 15
backup path, 9
backup segments, 45
blocking probability, 13
BSR: bandwidth saving ratio, 93
BSS: Basic Set of SRGs, 19
C: the cost column vector the links, 22
centralized control architecture, 4
complete routing information scenario, 43
D: a set of already established connections, 20
d: destination node, 15
dedicated protection, 9
distributed control architecture, 4
diverse routing, 5

\(D^K\): a set of already established working and protection path segments, 46

\(D_i\): a set of already established connections involved in the \(i^{th}\) BSS, 87

DPIM-M-A: Distributed Partial Information Management with Minimum Bandwidth Allocation, 88
DPIM-SAM: Distributed Partial Information Management with Sufficient and Aggregated Information, 85, 89, 96
DWDM: Dense Wavelength Division Multiplexing, 2, 13
dynamic routing problem, 12

E: unit matrix, 21
Easy Links, 35
EXC: Opto-Electronical Cross-Connect, 14
f: free capacity, 15
forward arcs, 50

\(G: SRG’ - BSS\) incidence matrix, 21
GMPLS: Generalized MPLS, 2, 13
$H$: sharable spare capacity matrix, 22

$h_j$: sharable spare capacity along link $j$, 16

$H^W$: non-sharable spare capacity vector, 22

$H^W_k$: sharable spare capacity vector of the $k^{th}$ segment, 47

$i$-counterpart, 68

IETF: Internet Engineering Task Force, 2

ILP: Integer Linear Programming, 1

ITU-T: International Telecommunications Union
Telecommunications, 2

$k_W$: the number of segments of $W$, 47

$L$: set of the links of the network, 15, 45

mass protection path, 50, 96

maximal waiting time, 13

MPLS: Multi-Protocol Label Switching, 1, 13

$N$: set of the nodes of the network, 15, 45

network load, 13

network utilization, 13

Next Generation SDH/SONET, 2

NMS: Network Management System, 4

Offset of Optimality, 58

OSPF weights, 15

OSPF: Open Shortest Path First, 15

OXC: Optical Cross-Connect, 13

$P$: protection path, 15

$P$: $|1| \times |L|$ size vector representing $P$, 21

$P$: a diagonal matrix of size $|L| \times |L|$ representing $P$, 22

$P^K$: a matrix of size $k_W \cdot |L| \times |L|$ representing all $P^K$s, 47

$P_k$: backup segments, 45

$P^k$: $|1| \times |L|$ size vector representing $P_k$, 47

post-exploration of sharing, 88

pre-planning spare capacity, 9

protection domains, 42

protection path, 9

protection segments, 45

$Q$: mass protection path, 50

$q$: working capacity, 15

QoS: Quality of Service, 1, 3

$R$: sharable capacity ratio matrix, 22, 47

$r$: scaling parameter of spare link-state, 17

RCI: Reduced Complete Information, 85

reversed arcs, 50

$S$: spare provision matrix, 19, 21, 46

$s$: source node, 15

$s_j$: non-sharable spare capacity along link $j$, 16

S/M: switching/merging capacity, 60

SCI: Sharing with Complete Information, 85, 87

sharable spare capacity, 9

single failure scenario, 12

SONET/SDH: Synchronous Optical Network/
Synchronous Digital Hierarchy, 2

spare capacity, 12

spare link-state, 16

SPI: Scenario of Partial Routing Information, 85, 88

SPP: Shared Path Protection, 5, 9, 24

SRG-disjoint, 11

SRG: Shared Risk Group, 11

SRG$_{ud}$: Second-type of SRG, 17
$\textit{SRG}_{\text{res}}$: First-type of SRG, 17

SRI: Sharing with Reduced Information, 89

SRLG: Shared Risk Link Group, 19

SSP: Shared Segment Protection, 5, 10, 42, 49

survivable routing, 5, 9

$\sum_{W_k}$: non-sharable spare capacity vector of the $k$th segment, 47

TE: Traffic Engineering, 2

trial, 68

Two-Step-Approach, 26, 27

UNI: User-Network Interface, 2

$V$: spare capacity vector, 22

$v$: spare capacity, 15

$V_f$: switching/merging nodes, 62

$V_l$: limited operational nodes, 62

$V_m$: merging nodes, 62

$V_s$: switching nodes, 62

$W$: working path, 15

$W$: $|L| \times |L|$ size vector representing $W$, 21

$W$: a diagonal matrix of size $|L| \times |L|$ representing $W$, 22

$W'$: $|L| \times |BSS|$ size vector representing $W$, 22

$W'$: a diagonal matrix of size $|BSS| \times |BSS|$ representing $W$, 22

$W_k'$: $|L| \times |BSS|$ size vector representing $W_k$, 47

wavelength assignment, 14

wavelength converter, 13

wavelength graph, 15

$W_k$: working segments, 45