CONFIGURATION OF FAULT TOLERANT INFOCOMMUNICATION NETWORKS

Péter Laborczi

PhD Dissertation

Supervised by

Dr. Tibor Cinkler and Dr. Gyula Sallai
High Speed Networks Laboratory
Inter-University Centre of Telecommunications and Informatics
Department of Telecommunications and Telematics
Budapest University of Technology and Economics

Budapest, Hungary
2002
© Copyright 2002
Péter Laborczi
High Speed Networks Laboratory
Inter-University Centre of Telecommunications and Informatics
Department of Telecommunications and Telematics
Budapest University of Technology and Economics

\footnote{The reviews and the minutes of the PhD defense are available from the Dean’s Office.}
“The only wise God, be honour and glory.”
1 Timothy 1.17
Contents

List of Figures vi
List of Tables viii
List of Abreviations ix
Abstract x
Kivonat xi
Acknowledgements xii

1 Introduction 1
  1.1 Graph Theory in Network Optimisation 2
    1.1.1 Fundamentals 3
    1.1.2 Paths 3
    1.1.3 Flows and Cuts 4
    1.1.4 Complexity of Algorithms 5
  1.2 Resilience Methods 6
    1.2.1 A Simple Example 7

2 Configuration of Infocommunication Networks 9
  2.1 Static LSP Routing Algorithms for MPLS Networks 9
    2.1.1 The Mathematical Problem 9
    2.1.2 The Unsplittable Flow Problem 10
    2.1.3 The Combinatorial Approach (CA) 10
    2.1.4 Original and Improved Simulated Allocation 11
    2.1.5 Numerical Results 13
    2.1.6 Summary 14
  2.2 Network Configuration with Dedicated and Shared Protection 14
    2.2.1 Problem Formulation 15
    2.2.2 Alternative Methods for Dedicated Protection 15
    2.2.3 Shared Protection: Thrifty Capacity Allocation (TCA) 18
    2.2.4 Heuristics for Improving the Performance 22
    2.2.5 Numerical Results 23
    2.2.6 Summary 26
  2.3 Algorithms for Asymmetrically Weighted Pair of Disjoint Paths in Survivable Networks 27
4.5 Numerical Results ......................................................... 81
  4.5.1 Simulation Environment ........................................... 81
  4.5.2 Simulations on Blocking Probability ........................... 82
  4.5.3 Simulations with Protected Connections ....................... 88
4.6 Summary ................................................................. 91

5 Conclusions .............................................................. 93

References ................................................................. 94
List of Figures

1.1 The connection s-t is realized along a single path or can be split ............. 4
1.2 "Vertex splitting": Substitute a vertex of capacity $c(v)$ with an edge of capacity $c(v)$ .................................................. 5
1.3 (b) Graph model of a network, (b) establishment of a connection, (c) path protection, (d) link protection ........................................ 8

2.1 Flow Chart of Iterative Capacity Splitting (ICS) ................................. 18
2.2 Ideas for shared protection: (a) Capacity compression, (b) Capacity reuse .... 19
2.3 The examined networks: N5, N15, N25 and N35 ................................. 24
2.4 Running time of the 6 algorithms in 10 test networks ........................... 25
2.5 Resource usage (Utilization) of the 6 algorithms in 10 test networks ... 26
2.6 Running time vs. Utilization in N25, N25+4l, N25+10% using the 6 algorithms . 27
2.7 Example when SFR gives optimal solution. (Dashed arrows represent the working path, with flow size of $\alpha$ and solid lines the protection path, with flow size of 1). 30
2.8 Example when SFR yields a split flow. (Numbers on the arcs represent the flow values.) 31
2.9 Example for repairable split of LPR. The numbers next to the arc represent the size of the flow. 31
2.10 Example for unrepairable split. If $\alpha = 5$ the total cost is $7\alpha + 15 = 50$, while the disjoint path’s cost would be $9\alpha + 9 = 54$ ............................ 32
2.11 An example of the cost function plotted against $\alpha$ ............................ 33
2.12 Flowchart for solving the asymmetrically weighted optimal disjoint path-pair problem .................................................. 34
2.13 A typical function of cost against alpha normalized with the value of the optimal solution, when the optimal solution differs from the Sunballe’s one. 34
2.14 Node and link availability model .................................................. 37
2.15 Flowchart of the Iterative Availability Enhancement (IAE) .................... 40
2.16 The extended (29 node) EON network ........................................ 41
2.17 The increase of minimum (MIN) and average (AVE) availability ............ 42

3.1 Peer model: Common control plane for IP and WDM layers ........................ 47
3.2 PLL example .............................................................................. 49
3.3 PUL example ............................................................................. 49
3.4 A bad mapping ........................................................................... 49
3.5 Legend ......................................................................................... 49
3.6 The new cost $i_{e2}$ of edge $e2$ is a function of edge-load $load_{e2}$: $i_{e2} = \tanh(a load_{e2} - b) + c$ .................................................. 57
3.7 Project two layers into one .......................................................... 58
3.8 In case of PBL there exist three "partly" disjoint paths: working path, upper layer protection path (PUL path) and lower layer protection path (PLL path) .......... 60
3.9 The examined networks: N5, N15, N25 and N35 ............................... 61
3.10 Percentage of node-pairs with physically disjoint path-pairs after one-by-one mapping (SPOO) and Smart-Mapping (SMART) ........................................ 62
3.11 Percentage of node-pairs where 2D (SPOO) and Smart-Routing (SMART) can find physically disjoint pair of paths ............................ 63
3.12 Throughput after iteration 1,2,3 in case of PLL for the 15-node network (N15) . 63
3.13 Comparison of total required capacity for the four test networks (N5, N15, N25, N35), three multilayer protection schemes (PLL, PUL, PBL), and two protection strategies (DP, SP) .............................................................. 64
3.14 N15A Network ............................................................................. 69

4.1 Ideas for shared protection: (a) Capacity compression, (b) Capacity reuse ...... 79
4.2 Blocking probability using different WS strategies in Net_P, Net_ATKT and Net_56 83
4.3 Blocking probability using different weight functions in Net_P, Net_ATKT and Net_56 85
4.4 Network utilization using different WS strategies and weight functions in Net_P, Net_ATKT and Net_56 ................................................................. 87
4.5 Blocking probability with different number of fibers and wavelengths in Net_P and Net_56 ................................................................. 88
4.6 Blocking probability without protection, with dedicated and shared protection in Net_ATKT and Net_56 networks ........................................ 89
4.7 Blocking probability with different number of fibers and wavelengths in case of Dedicated and Shared Protection in Net_P ....................................... 90
4.8 Number of wavelengths/fibers required for the same blocking probabilities in the three test networks .................................................. 91
List of Tables

2.1 Percentage of allocated demands and running times of the algorithms . . . . . . . . 13
2.2 Usage of link capacities .......................................................... 14
2.3 The 8 examined networks: originals and their extensions ................................. 23
2.10 Changed link capacities on the EON network ...................................... 42
2.4 The Numerical Results: Performance-Comparison of Algorithms for 1+1 and TCA
Allocation Strategies .................................................................. 43
2.5 Numerical results of a (denser) network with 30 nodes and 63 links ............... 43
2.6 Numerical results of a (sparser) network with 35 nodes and 51 links ............... 44
2.7 Numerical results of 10 relevant demands in a network with 400 nodes and 1378
links .................................................................................. 44
2.8 Percentage of optimal routed demands in the 8 examined networks ............... 44
2.9 The increase of runtime on different networks (α = 5) ............................. 44
3.1 The 4 examined networks: Number of nodes, links and IP demands .............. 62
3.2 Numerical results for Protection at the Lower Layer .................................. 66
3.3 Numerical results for Protection at the Upper Layer (in the two rows below physically
jointness of the two paths is enabled) ............................................ 67
3.4 Numerical results for Protection at Both Layers (in the two rows below physically
jointness of the two paths is enabled) ......................................... 68
3.5 The enhancement of average and minimal availability using MAE on network
N15A .............................................................................. 69
4.1 The test networks ........................................................................ 82
4.2 Gain of WS strategies - 55% offered traffic ........................................... 84
4.3 Gain of WS strategies - 75% offered traffic ........................................... 84
4.4 Gain of weight functions - 55% offered traffic ........................................ 86
4.5 Gain of weight functions - 75% offered traffic ........................................ 86
4.6 Effect of link failure on blocking probability ........................................... 90
List of ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSI</td>
<td>American National Standards Institute</td>
</tr>
<tr>
<td>APS</td>
<td>Automatic Protection Switching</td>
</tr>
<tr>
<td>ASON</td>
<td>Automatic Switched Optical Networks</td>
</tr>
<tr>
<td>ASTN</td>
<td>Automatic Switched Transport Networks</td>
</tr>
<tr>
<td>ATM</td>
<td>Asynchronous Transfer Mode</td>
</tr>
<tr>
<td>CA</td>
<td>Combinatorial Approach</td>
</tr>
<tr>
<td>CPS</td>
<td>Common Pool Survivability</td>
</tr>
<tr>
<td>DP</td>
<td>Dedicated Protection</td>
</tr>
<tr>
<td>DWD M</td>
<td>Dense Wavelength Division Multiplexing</td>
</tr>
<tr>
<td>EON</td>
<td>European Optical Network</td>
</tr>
<tr>
<td>ETSI</td>
<td>European Telecommunications Standards Institute</td>
</tr>
<tr>
<td>GMPLS</td>
<td>Generalized Multi Protocol Label Switching</td>
</tr>
<tr>
<td>IAE</td>
<td>Iterative Availability Enhancement</td>
</tr>
<tr>
<td>ICS</td>
<td>Iterative Capacity Splitting</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>ILP</td>
<td>Integer Linear Program</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>LDP</td>
<td>Label Distribution Protocol</td>
</tr>
<tr>
<td>LPR</td>
<td>Linear Programming Relaxation</td>
</tr>
<tr>
<td>LR</td>
<td>Linear Relaxation</td>
</tr>
<tr>
<td>LSP</td>
<td>Label Switched Path</td>
</tr>
<tr>
<td>LSR</td>
<td>Label Switching Router</td>
</tr>
<tr>
<td>MAE</td>
<td>Multilayer Availability Enhancement</td>
</tr>
<tr>
<td>MCF</td>
<td>Minimal Cost Flow</td>
</tr>
<tr>
<td>MCmCmCf</td>
<td>Minimal Cost Multi-Commodity Flow</td>
</tr>
<tr>
<td>MCNF</td>
<td>Minimum Cost Network Flow</td>
</tr>
<tr>
<td>MEMS</td>
<td>Micro Electro-Mechanical Systems</td>
</tr>
<tr>
<td>MNCA</td>
<td>Multilayer Network Configuration Algorithm</td>
</tr>
<tr>
<td>MPAS</td>
<td>Multi Protocol Lambda Switching</td>
</tr>
<tr>
<td>MPLS</td>
<td>Multi Protocol Label Switching</td>
</tr>
<tr>
<td>MTTF</td>
<td>Mean Time To Failure</td>
</tr>
<tr>
<td>MTTR</td>
<td>Mean Time To Repair</td>
</tr>
<tr>
<td>OADM</td>
<td>Optical Add-Drop Multiplexer</td>
</tr>
<tr>
<td>O-D</td>
<td>Origin - Destination</td>
</tr>
</tbody>
</table>
ODSI  Optical Domain Service Interconnect
OIF  Optical Interworking Forum
OSPF  Open Shortest Path First
OXC  Optical Cross Connects
PBL  Protection at Both Layers
PLL  Protection at the Lower Layer
PP  Protection Path
PUL  Protection at the Upper Layer
QoS  Quality of Service
RWA  Routing and Wavelength Assignment
SA  Simulated Allocation
SDH  Synchronous Digital Hierarchy
SFR  Single Flow Relaxation
SP  Shared Protection
TAC  Tiny Additional Costs
VC  Virtual Container
VP  Virtual Path
WDM  Wavelength Division Multiplexing
WP  Working Path
WR  Wavelength Routing
WS  Wavelength Selection
2D  Two Step Dijkstra Algorithm
Abstract

In my Dissertation algorithmical solutions are presented for unsolved configuration problems arising in modern infocommunication networks (WDM, MPLS, ATM). Furthermore, I propose better solutions for already solved problems, e.g., algorithms of shorter running time, or yielding lower utilization of the network, or less blocking probability.

Reliability of infocommunication networks is increasingly important, consequently, fault tolerant operation is to be ensured. This should be reached in a way that is as cheap, as efficient, and as fast as possible, according to the operator’s priorities. Other criteria is that the algorithms should not depend on the technology, but could be applied in networks of both single and multilayer, both dedicated and shared protection, with centralized and distributed management, with one wavelength or using wavelength division multiplexing (WDM). Instead of finding the globally optimal solution, good and fast approximate methods are to be developed.

First, I deal with existing, single-layer networks with capacity constraints, in which virtual paths can be established (e.g., ATM, MPLS). The method of Simulated Allocation will be applied and improved for configuration of LSPs in MPLS networks. I give a two phase heuristic method for configuration of fault tolerant networks, which is much faster than to solve an Integer Linear Programming task, and gives results in large scale networks as well, for the price that the used capacity increases slightly. In addition, a new optimization problem is proposed, the “asymmetrical weighted” pair of disjoint paths, that models the reality better than previous approaches, and I give approximate solutions for this problem. Furthermore, I study reliability from an other point of view: methods are proposed to evaluate and advance minimum and average point-point availability of single layer networks.

After this, I deal with unanswered questions of multilayer networks (e.g., IP-over-WDM) by jointly handling routing and protection issues of the network layers. The configuration problem of multilayer fault tolerant networks is formulated as an Integer Linear Programming task, assuming several protection methods, an efficient heuristic method is given, and availability is studied in multilayer networks as well.

Finally, a new method is proposed that handles shared protection in dynamic WDM context in an efficient way. Operations are defined that are necessary when the traffic demands are set up or torn down in the network. The effectiveness of the method is proved with numerous simulations.
Kivonat

Értekezésben korszerű infokommunikációs hálózatokban (WDM, MPLS, ATM) felmerülő megoldatlan, konfigurálással kapcsolatos problémákra adók algoritmusok megoldásait. Ezen kívül már megoldott problémákra keresk jobb megoldást, például rövidebb futási idővel rendelkező algoritmust, vagy olyat, amely a hálózat alacsonyabb terheltségét, vagy kisebb blokkolási valószínűséget eredményez. Az infokommunikációs hálózatok megértésére egyre nagyobb hangsúlyt kap, ezért megérthető, hibatűző működést kell kialakítani bennük. Ezt a hálózat üzemeltetőjének szempontjai alapján minden más, mind hatékonyabbá, mind gyorsabban kell elérni. Másik szempont, hogy az algoritmusok ne legyenek technológiafüggők, azaz alkalmazhatók legyenek mind több elterjedt infokommunikációs hálózatban: egyrészégi vagy többrétegű hálózatokban; hozzárendelt vagy megosztott védelmet használó hálózatokban; központi vagy elosztott vezérlésű hálózatokban; egy hullámhosszú vagy hullámhosszszabásos nyálkahatók (WDM) használó hálózatokban egyaránt. A globális optimum megkeresése helyett jö, de gyors közelítő módszereket keresnek.

Először olyan kapacitásokon kialakított egyrészégi hálózatok konfigurálásával foglalkozzunk, mélyekben kialakültak virtuális utak (pl. ATM, MPLS). A szimulált helyfoglalás módszerét alakozással LSP-k meghatározására MPLS hálózatokban, és a módszert négy ponton továbbfejleszten. Hibatűző hálózatok konfigurálására kétfázisú heurisztikus iterációs módszert adók, amely az optimális megoldást nyújtó egészségtelen-lineáris programozás módszerével szemben polinom idejű, nagyobb hálózatokban is talál megoldást, és jóval gyorsabban lefut, a lefoglalt kapacitás elsőként mértéke növekedése árán. Ezen kívül ezzel új, az eldijú útpárharcos algoritmusokkal a valósághoz közeli álló optimálisírási feladatot javasolják, a “férfi” súlyozott útpárharcos problémáját, és közelítő módszereket adók annak megoldására. Ezután más oldalról közelíten meg a megbízhatóság kérdését, és új módszereket javasolják a rendelkezésre állás kétféle eljárásra és növekedésre egyrészégi hálózatokban.

Ezt követően többrétegű hálózatokban (pl. WDM felett megvalósított IP) felmerülő megoldatlan kérdésekkel foglalkozom oly módon, hogy együtt kezeljük a különböző rétegek útválasztással és védelemmel kapcsolatos kérdéseit. Megfogalmazom a többrétegű hibatűző hálózatok konfigurálásának problémáját egészségtelen lineáris programozási feladatként különböző védelmi módszereket feltételezve, és hatékony heurisztikus algoritmust adók annak megoldására, majd többrétegű hálózatok rendelkezésre állását vizsgálom.

Végül olyan módszereket írjuk, amely hatékonyan kezeli a megosztott védelmet dinamikus WDM hálózatokban. Az igények – dinamikusán bekötvekező – felépítésekor és lebontásakor szükséges eljárásokat definiáló, és a módszer hatékonyágát szimulációkkal bizonyítom.
Acknowledgements

I would like to express my thanks to my advisors: Dr. Tibor Cinkler, who initiated me into this beautiful research field, has kept at guiding me throughout my research work expertly, persistently and enthusiastically; and Dr. Gyula Sallai for his valuable advice as well as for his help during the finalisation of my thesis.

My research work has been done in the framework of research co-operation between Ericsson and the High Speed Networks Laboratory (HSNLab) and within the Inter-University Centre of Telecommunications and Informatics (ETIK). I am grateful to Dr. Miklós Boda, Dr. Hans Eriksson (Ericsson) and Dr. Tamás Henk (HSNLab) for their continuous support.

I would like to thank all my co-authors I have worked with in research groups of HSNLab, Ericsson and ETIK. I have learnt particularly much from Dr. András Recski both on his lessons and during our co-operations. Furthermore, I thank Imre Budai (PanTel) for supporting the work on results of Section 2.4.
Chapter 1

Introduction

The demand to carry large amount of data as fast and as reliably as possible is continuously increasing, in parallel with formation of the information society. Nowadays, the role of optical networks is increasingly important, on which the information is transmitted using Wavelength Division Multiplexing (WDM) [1], Asynchronous Transfer Mode (ATM) [2], or Multi Protocol Label Switching (MPLS) [3, 6]. Each link is able to carry huge amount of traffic, thus a possible failure causes loss of tremendous data. In order to prevent this, the established paths should be protected. Questions that arise in infocommunication networks, such as routing [4, 5], protection [6, 7], reliability evaluation [8], reliability enhancement, and topology augmentation [9], are intensively studied in the literature. The research of these subjects is mostly separated and the implementation is also independent of each other. Solving these tasks jointly (e.g., routing and protection) leads to better result, but in this case the optimization problem will be more complex as well.

In the literature a shortest path algorithm (e.g., Dijkstra [10]) is used in general for determining a path. Using this method, the network utilization can be significantly deteriorated since some links may become overloaded while some of them under-utilized. Other possibility is to lead the traffic according to the solution of a minimal cost multi-commodity flow [11]. This has the disadvantage that split flows can occur in the solution. Furthermore, for protecting the paths, two edge- or node-disjoint paths are to be determined [12] and in case of failure, traffic should be switched from the working to the protection path with Automatic Protection Switching (APS) [13]. In addition to protection path determination, another issue is how much capacity the network operator should reserve for backup purposes on each link. Most of today's networks apply dedicated protection, that reserves capacity for each path one-by-one. Restoration after the failure needs less resource, however, it may cause significant data loss because of its slower operation. Employing shared protection will decrease both backup capacity and amount of lost data significantly since paths and resources are prepared for both working and failure state (see Section 1.2) [14, 15].

The majority of existing network planning methods and graph theoretical algorithms can be applied only on a given, separated network, e.g., WDM, ATM or SDH. This model should be extended, because these technologies work in practice not alone but jointly with other technologies (e.g., IP-over-WDM, ATM-over-SDH). In this case the network technologies are called layers. In multilayer networks the outlined questions will be also more complex because of interactions among different network layers.

In my PhD dissertation I am going to give answers on the above mentioned challenges related
to configuration that arise jointly with modern infocommunication networks. The main goal is to present algorithmical solutions for unsolved problems and to propose better solutions for already solved problems, e.g., algorithms with shorter running time (and thus able to solve more complex problems), or yielding lower utilization of the network, or less blocking probability. Instead of finding the globally optimal solution, good and fast approximate methods are to be developed. The methods should be capable to handle networks of both single and multilayer, both dedicated and shared protection, with centralized and distributed management, with one wavelength or using wavelength division multiplexing (WDM). The principle of APS is assumed by using dedicated or shared protection, while restoration techniques are according to the above beyond the scope of this work. We concentrate on existing, implemented networks and not on planning of them, although the results are also applicable in a later phase of a network planning process, e.g., in dimensioning or routing phase.

The structure of the Dissertation is as follows.

In the remaining part of this Chapter the essential terms and methods connected to graph theory and network resilience are summarized. In Chapter 2 we deal with configuration of existing, one-layer networks with capacity constraints. Methods are proposed for networks in which virtual paths can be realized (e.g., ATM, MPLS). In Chapter 3 unsolved routing and protection questions of multilayer networks are handled. In Chapter 4 application of shared protection in dynamic WDM context is studied and an efficient new method proposed.

1.1 Graph Theory in Network Optimisation

Graphs are efficiently manageable mathematical representations of telecommunication networks: the vertices of a graph correspond to routers or switches while edges represent the cables, radio and fiber links that connect the routers. In this Chapter terms and methods are summarized that are used in the remaining of the Dissertation for modeling telecommunication networks [B1].

In the beginning physical transmission medium has been directly used to transmit information, consequently it was easy to handle. The growing need to transfer information requires networks that are able to carry data of complex structure (picture and voice) efficiently. Since Internet Protocol (IP) is the most widespread application, which enables different paths for different packages, an efficient traffic management model is of great importance. MultiProtocol Label Switching (MPLS) that satisfies the above requirements in a simple way has been worked out for IP technology. MPLS Networks relay on sending data packets over Label Switched Paths (LSPs). The system of LSPs is similar to the concept of Virtual Path (VP) system applied in Asynchronous Transfer Mode (ATM) networks. In Synchronous Digital Hierarchy (SDH) networks the cross-connect for Virtual Containers (VCs) has to be set up, using as few resources and as short paths as possible. In Wavelength Division Multiplexing (WDM) networks or MultiProtocol Lambda Switching (MPAS) wavelength channels should be set up.

For these and other future types of networks an obvious model, the graphs are widely used; and the common representation of virtual paths applied by the above technologies is the graph theoretical term path. This model has the advantage that several earlier graph theoretical methods can be applied in telecommunication networks. The method of paths selection (Label Switched Paths or Virtual Paths or λ paths) is one of the interesting questions in traffic engineering. Paths can be either static when the connection is permanently alive or dynamic if it is built up on-demand. Static paths are efficient in case of constant bit-rate while dynamic paths in case of bursty traffic. The task of the network manager is to set up connections, keep track
of them, and taking them down. The network management is either centralized in which case it is realized by a network management center or some functions are distributed. In the latter case every network element has a module that manages this element. These modules communicate with the network management center. To each network element to be managed belongs a management information base that contains the variables representing the information that are monitored and controlled by the network element manager, e.g., the type of the fiber, the maximal bit-rate and the type of protection.

Algorithms of graph theory are used in several areas of network design and management. In case of establishment of a point-to-point connection shortest path or minimal cost flow is searched depending on whether the connection can split. There exist more sophisticated methods to protect networks against failures, which determine two or more edge- or vertex-disjoint paths. A spanning-tree of minimum cost is required in case of establishing a point-to-multipoint connection and in most cases access networks are trees as well. The wavelength assignment problem in Wavelength Division Multiplexing (WDM) networks and the frequency assignment problem in mobile networks can be handled by graph coloring algorithms.

1.1.1 Fundamentals

A graph is an ordered pair \( G = (V, E) \) where \( V \) is a nonempty set and \( E \) is a set of pairs composed of the elements of \( V \). The elements of \( V \) are called vertices and the elements of \( E \) edges. Two edges of a graph are adjacent if they have a common vertex and two vertices are adjacent if they are connected by an edge. The number of incident edges of a vertex is the degree of the vertex.

A sequence \( (v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k) \) is a path if \( e_i \) is an edge connecting \( v_{i-1} \) and \( v_i \) and all vertices are different. If the vertices are all different except \( v_0 = v_k \) then it is a circuit in the graph. If there exists a path between each pair of vertices then the graph is connected.

Networks are usually modeled by directed graphs (e.g., if the link capacity from node \( A \) to node \( B \) differs from the capacity from \( B \) to \( A \)). The edges of directed graphs are ordered pairs of the form \( (v_1, v_2) \) instead of unordered pairs of the form \( \{v_1, v_2\} \). The vertex \( v_1 \) is the tail while \( v_2 \) is the head of such an edge \( (v_1, v_2) \).

The definition of directed path and directed circuit are analogous to those of path and circuit. A directed graph is strongly connected if there is a directed path from every vertex to every other vertex.

In many applications numbers are associated with the edges or vertices of the graph. These numbers represent for example the physical length of, or delay over a link, or the cost of using the link, or the delay or cost of the equipment. In this case the graph is called edge-weighted or vertex-weighted graph.

A graph is sometimes stored as a matrix. The adjacency matrix - indexed by two vertices of the graph - contains ones (or in case of weighted graphs the weight of the edge) if the two vertices are connected by an edge and zeroes otherwise. In case of directed graphs the direction of the edge is given by the sign of the numbers. In the incidence matrix - indexed by a vertex and an edge of the graph - the non-zero elements indicate that the edge is incident to the vertex. The directions of the edges are given by signs again.
1.1.2 Paths

The connection between two points of the networks can be of two types according to the technology: Either the connection should be realized along a single path (described in this section) or the connection can split (Figure 1.1). The latter case is discussed in the next section.

![Diagram of paths]

Figure 1.1: The connection $s$-$t$ is realized along a single path or can be split

In most cases one must find a minimum weight (shortest) path, i.e., the sum of the weights of edges are to be minimized along the path. The best known shortest path algorithm is due to Dijkstra, which is applicable both in undirected and in directed graphs and yields the distance between a given point and all the other points, if the weights are positive numbers. Negative weights are not permitted in Dijkstra’s algorithm but the algorithm of Ford allows negative weights as well, but only for directed graphs. In this case directed circuits with negative total weights are not permitted. The algorithm of Floyd determines the distance in such a graph from any vertex to any vertex. Two paths are edge-disjoint if they do not have any common edge and are vertex-disjoint if they do not have any common vertex (except the end-vertices). Obviously, vertex-disjoint paths are edge-disjoint as well.

In survivable networks two or more paths are determined for the connections, which are edge- or vertex-disjoint depending on the type of protection. There is a modification of Dijkstra’s algorithm (also known as Suurballe’s algorithm [12]) that finds two or more edge- or vertex-disjoint paths of minimum total weight in an undirected or directed graph with positive weights, if they exist. A $k$-connected graph has at least $k+1$ vertices and after deleting any set of less than $k$ vertices the graph remains connected. In case of a $k$-edge-connected graph after deleting any set of less than $k$ edges the graph remains connected. The graph is $k$-connected if and only if it has at least $k+1$ vertices and any two distinct vertices are connected by $k$ vertex-disjoint paths; similarly, it is $k$-edge-connected if and only if any two distinct vertices are connected by $k$ edge-disjoint paths. In a directed graph the highest possible number of pairwise edge-disjoint directed paths equals the minimum number of edges intersecting all directed $s$-$t$ paths. Similar statement holds for the highest possible number of pairwise vertex-disjoint directed paths if the graph does not contain an $(s,t)$ edge.

1.1.3 Flows and Cuts

Let $G$ be a directed graph. Associate with each edge $e$ a non-negative number, capacity, denoted by $c(e)$. Designate two vertices $s$ and $t$ in $G$, called source and sink, respectively.

The capacity of an edge usually corresponds to the highest bit-rate (transmission capacity) of a link. Suppose that an amount of data ($m_j$) is to be transmitted from $s$ to $t$.

Let $f(e)$ be the amount of data flowing over edge $e$. The function $f$ is feasible if for each edge
\( f(e) \leq c(e) \) and for each vertex the flow conservation constraint holds, i.e., the sum of \( f \) values on the out-going edges equals the sum on the in-coming edges, except for the vertices \( s \) and \( t \) where the sum of the values of the out-going and the in-coming edges should be \( m_f \), respectively. In this case the function \( f \) is called flow and \( m_f \) is called the value of the flow. An edge is called

- saturated if \( f(e)=c(e) \) and
- unsaturated if \( f(e)<c(e) \).

The maximum value flow can be determined by the following algorithm. Suppose that there is a directed path from \( s \) to \( t \) so that for every edge on the path the flow value is smaller than the capacity, in other words, every edge of the path is unsaturated. Along this path the flow value of every edge can be increased until at least one edge will be saturated. The value of the flow can also be increased if the flow is decreased on an oppositely oriented directed edge. Such a path is an augmenting path. A theorem states that the value of a flow is maximal if and only if there is no augmenting path from \( s \) to \( t \).

Divide the vertex set \( V \) of the graph into two subsets: let one set be \( X \) and the other one \( V-X \). Set \( X \) is assumed to include vertex \( s \) and set \( V-X \) is assumed to include vertex \( t \). An \((s,t)\)-cut of a flow is the set of edges having one end-vertex in \( X \) and the other in \( V-X \). The value of the cut is the sum of edge capacities on the edges pointing from a vertex in \( X \) to a vertex in \( V-X \), i.e., only the "forward-edges" influence the value of the flow. The theorem of Ford and Fulkerson says that the value of the maximum flow equals the value of the minimum cut. The above-described method of augmenting paths finds both the maximum flow and the minimum cut if always the shortest possible augmenting path is chosen. If the capacities are integers, the maximum flow will be integer and it can also be realized so that each edge will have integer flow value. In some applications numbers are associated not only with the edges but with the vertices as well, bounding the flow on the vertices from above. This problem can be reduced to the previous one in the following way. Substitute each vertex \( v \) of capacity \( c(v) \) with two vertices \( v' \) and \( v'' \). The new head of all edges pointing to \( v \) should be \( v' \) and the new tail of all edges leaving \( v \) should be \( v'' \). Moreover, add a new edge of capacity \( c(v) \) with tail \( v' \) and head \( v'' \) (Figure 1.2). The capacity of the new edge \((v', v'')\) expresses the capacity of vertex \( v \).

![Figure 1.2: "Vertex splitting": Substitute a vertex of capacity \( c(v) \) with an edge of capacity \( c(v) \)](image)

If undirected edges are permitted then they can be substituted by two oppositely oriented directed edges, that means two directed edges \((u,v)\) and \((v,u)\) of capacity \( c \) in place of a directed edge \((u,v)\) of capacity \( c \). Up to this point, we assumed one type of data flow (i.e., one commodity), which is called single commodity flow. However, in telecommunication networks several connections are established which have to be distinguished (in general by their source and sink). These flow problems cannot be solved one by one because sharing of the common edge capacities binds the different commodities together, i.e., the sum of the flow values on each edge should not be greater than the capacity of the given edge. This is the so-called multi-commodity flow that is often only approximately solvable.
1.1.4 Complexity of Algorithms

Most of the described algorithms (shortest path, maximum flow or minimum cut) are useful also in practice: the number of steps is in the worst case upper bounded by a (generally low degree) polynomial of the input size. Thus, an implementation yields optimal solution in reasonable time even in case of a graph with thousands of vertices. It can be decided similarly in polynomial time whether a graph is connected or 2-connected; whether a directed graph is strongly connected; whether a graph is a planar graph.

On the other hand, there are many problems for which there is no known exact, polynomial-time algorithm. Among the aforementioned problems, the longest path, the maximum cut, or the general solution of the minimal cost multi-commodity flow have this property. These problems belong to the class of NP-hard [20] problems: if somebody would be able to find a polynomial time algorithm for any of them then by calling that algorithm as a subroutine it would provide a polynomial-time algorithm for every other NP-hard problem. The running time of an NP-hard problem can be very long - depending on the size of the network and the type of the problem. Solutions of this type of problems, even those which will be discovered in the future, will most probably have at least one of the following properties:

- either the number of steps will not be a polynomial of the size of the input,
- or they will not find the optimal solution, just an approximate one,
- or a theoretically different model of calculation is needed (e.g., an algorithm requiring the generation of random numbers).

However, it is to emphasize that

- if the size of the input is not too large then this distinction is not critical;
- an NP-hard problem may have special cases that can be solved in polynomial time (e.g., finding the longest directed path in graphs without directed circuits);
- there are problems for which no polynomial-time algorithm is known but - to our best knowledge - they do not belong to the class of NP-hard problems either (e.g., deciding whether two graphs are isomorphic).

For further introduction to graph theory we refer to books [16] or [17] and the problem collection [18], while [19] is more application-oriented. Algorithms in graph theory are deeply covered in [20]. The handbook [21] is a must for active researchers in the field.

1.2 Resilience Methods

Most of the algorithms proposed in this Dissertation include the protection against network element failures. The network should be able to react as fast as possible to any link or node failure to cut down loss of data. There are different protection and restoration techniques which can be classified according to different criteria [C2, 15].

Protection or Restoration. Protection makes use of pre-assigned capacity between nodes, i.e., for each node-pair a working and one or more protection paths are assigned and no changes take effect. If the backup paths are determined after the failure (in any available free capacity), the method is called restoration. The second one is the slower one, it needs more processing, but it does not allocate resources in advance.
**Diverse Protection** means that the working and protection paths are using diverse, i.e., independent paths. This ensures that no single cable break can affect both the working and the protection paths.

**Automatic Protection Switching (APS)** automatically reroutes signals from working to the protection line in case when a failure affects any part of the working path. This is the simplest and fastest resilience technique. This method is particularly efficient when used with Diverse Protection.

**Dedicated or Shared Protection.** When using Dedicated Protection (DP), each working path has one standby path (protection path) with dedicated (reserved) resources. In case of Shared Protection (SP) protection paths share resources allocated for protection. The advantage of DP is its simplicity and fast operation. Its drawback is its very high resource usage, especially when the network has to be able to survive multiple \( n \) failures. In this case up to \( n + 1 \) independent paths with allocated capacities should be dedicated to each node-pair. The advantage of SP is the ability to survive even multiple failures while the resource usage is moderate. These questions will be further discussed in this work.

Instead of classification to dedicated and shared the terminology “1 + 1”, “1 : 1”, “1 : \( m \) : \( n \)” and “\( m : n \)” is also used. 1 + 1 protection is when all data is sent on two paths simultaneously, 1 : 1 is the DP while 1 : \( n \) and \( m : n \) are SP techniques where 1 and \( m \) protection facilities are available for \( n \) working facilities respectively.

**Link Protection or Path Protection.** The two main groups of protection are Link Protection and Path Protection. The first one restores paths affected by a link failure by over-bridging the failed link. In this case the load of the network can be quite high as well as the latencies. Path protection restores paths between the two end-points. When restoration is used it can be time consuming while with protection strategy large alternative path tables are needed (for each link of each path there should be a protection path). The simplest method for APS is Path Protection with link-disjoint routes. It has good performance for a single failure and it is simple to apply. The problem is how to choose optimally the working and protection paths.

### 1.2.1 A Simple Example

Consider the network model example in Figure 1.3(a). In order to establish a connection a shortest path is searched between two routers \( R_1 \) and \( R_2 \) (Figure 1.3(b)). This traffic can be protected in two ways:

- by **path protection**, when the traffic is re-routed between source and destination avoiding the failed link. A special case of path protection is when working and protection paths are disjoint (Figure 1.3(c)),

- or by **link protection**, when just the traffic of the failed link is to be rerouted (Figure 1.3(d)).
Figure 1.3: (b) Graph model of a network, (b) establishment of a connection, (c) path protection, (d) link protection
Chapter 2

Configuration of Infocommunication Networks

In this Chapter several methods are developed for configuring the Virtual Path (VP) system of ATM or Label Switched Paths (LSPs) of MPLS networks. In Section 2.1 we consider the case without protection while in 2.2 the sessions are protected. For the latter case a new optimization problem is proposed in Section 2.3, the “asymmetrically weighted” pair of disjoint paths. In Section 2.4 availability is considered for characterization of telecommunication networks.

2.1 Static LSP Routing Algorithms for MPLS Networks

The ever-growing need to transfer information requires networks that carry huge amount of data with very small latency and losses. Since the Internet Protocol (IP) is the most widespread application, MultiProtocol Label Switching (MPLS) [J6, 3, 22] has been worked out to satisfy the above requirements in a simple way. MPLS Networks relay on sending data packets over Label Switched Paths (LSPs) which are established by setting up forwarding tables within Label Switching Routers (LSRs) using a Label Distribution Protocol (e.g., LDP). The method for choosing LSPs to be set up by LDPs is not yet standardised, and it is one of the interesting open questions of Traffic Engineering. It can be either static, which is the simplest and most efficient way in homogeneously heavily loaded networks, or dynamic, using any of known routing protocols, e.g., OSPF. In this Section two types of centralised algorithms for configuration of the system of LSPs are proposed. First a combinatorial approach is described, then a randomised heuristic method. The methods are particularly useful for determining the Strict Explicit Routes of MPLS “clouds” of a larger network. The obtained results can be applied for configuring the system of virtual paths in networks like SDH/SONET, ATM or WDM as well.

Our approach combines theoretical results with randomised heuristics, and gives a fast algorithm, that proved to be very efficient in practice for realistic sparse networks. The solution provides a bound useful for estimating the quality of the routing, and our computations show that the results are very close to the optimum [C3].

2.1.1 The Mathematical Problem

Flows can model a large variety of network problems, and have been studied extensively. Many fast algorithms exist to solve the maximal flow problem, or the multi-commodity flow
problem with or without costs. If we restrict the flow in a way that all commodities should be transferred along a single path without branching we get the mathematical model of static routing, the Unsplittable Flow Problem. This problem like many problems in network design is NP-complete.

Given such a complex problem one may choose to use general purpose optimisation tools, like tabu search, genetic algorithms, stochastic methods, one may find “common sense” heuristics, or try to understand the underlying structures of the problem to get a solution that has guarantees. Unfortunately general purpose tools do not “understand” the problem, heuristics may get stuck in local optima, and theory is not strong enough to provide solution for the general case. Our experience shows that combining the three methods gives the best results.

2.1.2 The Unsplittable Flow Problem

We are given a directed graph $G$ with vertex set $V$, edge set $E$, and with capacities on the edges $c$. We have a set of demands $D$ for each node pair in the form of $(s_k, t_k, d_k)$ where $s_k$ is the source of the demand, $t_k$ is the terminal of the demand and $d_k$ is the demand size. For all demands our task is to route $d_k$ units of flow along a single path between the source and the terminal, such that the total flow does not violate the edge capacities.

This problem is approximately solvable under restrictions. In [23] Dinitz, Garg and Goemans assume that the sources of all demands are at the same node of the graph. This version is called the Single-Source Unsplittable Flow Problem. They present a polynomial time algorithm (referred to as DGG further on) that finds a routing with the following property: If there exists a fractional flow satisfying all demands, the DGG algorithm finds a routing, such that every edge capacity is violated by at most the maximum demand.

This result is particularly good, when the edge capacities are large relative to the demands. However, to solve the original problem we have to face two difficulties: their result is not feasible in the sense, that it may violate edge capacities, and we have to solve the problem for general demands, instead of single-source demands.

2.1.3 The Combinatorial Approach (CA)

The main idea is to collect the demands whose source is at a given node $v$, calculate edge capacities for this part of the problem, then solve the single source problem using the DGG algorithm. In a post processing phase the obtained routing is improved and demands that violate edge capacities are deleted. Our algorithm referred to as CA works as follows:

- **Step 1.** Given the graph, the demands and the edge capacities, we find a minimal cost (possibly fractional) multicommodity flow, where the cost function is 1 for all edges.

- **Step 2.** For all nodes:
  - We collect the demands, whose source is the given node.
  - For these demands we sum the fractional flow we got for them in Step 1.
  - We execute the DGG algorithm for these demands and use the fractional flow sum as capacity.

- **Step 3.** We delete demands that violate edge capacities.
Infeasible Demand Deletion

The total edge capacity violation was chosen to measure the badness of the routing. We call an edge overloaded if the flow through the edge is bigger than its capacity; the excess of an overloaded edge is its load minus its capacity. So the badness of a routing is the sum of the excess on overloaded edges. (For a feasible routing the badness value is 0.) A very simple greedy rule was used in Step 3 to delete demands that caused edge capacity violations. For all demands we calculate the improvement in badness coming from the deletion of the demand. The demand that has the biggest badness improvement / demand size value is deleted, until a feasible routing results.

Routing Improvement Strategies

Two strategies were used to improve the result of CA. One was SA++ described in Section 3, which used the output of CA as a starting point. Beside SA++ a second heuristic algorithm has been worked out as follows. CA assigns a path to each demand. Then we apply a local greedy strategy, which reroutes some demands if this would decrease the badness value. To allow escape from local minimum, the algorithm is randomised. Using CA with the following improvement strategy is algorithm CA++.

- Iterate the following:
- For the demands whose current path uses an overloaded edge:
- Free the resources used by the path of the demand
- Define a cost function for all edges: the cost of an edge is the excess of the edge if the demand would use it, perturbed by a small random value.
- Find a shortest path from the source to the terminal regarding this cost function
- Use this path as the new path for the demand

2.1.4 Original and Improved Simulated Allocation

Our second approach uses Simulated Allocation, a stochastic method by Picon. The main idea behind SA [5] is a randomised alternative resource allocation and deallocation of the traffic demands. Each step of the algorithm can be determined by two possible actions:

- **allocation** of one or more demands to a path with probability \( q \).
- **deallocation** of one or more demands with probability \( 1-q \).

If allocation is chosen then a shortest path is to be found within the free capacities. If deallocation is chosen then the allocated resource for a commodity is freed. The algorithm succeeds if every commodity is allocated. This approach allows escape from local optimum but increases running time while reaching the globally optimal solution is not guaranteed.

Improved Simulated Allocation

The simple principle of this method can be varied in many ways.

1. How to choose the length function \( l \) of links for the shortest path subroutine?

   Let \( f_e \) denote the current amount of free space on link \( e \) and \( d_k \) the size of the demand that is chosen to be allocated. Links where \( f_e < d_k \) are not considered. For other links let
   
   \[
   \begin{align*}
   (A) \quad & l_e = 1 & \quad (B) \quad & l_e = 1/f_e & \quad (C) \quad & l_e = 1/(f_e - d_k)
   \end{align*}
   \]
Length functions (B) and (C) are better for tight networks, while the simple length function (A) gives better results for loose networks, because it finds the physically shortest path for all demands, minimising the network utilisation.

2. How to choose the allocation probability $q$?
   
   (A) Set the allocation probability to constant between 0.5 and 1, e.g., 0.51, 0.7, 0.9, 0.99.
   
   (B) Always allocate, until an allocation procedure fails, then deallocate some commodity.
   
   (C) Let $S_k$ and $U_k$ denote the number of successful and unsuccessful allocations after the $k^{th}$ step. Set in each iteration $q = S_k / (S_k + U_k)$.
   
   (D) To avoid that $q$ is influenced by "very old" values, consider only the last $C$ iterations, where $C$ is the number of commodities that are to allocate. Let $S_c$ denote the number of successful allocations in the last $C$ iterations. Let $q = S_c / C$.
   
   If the allocation probability is increased, the results improve, but there is a point where unsuccessful allocations appear. The compromise where results are good enough and allocations are successful even on tight networks are between 0.8 and 0.9.

3. In case of a deallocation one or more commodities can be deallocated:
   
   (A) Deallocate always one commodity.
   
   (B) Dealocate always a constant number of commodities.
   
   (C) Dealocate all commodities from an overloaded link.
   
   (D) In case of deallocation, deallocate half of the commodities from the most overloaded link and 10 percent of the commodities from other links.
   
   On loose networks only few deallocations occur, all strategies give similar results. Variation A is very fast, but it can easily get stuck in a bad state, from which it can hardly escape. In these cases the running time increases substantially. Variation D seems to be the best choice for both loose and tight networks.

4. When is it time to stop the algorithm? If the number of (A) unsuccessful allocations or (B) deallocations exceeds a constant, e.g., limit times the number of commodities, where limit = 1, 2, 10, etc. In most cases the stop barrier does not influence anything since the algorithm succeeds before the stop barrier is reached. In tight networks the stop barrier is suggested to be set to a large number, e.g., three times the number of commodities.

The simple Simulated Allocation (SA) is when version A is taken for all options, that means link costs are set to 1, the allocation probability is set to the constant value 0.8, always deallocate one commodity, and set the stop barrier equal to the number of commodities.

The advanced Simulated Allocation (SA++) is when link length is set to the inverse of the free space on the given link (1.B). Always allocate, until an allocation procedure fails (2.B), then deallocate half of the commodities from the most overloaded, and 10 percent of the commodities from other links (3.D). SA++ is also improved by sorting the traffic demands according to decreasing bandwidth demands, i.e., the node-pairs with larger demands are routed first. The allocation probability is set to three times the number of commodities.

**Time Complexity**

The time complexity of the Combinatorial Approach (Step 2, Step 3) is $O(|V|^2 |E| + |D||E|)$, where $|V|$, $|E|$ and $|D|$ stand for the number of vertices, number of edges and number of demands. The running time of the heuristic improvement strategies depends on the number of iterations. In SA the number of allocation steps can be estimated by $|D|/(2q-1)$ if no allocation is rejected.
However, this is the case only in trivial solvable networks, while in practice the running time may tend to infinite. The time complexity of Dijkstra’s algorithm is \( O(|V|^2) \) [10]. Since after each deallocation at most \( |D| \) demands can be allocated and the number of deallocations is bounded by \( O(D) \), the worst case time complexity of SA ++ is \( O(|V|^2 |D|^2) \).

### 2.1.5 Numerical Results

We have implemented our algorithms in C++ using Kurt Mehlhorn and Stefan Näher’s LEDA library [25]. For finding the fractional flow in the first step of the CA algorithm, we used Jordi Castro’s PPRN library [26] that is capable to find a minimal cost multicommodity flow. In some cases it may be practical to use an approximation algorithm to solve the multicommodity flow problem, in order to speed up the algorithm, since most of the running time is spent in Step 1 of CA finding the fractional solution (88\% of running time in the case of the 60-node network).

All computations were carried out on an Intel Pentium 400 MHz processor running Linux. Our main objective was to route as much demand as possible while minimizing the edge load. The test files were generated by a network design program described in [24]. These networks were planar and sparse and we also tested dense graphs. Given the number of vertices and the demands, that were randomly sized, the network design program created a network and a routing for all the demands. We took the network and the capacities used by the resulting routing and routed the demands using our algorithm. In this sense the capacities were tight.

We have compared our results from three aspects: proportion of allocated demand, speed and economical usage of link capacities. For reference the algorithm SP is included in the results, which simply finds the shortest path for all demands, until it gets stuck.

Our computational experiences can be seen in Table 2.1: "Nodes" denotes the number of vertices in the test networks (‘d’ means the dense network, the other cases are the sparse networks.) For all methods and instances the percentage of routed demands and the running time of the algorithm is given in seconds.

The edge capacity usage of the different algorithms for the 60-node sparse network is presented in Table 2.2 in row “Load”. Please note that if an algorithm routes more demand, it naturally uses more edge capacity, that is the reason why the percentage of routed demands is also included in row “Routed”. The used link capacity relative to the routed demand is in row “L/R”, the smaller value means the more economical algorithm.

<table>
<thead>
<tr>
<th>Nodes</th>
<th>SP</th>
<th>SA</th>
<th>SA ++</th>
<th>CA</th>
<th>CA ++</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>97.7%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>0.01s</td>
<td></td>
<td>0.2s</td>
<td>0.1s</td>
<td>0.1s</td>
</tr>
<tr>
<td>30</td>
<td>83.0%</td>
<td>95.0%</td>
<td>96.7%</td>
<td>99.47%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>0.2s</td>
<td>34.6s</td>
<td>15.0s</td>
<td>1.0s</td>
<td>9.6s</td>
</tr>
<tr>
<td>60</td>
<td>74.0%</td>
<td>97.2%</td>
<td>97.8%</td>
<td>99.6%</td>
<td>99.9%</td>
</tr>
<tr>
<td></td>
<td>1.9s</td>
<td>10668s</td>
<td>715.0s</td>
<td>44.0s</td>
<td>267.0s</td>
</tr>
<tr>
<td>100</td>
<td>65.1%</td>
<td>96.4%</td>
<td>97.4%</td>
<td>99.8%</td>
<td>99.9%</td>
</tr>
<tr>
<td></td>
<td>10.2s</td>
<td>13044s</td>
<td>8749s</td>
<td>2183s</td>
<td>4535s</td>
</tr>
<tr>
<td>90d</td>
<td>62.6%</td>
<td>83.0%</td>
<td>94.5%</td>
<td>80.3%</td>
<td>94.5%</td>
</tr>
<tr>
<td></td>
<td>2.0s</td>
<td>155.0s</td>
<td>219.0s</td>
<td>80.3s</td>
<td>299.3s</td>
</tr>
</tbody>
</table>

Table 2.1: Percentage of allocated demands and running times of the algorithms
<table>
<thead>
<tr>
<th></th>
<th>SP</th>
<th>SA</th>
<th>SA++</th>
<th>CA</th>
<th>CA++</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load</td>
<td>90.6%</td>
<td>97.9%</td>
<td>98.1%</td>
<td>97.1%</td>
<td>99.6%</td>
</tr>
<tr>
<td>Routed</td>
<td>74.0%</td>
<td>97.2%</td>
<td>97.8%</td>
<td>99.6%</td>
<td>99.9%</td>
</tr>
<tr>
<td>L/R</td>
<td>122.4%</td>
<td>100.7%</td>
<td>100.3%</td>
<td>97.5%</td>
<td>99.7%</td>
</tr>
</tbody>
</table>

Table 2.2: Usage of link capacities

For the sparse networks the proposed CA++ had the best performance, it could route almost all (99.9%) demands, while CA proved to be the fastest and most economical in link capacity usage. In the case of the dense networks further research is needed to obtain results closer to optimum.

2.1.6 Summary

We presented a heuristic algorithm for solving the Unsplittable Flow Problem in general graphs. Our approach proved to be fast and effective in practice for sparse networks, and comes with a guarantee for the cost of the routing. Our solution outperforms the Simulated Allocation, and Shortest Path algorithms both in the quality of the result and in running time. The proposed method configures the LSP system of MPLS Networks and can be generalised for the configuration of SDH, ATM and WDM networks as well. In the next section we extend the model with dedicated and shared protection and develop algorithms for fault tolerant infocommunication networks.

2.2 Network Configuration with Dedicated and Shared Protection

An efficient, network-flow-theory based model with an algorithm is proposed for configuration of WDM, SDH, ATM, and MPLS networks. The configuration results in a structure of paths which can survive any single link or node failure. For each node-pair two link- or node-disjoint paths are obtained: A working one and one used for backup when a failure affects the working one. The subject of minimisation is the length of working paths, and the total capacity used for both, working and backup paths. We compare results obtained by three methods: First, the problem has been formulated as an Integer Linear Program (ILP) and solved by CPLEX. Second, a heuristic method called Iterative Capacity Splitting (ICS) has been worked out. Third, the randomised method called Simulated Allocation (SA) introduced in Section 2.1.4 has been implemented. For minimising the total amount of allocated capacity we propose a method referred to as TCA for reducing the capacity allocation, which can be applied to all three methods. The significance of the TCA method is that around 40% of the total capacity can be saved! The obtained results can be used in the resource management system of SDH, ATM, WDM and other networks.

In Subsection 2.2.1 the problem is formulated which is going to be solved. Subsections 2.2.2 and 2.2.3 present methods for solving the problem for Dedicated and Shared Protection, respectively. Section 2.2.4 deals with heuristics for improving the performance of the algorithms. Section 2.2.5 evaluates the obtained numerical results [C2].
2.2.1 Problem Formulation

The network topology with link capacities and the traffic matrix containing the traffic demands are given. The task is to find two paths for each node-pair. The first of two paths should be the shortest possible (working) while the second one should be the shortest one among those which are independent of the first one. The total system of paths must fit into the link capacities and has to be optimal in a sense of utilising as few resources as possible.

In many cases a network operator's aim is not to reduce the total utilisation but to reduce the working utilisation and not to pay attention on the protection utilisation at all. Or in the objective the working utilisation is of some constant (e.g. 3) times more important than the protection utilisation. Consequently, the task can be more complicated if the two disjoint paths has different weights along the paths (see Section 2.3).

This problem can be formulated mathematically using graph theory and theory of network flows as follows [10, 27]. In graph theory 'edge', 'vertex' and 'commodity' correspond to 'link', 'node' and 'traffic demand' in telecommunication networks, respectively. The analogy for 'diverse' is 'disjoint' in graph theory. Two paths are edge-disjoint when they do not have any common edge. Two paths are vertex-disjoint when they do not have any common vertex except the source and sink vertices. The last definition is stronger - since not having common vertex excludes having a common edge1.

Thus, the task is to solve optimally a capacitated minimal cost unsplittable multi-commodity flow (MCMCF) problem where for each commodity two paths are to be determined. Unsplittable means that branching of flows is prohibited along the paths. This problem belongs to the class of NP-hard problems [28].

Each link has been assigned a capacity which is divided among allocated and spare capacities. The allocated capacity is further divided into two parts one is allocated to working paths the other is allocated to restoration paths. Minimising the allocated capacity the network will be able to accept and serve further traffic demands, while minimising the capacity allocated to restoration paths we minimise the amount of spoil capacities keeping more place for profitable traffic.

When solving the problem automatic protection switching (APS) with diverse paths seems to be the most efficient technique. This is the Path Restoration with Link-disjoint Routes. It is static and distributed. It is the fastest and simplest to implement when failure occurred, but quite complex to configure the system of paths optimally in advance.

2.2.2 Alternative Methods for Dedicated Protection

This problem is discussed by both mathematicians and telecommunications researchers [10, 29, 30, 11, 31, 4]. Mathematicians deal mostly with approximations for solving the MCMCF problem and the problem of disjoint paths. Telecommunications engineers deal mostly with approximations which lead to practically acceptable results in considerably short time sometimes even without being able to prove convergence property of algorithms and to bound their complexity. This problem belongs to the class of NP-hard problems. The NP-hardness indicates that it can not be solved optimally in polynomial time for larger networks, and that is why heuristics are looked for.

There are many papers dealing with similar topics see, e.g., [32, 13, 33, 34, 35, 12, C2]. First, the problem is formulated as an ILP task, then the proposed method is explained.

1exception is when the source and sink are connected by a direct edge
Notations

To model the problem we used the following variables. Input variables are: $C_l$, the capacity of link $l$; while $c^o$ is the capacity demand of commodity $o$. The outputs, $x^o_i$ and $y^o_i$ are flow indicator variables. $x^o_i, y^o_i \in \{0, 1\}$ indicate working/backup flows of commodity $o$ over links $l$. $X_l$ is the amount of capacity to be allocated for working flows on link $l$. $Y_l$ is the amount of capacity to be allocated for backup flows on link $l$.

Integer Linear Programming (ILP)

The problem can be formulated as an Integer Programming (IP) task [36]. The formulation is based on that one for the minimal cost multi-commodity flow (MCMCF) problem [27]. The difference is that for each demand two commodities are defined: one is the working one while the other is for the protection and there is an additional constraint for each demand which ensures path disjointness. IP has to be used instead of LP to avoid flow branching. For solving this problem any commercially available package can be used, e.g., CPLEX, using, e.g., branch-and-bound algorithm. However, these IP solvers are efficient only when the problem is linear! Therefore the objective and the constraints must be linear functions of variables.

Objective:

$$\text{minimise} \sum_{l \in L} \sum_{o \in O} (x^o_i + y^o_i)c^o$$  \hspace{1cm} (2.1)

Subject to constraints:

$$\sum_{o \in O} (x^o_i + y^o_i)c^o \leq C_l \text{ for all links } l \in L$$  \hspace{1cm} (2.2)

$$\sum_{j=1}^{N} x^o_{ij} - \sum_{k=1}^{N} x^o_{ki} = \begin{cases} 
0 & \text{if } i \neq \text{ source of } o \land i \neq \text{ sink of } o \\
1 & \text{if } i = \text{ source of } o \\
-1 & \text{if } i = \text{ sink of } o 
\end{cases} \hspace{1cm} (2.3)$$

$$\sum_{j=1}^{N} y^o_{ij} - \sum_{k=1}^{N} y^o_{ki} = \begin{cases} 
0 & \text{if } i \neq \text{ source of } o \land i \neq \text{ sink of } o \\
1 & \text{if } i = \text{ source of } o \\
-1 & \text{if } i = \text{ sink of } o 
\end{cases}\text{ for all nodes } i \text{ and commodities } o$$

$$x^o_i \in \{0, 1\}, y^o_i \in \{0, 1\}, \text{ for all links } l \in L \text{ and commodities } o \in O$$ \hspace{1cm} (2.4)

$$x^o_i + y^o_i \leq 1 \text{ for all links } l \in L \text{ and commodities } o \in O$$ \hspace{1cm} (2.5)

According to constraint (2.5) which ensures link-disjoint paths the working and backup paths of the same commodity may not use the same link. For making the working and backup paths node-disjoint the following constraint should be used instead of (2.5): $\sum_{j=1}^{N} (x^o_{ij} + y^o_{ij}) \leq 1$ if nodes $i$ and $j$ are neither source nor sink of commodity $o$, for all nodes $i$ and commodities $o \in O$.

Objective (1) express that the total utilization of the network should be minimised. The following Objective forces working paths to be shorter than the backup one: **minimise**

$$\sum_{l \in L} \sum_{o \in O} (\alpha \cdot x^o_i + y^o_i)c^o$$ Increasing the weight of $\alpha > 1$ ($\alpha = 3$ has been used) will make working paths even shorter on average comparing to the backup paths.
If one would like to increase the probability of finding path for a new demand, free space on
the links should be distributed smooth. It can be ensured, e.g., by minimising the sum of square
of link utilizations: \( \min \sum_{\ell \in \mathcal{L}} \left( \sum_{e \in \mathcal{E}} \alpha \cdot x^e_{\ell} + y^e_{\ell} \right)^2 \).

**The Proposed Method: Iterative Capacity Splitting (ICS)**

We propose here a method for solving the formulated problem. The network is modelled in
standard way as either a directed or undirected graph. The traffic demands are the commodities.
The algorithm consists of 3 main phases:

1. In the first phase one path is determined for each demand solving the unsplittable MCMCF
   problem. In Section 2.1 and in [3] two types of such algorithms are proposed: a combinatorial
   approach and a randomised heuristic method. These methods combine theoretical results with
   randomised heuristics, and give results that are proved to be very efficient in practice. The
   solution provides a bound useful for estimating the quality of the routing, and our computations
   show that the results are very close to the optimum.

   The other possibility is to formulate this phase by ILP. The formulation is very similar to the
   above one: merely the variables and equalities for the protection paths are to be omitted. And
   there are other combinatorial algorithms which give results very close to the best possible and
   run by 1-2 orders of magnitude faster than the method which uses linear programming.

2. In the second phase we determine the secondary path which is independent of the first one,
   i.e., disjoint. The easiest way of completing this task is first to delete temporarily the working
   path, second to find all links which do not have enough capacity to accommodate the traffic of
   the considered commodity and to delete them also temporarily. Third, to use a shortest path
   algorithm for finding the backup path, and then to allocate capacities and restore the deleted
   edges. The secondary paths can be searched by a single commodity flow as well.

   For simultaneous determination of two diverse paths method proposed by Sunurballe [12] is
   preferred because it is able to find a pair of shortest paths even in those situations one the above
   described methods fail to. This method gives better results in shorter time, therefore in our
   numerical investigations this method has been intensively used.

   In the second phase the commodities are allocated one-by-one. Both Simulated Allocation
   and Integer Progrning are easily applicable to the second phase, and increase the number of
   found protection paths. Solving the Integer Programming is very slow, but still faster in two
   phases than simultaneously finding working and protection paths as described.

3. Iteratively repeating phases 1 and 2 capacities are being allocated to working and backup
   paths.

   The Algorithm works as follows.

   - **Phase 0.** First divide the capacity \( C_\ell \) of each link \( \ell \) into two equal parts - one reserved
     for the working paths \( C^w_\ell \) and the other for restoration paths \( C'^w_\ell \). Set the Total Traffic
     counters \( TT^w_\ell \) and \( TT'^w_\ell \) to zero for working and restoration paths respectively on link \( \ell \).
     These counters will sum the capacity allocated for traffic demands on each link.

   - **Phase 1.** Now run an algorithm for solving unsplittable MCMCF problem on the network
     with \( C^w_\ell \) link capacities. If feasible update \( TT^w_\ell \) values and proceed to Phase 3; else increase
     the amount of \( C^w_\ell \) capacities by decreasing \( C'^w_\ell \) capacities \( (C'^w_\ell = C_\ell - C^w_\ell) \) by the same
     amount (by, e.g., 10%: \( C'_\ell = 0.9 \cdot C^w_\ell \)) for all links \( \ell \) and repeat Phase 1.

   - **Phase 2.** Recalculate protection capacities \( (C'^w_\ell = C_\ell - C^w_\ell) \) and find a disjoint path for
     every demand. If all secondary paths fit into the reserved capacity update \( TT'^w_\ell \) and proceed

17
to Phase 3, else for each link \( l \) modify \( C_i^l \) and \( C_i^{nl} \) \( (C_i^l + C_i^{nl} = C_i) \) and goto Phase 1.

- **Phase 3.** Set values of \( C_i^l \) and \( C_i^{nl} \) for all links as follows. \( C_i^l = TT_i^l + (C_i - TT_i^l - TT_i^{nl})/2 \) and \( C_i^{nl} = TT_i^{nl} + (C_i - TT_i^l - TT_i^{nl})/2 \) and repeat from Phase 1 while there are changes in the values of \( C_i^l \) and \( C_i^{nl} \) in recurrent iterations.

The third phase is for improving the performance. It can be omitted if you are looking just for a feasible solution.

To avoid eventual oscillations the convergence can be slowed down by using the following formula for iteration \( k \). \( C_i^l(k) = C_i^l(k - 1) - \alpha \cdot (C_i^l(k - 1) - C_i^l(k))/2 \), where \( C_i^l(k - 1) \) is the \( C_i^l \) value obtained in \( k \)-th iteration, while \( C_i^l(k) \) is the value obtained as described in Phase 3. Now, \( C_i^{nl}(k) \) can be calculated as \( C_i^{nl}(k) = C_i - C_i^l(k) \). Parameter \( \alpha \) is the convergence speed factor and \( 0 < \alpha \leq 1 \). If \( \alpha \) is close to 0 the convergence is very slow while when it is approaching 1 oscillations can appear.

The flow chart of the proposed method can be seen in Figure 2.1.

![Flow Chart of Iterative Capacity Splitting (ICS)](flowchart.png)

**Figure 2.1: Flow Chart of Iterative Capacity Splitting (ICS)**

### 2.2.3 Shared Protection: Thrifty Capacity Allocation (TCA)

In the previous section dedicated protection has been used. This is the easiest way to allocate capacity for backup paths, that is allocating the same amount as for the working paths. The advantage of this approach is its simplicity and that the network employing APS can survive any single failure and very likely also more failures. The disadvantage of this approach is in wasting the capacity.
The probability of having two independent failures in the network at the same time is extremely low. For this reason when configuring the system of restoration paths, we allocate fewer capacity on each link that still guarantees to survive every single failure.

Before explaining this method, two illustrative examples are given. Figure 2.2(a) shows a simple network where two connections are to be set up: two units from node 'a' to 'd' and one from node 'c' to 'f'. Each connection has a working (solid line) and a protection (dotted line) path as shown in Figure 2.2(a). We consider two links which can fail: (a,d) and (c,f). All node-pairs are using paths denoted by solid lines as their working paths. If link (a,d) fails then the backup path of (a,d) [a-b-e-d] should overtake two units. But if (c,f) fails then the backup path of it should overtake the traffic of only one unit. For this reason (b,e) has to ensure enough capacity for the maximum of the two connections, i.e., two units of capacity for protection purposes. In this case link (b,e) has to have capacity for two units, since working paths do not use this link. This is enough capacity for full protection under assumption that only one link can fail at a time, therefore, two units of capacity are enough instead of three. This method of determining less backup capacity is called capacity compression.

![Figure 2.2: Ideas for shared protection: (a) Capacity compression, (b) Capacity reuse](image)

Figure 2.2(b) shows another case where a working path using links (a,b), (b,c) and (c,f) is depicted. Now consider the case when link (a,b) fails. After the failure the protection path is used and the capacity of the whole working path can be used for protection purposes of other demands. Consequently, in case of link failure (a,b) the backup capacity of links (b,c) and (c,f) can be extended by the capacity of this demand. This phenomenon is called capacity reuse.

To capitalize the gain of capacity compress and capacity reuse, the backup capacity on link \( l \) can be determined as follows: Simulating the failure of link \( f \) for each link \( l \), define the following backup factor: \( b_{l,f} \) = sum of capacity of demands whose working path uses link \( f \) and protection path uses link \( l \) minus the sum of capacity of demands whose working path uses both links \( l \) and \( f \). This represents a matrix \( B \) indexed by two links \( (b \) and \( f) \). \( b_{l,f} \) express that at least so many backup capacity is to be allocated on link \( l \) “on the account of” link \( f \). Backup capacity on link \( l \) is the maximum of \( b_{l,f} \).

Applying this method guarantees that the network survives any single link/node failure and gives resource savings around 30%-40% compared to the dedicated protection.

Shared protection can be applied to all three methods. The ILP formulation is described in the next section, and there are four ways of applying to SA or ICS:

1. The whole procedure is taking place at the very end of the algorithm. That means that dedicated protection is used and in the end the backup capacity will be “compressed”.

2. It can be applied after phase 2. That means backup capacity is calculated after each iteration.
3. Backup capacity recalculation takes place after each protection route placement.

4. Backup capacity recalculation takes place before each protection route placement.

In this order the algorithm is increasingly memory consuming but decreasingly time consuming and is better in performance (i.e., resource utilization). Furthermore, in latter cases it is more likely to find a feasible solution when having tight capacity limits, but the computational complexity is significantly higher. Furthermore the later case gives resource savings around 40% while the former one around 30% only.

All four cases can be implemented in a way that the B matrix values and the backup capacity is always recalculated if needed. If doing so, there is no need to store the whole matrix just one row of it, that is why it is not memory consuming. However, it is very time consuming; one recalculation costs $O(|L|^2|C|)$, accordingly in cases 3 and 4 a smarter solution is proposed:

1. Insert working path of commodity $\alpha$: For all used links $l$: $X_l := X_l + c^\alpha$

2. Delete working path of commodity $\alpha$: For all used links $l$: $X_l := X_l - c^\alpha$

3. Insert protection path of commodity $\alpha$: For all linkpairs $(l \neq f)$: (1) If the working path of commodity $\alpha$ uses link $l$ and the protection path of commodity $\alpha$ uses link $f$ then: $b_{l,f} := b_{l,f} + c^\alpha$. (2) If the working path of commodity $\alpha$ uses both links $l$ and $f$ than: $b_{l,f} := b_{l,f} - c^\alpha$.

4. Delete protection path of commodity $\alpha$: For all linkpairs $(l \neq f)$: (1) If the working path of commodity $\alpha$ uses link $l$ and the protection path of commodity $\alpha$ uses link $f$ then: $b_{l,f} := b_{l,f} - c^\alpha$. (2) If the working path of commodity $\alpha$ uses both links $l$ and $f$ than: $b_{l,f} := b_{l,f} + c^\alpha$.

Aside from these an even better solution can be obtained if the following procedure takes place before each protection route placement:

- For each link $l$ examine whether the protection path of demand $\alpha$ can take this link, and enable link $l$ only if the answer is true. This can be done by temporarily modifying matrix $B$ on the basis of the above mentioned 3. method.

Please notice that link $l$ can be enabled even if there is less free capacity on link $l$ than the capacity of demand $\alpha$. This option is worth the additional complexity only if the link capacities are very tight.

ILP Formulation of Applying the TCA

TCA can be incorporated into the ILP formulation. The amount of backup capacity can be expressed by the following expression by taking advantage of the capacity compress and capacity reuse gains:

$$\sum_{o \in O} x^o_{l', f'} c^o - \sum_{o \in O} x^o_{l} x^o_{l'} c^o \leq Y_{l'} \text{ for all } (l', l') \in L^2 \text{ except for } l = l'$$ (2.6)

The left side of the equation can be simplified: $\sum_{o \in O} x^o_{l'} (y^o_{l'} - x^o_{l}) c^o$. Unfortunately, this equation is not linear because it has two quadratic terms, namely $x^o_{l'} y^o_{l'}$ and $x^o_{l'} x^o_{l}$. But it can be linearized in the following method:

Two groups of new variables $\gamma^o_{l,l'} \in \{0,1\}$ and $\delta^o_{l,l'} \in \{0,1\}$ are needed for all pairs of links $(l, l') \in L$ where $l \neq l'$ and commodities $o \in O$. Its purpose is to substitute the logical AND
operation of variables in the expressions $x_{i}^{o} = \gamma_{i}^{o}$ and $x_{i}^{o} = \delta_{i}^{o}$ in Equation 2.6. For this purpose two times four equations are required for all pairs of links and commodities, i.e., $8 \cdot (|L|^2 \cdot |O|)$ equations. These equations ensure the above condition according to Boole-algebra.

Now the problem can be formulated as an Integer Linear Programming (ILP) task, and can be solved by an ILP software, e.g., CPLEX or LP SOLLVE.

The complete formulation is as follows:

Objective:

$$\text{minimise} \sum_{l \in L} \left( \sum_{o \in O} x_{i}^{o} c^{o} + Y_{l} \right)$$  \hspace{1cm} (2.7)

Subject to constraints:

$$\sum_{j=1}^{N} x_{ij}^{o} - \sum_{k=1}^{N} x_{ki}^{o} = \begin{cases} 0 & \text{if } i \neq \text{source of } o \land i \neq \text{sink of } o \\ 1 & \text{if } i = \text{source of } o \\ -1 & \text{if } i = \text{sink of } o \end{cases}$$  \hspace{1cm} \text{for all nodes } i \text{ and commodities } o \hspace{1cm} (2.8)

$$\sum_{o \in O} x_{i}^{o} c^{o} + Y_{l} \leq C_{l} \text{ for all links } l \in L$$  \hspace{1cm} (2.9)

$$\sum_{o \in O} (\gamma_{i}^{o} - \delta_{i}^{o}) c^{o} \leq Y_{l'} \text{ for all } (l', l'') \in L^2 \text{ except for } l'' = l''$$ \hspace{1cm} (2.10)

$$x_{i}^{o} \in \{0,1\}, y_{l}^{o} \in \{0,1\} \text{, for all links } l \in L \text{ and commodities } o \in O$$ \hspace{1cm} (2.11)

$$x_{i}^{o} + y_{l}^{o} \leq 1 \text{ for all links } l \in L \text{ and commodities } o \in O$$ \hspace{1cm} (2.12)

$$x_{i}^{o} + y_{l'}^{o} \geq \gamma_{i}^{o}$$
$$x_{i}^{o} - y_{l'}^{o} + 1 \geq \gamma_{i}^{o}$$
$$-x_{i}^{o} + y_{l'}^{o} + 1 \geq \gamma_{i}^{o}$$
$$x_{i}^{o} - y_{l'}^{o} - 1 \leq \gamma_{i}^{o}$$

for all pairs of links $(l', l'') \in L, l' \neq l''$ and commodities $o \in O$ \hspace{1cm} (2.13)

$$x_{i}^{o} + x_{l''}^{o} \geq \delta_{i}^{o}$$
$$x_{i}^{o} - x_{l''}^{o} + 1 \geq \delta_{i}^{o}$$
$$-x_{i}^{o} + x_{l''}^{o} + 1 \geq \delta_{i}^{o}$$
$$x_{i}^{o} + x_{l''}^{o} - 1 \leq \delta_{i}^{o}$$

for all pairs of links $(l', l'') \in L, l' \neq l''$ and commodities $o \in O$ \hspace{1cm} (2.14)

$$\gamma_{i}^{o}, \delta_{i}^{o} \in \{0,1\} \text{ for all pairs of links } (l', l'') \in L, l' \neq l'' \text{ and commodities } o \in O$$ \hspace{1cm} (2.15)
2.2.4 Heuristics for Improving the Performance

The described methods can be further generalised, can be applied to some other problems and can be improved by some simple heuristics. Here follow some ideas.

Both Phase 1 and Phase 2 can be solved by the improved Simulated Allocation (SA++) (Section 2.1.4).

The method can also be improved by sorting the traffic demands according to decreasing bandwidth demands, i.e., the node-pairs with larger demands are routed first. This is analogous to the approximation used for solving the bin packing problem.

Another idea for improving the second phase can be sorting by increasing number of hops. This means that node-pairs which are closer, i.e., connected by less hops are routed first. The best way is to use the normalised linear combination of both traffic and distance factors and then to sort the node-pairs in non-decreasing order according to the given Sort values. Sort, Distance and Traffic are arrays, indexed with the indexes of node-pairs.

\[ \text{Sort}[i] = (1 - a) \cdot \frac{\text{Distance}[i]}{\max \text{Distance}} - a \cdot \frac{\text{Traffic}[i]}{\max \text{Traffic}} \]

Note, that the impact of traffic demand values are also taken into account while sorting the node-pairs. The Distance and Traffic values are first normalised to have values between 0 and 1. Since the edges should be sorted by non-decreasing distances and by non-increasing traffic demands we subtract the normalised traffic value from normalised distance value. If the distance is more important for sorting than the traffic demand then value of parameter \(a\) is lower (0 \(\leq\) \(a\) \(\leq\) 1). Values of Sort are between \(-1\) and \(1\).

The initial solution of the method proposed by Pióro [5] can be derived by this greedy ranking followed by randomised improving iterations. This combined method improves the method, because the quality of results obtained by SA depends on the running time. If SA is started from a "good" initial point it converges faster or gives better results within the same, limited time period.

In the first phase an approximation can be applied for the standard MCMCF problem instead of formulating and solving it as a linear program. The approximations give slightly worse results, but the running time of this phase can be decreased by about two orders of magnitude.

The thrifty capacity allocation (TCA) - as already mentioned - can be applied in four ways: at the very end of the whole configuration, at the end of each iteration, before or after searching for a new restoration path.

When applied at the very end it reduces the total allocation for restoration paths. For this reason it may be allowed to slightly override the capacity limit by allocations, because at the end these allocations are reduced.

When applied at the end of each iteration, the free resources can be used in the next cycle by both primary and restoration paths. In this case TCA should be applied between Phase 2 and Phase 3.

The third approach - applying TCA before searching for each new restoration path - is the more time consuming one, but gives the even better results. The capacity allocations should be recalculated here after determining any single restoration path.

The fourth approach - applying TCA before searching for each new restoration path - is the most time consuming one, but gives the best results. The capacity allocations should be recalculated here before determining any single restoration path.

The TCA method is independent of the proposed configuration method and can be applied for SA or even for any other management or topological design task in unchanged manner in all three variants.
The proposed method is independent of the physical topology, i.e., it is efficient for mesh, interconnected ring or any other topology as well.

**Time Complexity**

For Dedicated Protection the SA++ algorithm is called $k$ times where $k$ is a small integer, i.e., in this case the time complexity of ICS++ is $O(|V|^2|D|^2)$. In case of Shared Protection at most $(|E|/2)^2$ matrix operations are needed in each step of the algorithm, i.e., the time complexity of ICS++ is $O(|V|^2|D|^2|E|^2)$. On the other hand ILP based methods are of exponential complexity.

### 2.2.5 Numerical Results

**Dedicated Protection**

In this section performance of 4 methods has been compared. First, the ILP approach where both paths are determined simultaneously: ILP2. Second, the ICS with heuristics for improving: ICS++. In the first phase the ILP has been used as described in Section 2.2.2. In the second phase the demands were sorted according to the amount of traffic to be carried. In the case of unsuccessful allocation, i.e., when there is not enough capacity on links, which are necessary for accommodating traffic of the considered node-pair, then one of these critical links is randomly chosen, then one of the commodities using this link is being randomly deallocated. Then the next demand is tried, etc. Third, the original SA, and fourth, its improved version: SA++.

The “improved” stands for sorting node-pairs according to decreasing traffic demands; starting to delete routed traffic demands when link capacities are exhausted; deleting traffic demands crossing links with capacities exhausted; and applying Suurballe’s algorithm for finding a pair of shortest paths.

All results were obtained on an Sun Ultra Sparc running SunOS 5.5. The test networks were optimally designed networks with tightly dimensioned capacities for optimally routed demands. The input of the design method [24] was randomly generated position of nodes and a random traffic matrix. The test networks N5, N15, N25 and N35 were of size of 5, 15, 25 and 35 nodes respectively. The test networks are depicted in Figure 2.3 and some other characteristic parameters can be seen in Table 2.3. The networks obtained by adding new links are described in Table 2.3 as well: N5+1, N15+1, N25+4, N35+1.

We tested first the algorithms for these tightly dimensioned networks, then we first inserted additional links, or increased all link capacities by 10% to increase the probability of finding a valid configuration by allowing more freedom to the algorithms.

<table>
<thead>
<tr>
<th>Original Networks</th>
<th>No. of Nodes</th>
<th>No. of Links</th>
<th>No. of Commodities</th>
<th>Extended Networks</th>
<th>New Links (by node-pairs)</th>
<th>Capacity of New Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>N5</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>N5 + 1 link</td>
<td>0-2</td>
<td>53</td>
</tr>
<tr>
<td>N15</td>
<td>15</td>
<td>15</td>
<td>105</td>
<td>N15 + 1 link</td>
<td>1-11</td>
<td>20</td>
</tr>
<tr>
<td>N25</td>
<td>25</td>
<td>31</td>
<td>300</td>
<td>N25 + 4 links</td>
<td>2-20, 4-20, 6-13, 9-20</td>
<td>200</td>
</tr>
<tr>
<td>N35</td>
<td>35</td>
<td>51</td>
<td>595</td>
<td>N35 + 1 link</td>
<td>10-23</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 2.3: The 8 examined networks: originals and their extensions

We compared the four algorithms on 10 networks from three aspects: the running time of the algorithms, the total amount of resources used and the total amount of resources used by
primary paths only. The results can be also seen in Table 2.4. The dash ‘-’ stands were no results were found at all. The ILP 2 is the most accurate method, it gives the globally optimal result, but its running time is the longest and it increases exponentially with the increasing number of variables, and for this reason it can not be used for larger networks. During the tests it ran out of memory for N35 in all cases and for the tightly dimensioned N25 network as well. The ICS++ is the second slowest method, because of its slow first phase, but it finds result for all networks, even for the N25. This method gives shorter working paths. The SA++ gives results faster then SA, and it finds result even when SA fails.

It can be seen that the algorithms utilise less than 100% of resources for N25 and N35. This happens when the configuration algorithm finds better result than the initially designed and tightly dimensioned network. It can also be seen, that our heuristics improve the performance of the SA algorithm.

The proposed ICS method has a very important advantage due to splitting phases in which working and restoration paths are being determined. When not enough resources are available not all but only some O-D pairs may afford to have a backup path. There can be assigned restorability priorities to these O-D pairs. Then in the first phase all working demands are satisfied optimally, and then in the second phase the demands are sorted according to their restorability priorities. This simple sorting ensures obeying restoration priorities, while the other methods (e.g., SA) cannot handle them efficiently. Although it has longer running time, it yields always solution.
Shared Protection

In this section performance of two algorithms (ICS++ and SA++) applied for TCA has been compared (see Table 2.4). Because of the problem complexity ICS failed to give any solution for networks larger than 15 nodes.

The running time of these algorithms is much longer than those without TCA. On the other hand, savings of around 40%, for our 10 test networks 35-46.5% can be achieved.

Dedicated vs. Shared Protection

In this Subsection we study and compare all six algorithms, i.e., Dedicated and Shared Protection. In Figure 2.4 we can see the running time of the six algorithms on a logarithmic scale. ILP2 was not able to find solution in acceptable time in networks N25, N35, N35+11, N35+10%, while with SA, SA++ and ICS++ the running time is moderate. On the other hand TCA-ICS++ (SP) is of one order slower than ICS++ (DP).

![Graph showing running time of algorithms](image)

Figure 2.4: Running time of the 6 algorithms in 10 test networks

In Figure 2.5 the network utilization is depicted yielded by the algorithms. The gain of Shared Protection is convincing.

In Figure 2.6 the network utilization is depicted versus running time in three of the ten test networks: N25, N25+41, N25+10%, and each with six algorithms (each point on the Figure represents a network and an algorithm). TCA ICS++ is a good compromise between running time and network utilization.
2.2.6 Summary

A heuristic two phase method (ICS) has been proposed which gives approximate results for an NP-hard problem in polynomial time\(^2\) This problem is inspired by the problem of configuring telecommunication networks. The advantage of it over other methods is, that in networks where limited amount of resources is available, it finds restoration paths for O-D pairs according to their restorability priorities. In the first phase the Minimal Cost Multicommodity Flow problem was solved by the improved Simulated Allocation (SA++). In the second phase the backup paths were found by using any of shortest path algorithms for each node-pair one-by-one according to their restorability priorities or by SA++.

Second, the problem is formulated as an Integer Program, and solved by CPLEX. The path diversity has been taken into account as a constraint. The results obtained by the proposed ICS and IP methods were compared to Simulated Allocation.

Third, simple heuristics were proposed for improving the performance of the ICS and SA methods.

The fourth novelty is the TCA method, which saves a significant amount of resources. This method can be used whenever very fast restoration is required (e.g., by APS) in an network and under assumption that a single failure only can occur at time. TCA can be applied for design, configuration and reconfiguration of networks.

In the following section a subroutine of this problem is considered, namely, how to find an optimal pair of disjoint paths.

\(^2\)Except for those evaluations where ILP has been used for optimisation in the first phase. In that case the first phase is more time demanding, but still much faster then ILP applied for the whole problem.
2.3 Algorithms for Asymmetrically Weighted Pair of Disjoint Paths in Survivable Networks

To find node-disjoint path-pairs is a critical issue of survivable networks. One of the paths have frequently higher priority in which case the problem is called asymmetrically weighted pair of disjoint paths. In this novel approach a special, weighted objective function should be minimized. We formulate the problem, prove it to be NP-complete and several heuristics are proposed based on Surballe’s algorithm, Integer Linear Programming (ILP), Linear Relaxation (LR), and Minimum Cost Network Flow (MCNF) algorithm to achieve best performance in polynomial time. The simulation results show that the heuristic based on MCNF yields the best performance in terms of cost and running time. The optimal solution is found in about 99.9% of all cases while a near-optimal solution in the remaining few cases [C6].

2.3.1 Introduction

Modern telecommunication networks carry tremendous amount of traffic that risks a serious data loss when a failure occurs (e.g., a fiber cut or a node failure). The most common way to protect a path is to find a protection route over which the protected data flow can be switched when a failure occurs in the network element the path passes through. The working and protection path has to be node-disjoint. The protection path can be computed after its corresponding working path is determined, however, this may lead to a non-optimal solution. On the contrary, a working path and a protection path can be found simultaneously to yield the globally optimal node-disjoint path-pair.

This problem is also known as node-disjoint diverse routing problem and usually solved intuitively by the Two-Step-Approach algorithm [44]. This first finds the shortest path, then
finds the shortest path in the same graph with the edges and nodes of the first path being temporarily erased. Although this method is straightforward and simple, it may fail to find any disjoint path-pair since erasing the first path can isolate the source node from the destination. This can happen even if the network is highly connected\(^3\). This approach may perform well if the task is to reduce the cost of working path without any attention paid to the protection path. We will call this approach 2D (two-step-Dijkstra).

To find two disjoint paths with minimal total cost, Suurballe’s algorithm (SUURB) [12] [41] can be used. This is a modification of Dijkstra’s algorithm and guarantees to find the disjoint path-pair if it exists, but the found working path can be longer than the optimal one.

However, in many cases a network operator’s aim is not only to reduce the total cost but to optimize the cost sum with weighting one of the paths (e.g., the working path). In this Section, we focus on the optimization of the sum

\[ c\text{cost}(P_1) + \text{cost}(P_2), \]

where \( \text{cost}(P_1) \) and \( \text{cost}(P_2) \) are the cost of the working path and of the protection path, respectively.

Parameter \( \alpha \) is the weight of the working path, which is determined by the protection strategy. In other words the consumption of network resources by the working path is \( \alpha \) times more important than that of the protection path. For example, if a shared protection scheme (e.g., \( 1 : N \) or \( M : N \)) is adopted, \( \alpha \) could be 10, 100 depending on how many paths share a link and on the amount of best-effort traffic flowing in the network. On the other hand, \( \alpha \) could be small (e.g., \( 1 \sim 5 \)) for dedicated protection (e.g., \( 1 + 1 \)) since there is no difference in the consumption of network resources between a working path and a protection path.

In this Section we will present several solution alternatives. After formulating the problem we will prove that the problem is NP-hard. In Section 2.3.4 the problem is formulated as an Integer Linear Program (ILP). Then the proposed methods are discussed that are based on Linear Relaxation and Minimal Cost Flow. In Section 2.3.6 methods are given to solve the split-flow problem that arise in relaxations. Finally, the numerical results are discussed.

### 2.3.2 Problem Formulation

The input of the problem is a directed or undirected graph \( G(V, E) \) representing the network topology where the set of nodes denoted by \( V \) and the set of arcs denoted by \( E \) with a positive cost for each link (denoted as \( c_e \) of edge \( e \in E \)).

The demand has a source \((s)\), a destination \((t)\), and a factor (or weight) \( \alpha \) for weighting the working path. The task is to find edge- or vertex-disjoint working \((P_1)\) and protection \((P_2)\) paths from a given source to destination such that \( f(s, t, \alpha) = \alpha \text{cost}(P_1) + \text{cost}(P_2) \) should be minimal. In this Section we are focusing on vertex-disjoint paths and we mean this by the term disjoint path. Please note that here the routing of one demand is solved. The described methods should be called as a subroutine if a demand is allocated.

As mentioned if \( \alpha = 1 \) then the problem can be solved by Suurballe’s algorithm [12] with complexity of \( O(n^2 \log n) \).

### 2.3.3 Complexity Study

We are going to present a series of approximation algorithms to solve the asymmetrically weighted disjoint path problem. The optimization is proved to be NP-complete [6].

---

\(^3\)One can easily construct a graph in that the shortest path between \( s \) and \( d \) takes all nodes.
The result given by Suurballe’s algorithm is optimal only in the case of $\alpha = 1$. On the other hand, it can approximate the derivation of $f(s, t, \alpha) = \alpha \cos(P_1) + \cos(P_2)$ no worse than the optimal cost factored by $\frac{\alpha + 1}{2}$, where $P_1$ and $P_2$ are two disjoint paths between a given source $s$ and destination $t$ and $\alpha$ is a given constant, i.e.,

$$f_{app}(s, t, \alpha) \leq \frac{\alpha + 1}{2} \cdot f_{opt}(s, t, \alpha)$$

where $f_{app}(s, t, \alpha)$ is the result given by the algorithm that approximates (which is Suurballe in this case), $f_{opt}(s, t, \alpha)$ is the optimal solution. The above relationship is defined as an approximation with a factor of $\frac{\alpha + 1}{2}$. The proofs for the following theorems can be found in [C6].

**Theorem 1.** Suurballe’s algorithm [41],[12] approximates the cost of the optimal asymmetrically weighted path-pair with a factor of $\frac{\alpha + 1}{2}$.

The following theorem is to claim that there is no any polynomial-time algorithm that can approximate the derivation of an asymmetrically weighted path-pair within a factor of $\frac{\alpha + 1}{2}$, unless $P = NP$.

**Theorem 2.** [42]. On directed graphs it is $NP$-hard to approximate the cost of the optimal asymmetrically weighted path-pair within a factor of $\frac{\alpha + 1}{2}$, i.e, if an algorithm achieves an approximated cost $f_{app}(s, t, \alpha)$, in which

$$f_{app}(s, t, \alpha) \leq C \cdot f_{opt}(s, t, \alpha)$$

$$1 \leq C < \frac{\alpha + 1}{2}$$

then the algorithm is not polynomial, unless $P = NP$.

### 2.3.4 Integer Linear Programming (ILP)

The problem is also formulated as an Integer Linear Program (ILP) and can be solved optimally by any available solver.

**Objective:**

$$\text{minimize } \sum_{e \in E} (\alpha x_e + y_e) c_e \quad (2.16)$$

**Subject to constraints:**

$$\sum_{j=1}^{N} x_{ij} - \sum_{k=1}^{N} x_{ki} = \begin{cases} 0 & \text{if } i \neq \text{source} \land i \neq \text{sink} \\ 1 & \text{if } i = \text{source} \\ -1 & \text{if } i = \text{sink} \end{cases}$$

$$\sum_{j=1}^{N} y_{ij} - \sum_{k=1}^{N} y_{ki} = \begin{cases} 0 & \text{if } i \neq \text{source} \land i \neq \text{sink} \\ 1 & \text{if } i = \text{source} \\ -1 & \text{if } i = \text{sink} \end{cases}$$

for all nodes $i$ \hspace{1cm} (2.17)

$$x_e \in \{0, 1\}, y_e \in \{0, 1\}, \text{ for all edges } e \in E \quad (2.18)$$

$$x_e + y_e \leq 1 \text{ for all edges } e \in E \quad (2.19)$$
The binary flow indicators \( x_e \) \((y_e) \) take value 1 if the working (protection) path uses edge \( e \) or 0 if not. \( \alpha \) is the weight factor for the primary path and \( c_e \) expresses the cost of using edge \( e \).

According to constraint (2.19) which ensures edge-disjoint paths, the working and backup paths may not use the same edge. For making the working and backup paths node-disjoint the following constraint should be used instead of (2.19): \( \sum_{j=1}^{N}(x_{ij} + y_{ij}) \leq 1 \) for all nodes \( i \) except the source and the sink.

### 2.3.5 Relaxations

As shown in 2.3.3 even a "good" relaxation of the problem is \( \mathbf{NP} \)-hard, therefore, derivation of the optimal solution for large networks (e.g., with 100 or more nodes) needs an unacceptable amount of computation time. It is crucial especially in the case of dynamic route computation. To solve a linear program (LP) or a flow problem is much less complex task than to solve an integer program. With the following relaxation process we got 1.5 to 100 times faster running time, depending on the size and density of the network. Both LP and ILP can be solved by many software packages. We did experiments with CPLEX and LP\_SOLVE. Besides, we adopted two types of relaxation: Linear Programming Relaxation (LPR) and Single Flow Relaxation (SFR) in this work.

**Linear Programming Relaxation (LPR).** LPR is a relaxation of the above ILP formulation by relaxing the constraint (2.18), i.e., the condition that the variables should be integers. That is, instead of (2.18) the following constraints are to be applied: \( 0 \leq x_e \leq 1 \) and \( 0 \leq y_e \leq 1 \), for all edges \( e \in E \), which however allows fractional flows and may yield useless result (as shown in Figures 2.9 and 2.10). Although relaxation is of polynomial complexity in contrast to the integer problem but the split-flow problem that may occur needs to be further manipulated. We will discuss this in details in Section 2.3.6 and show that in most of the situations the optimal solution still can be found.

**Single Flow Relaxation (SFR).** The idea of SFR is based on Minimal Cost Flows (MCF) with the following conditions: all capacity constraints are set to \( \alpha \) and a flow of size \( 1 + \alpha \) is to be found. SFR renders split-flow even more often as shown in Figures 2.7 and 2.8.

![Figure 2.7: Example when SFR gives optimal solution. (Dashed arrows represent the working path, with flow size of \( \alpha \) and solid lines the protection path, with flow size of 1).](image)

There exists a large number of fast Minimal Cost Network Flow (MCNF) implementations, e.g., that of Goldberg [43]. In addition the Network optimizer of CPLEX can solve the MCNF problem efficiently (in \( O(n^2m \log n) \) time). It can also be solved by any other available LP solver, such as LP\_SOLVE.
2.3.6 Methods for Solving the Split-Flow Problem

The result of our experiments shows that by the assistance of LR and SFR an optimal solution can be derived in around 80-90% of all cases (in which the relaxation yields a two-path solution). In the other cases (10-20%) the relaxation does not yield a two-path solution but a split flow. However, if the split-flow problem occurs, the result of the relaxation may be useless. In the following paragraphs we propose two approaches, Tiny Additional Costs (TAC) and α-Shifting, for solving the split-flow puzzle, and deriving an optimal or a near-optimal solution for the diverse-routing problem. Before presenting our approaches, two types of split-flow problems are defined and described as below.

Reparable split. Because of the essential characteristics of the LPR and MCF solvers, the flows in the network may be split if there are two or more optimal paths between source and destination. As an example shown in Figure 2.9, every link cost is one, and the total amount of flow from $s$ to $d$ is one as well. The numbers specified on the edges represent the percentage of flow. In this case, both the working (solid arrows) and the protection (dashed arrows) paths are split although they could take single paths with the same minimum cost as well. Although this situation occurs in a practical network with relatively small probability \(^4\), it may ruin the whole computation process once it exists. To solve this type of split-flow problem, Tiny Additional Costs (TAC) is proposed and discussed in Section 2.3.6.

---

\(^4\)having more than one path of equal cost between source and destination
Unreparable split. Figure 2.10 shows another type of split-flow problem (links s-1 and 12-d have high cost, all other links have low cost). In this case the split-flow would have minimum cost, which cannot be repaired in polynomial time\(^5\). In other words, if a polynomial computation time has to be guaranteed, it may not be able to find an optimal solution. To cope with this situation, we propose a heuristic, \(\alpha\)-shifting, to find a possibly good repair in polynomial time.

Tiny Additional Costs (TAC)

TAC is proposed to solve the reparable split problem, in which a random number \(\epsilon\), is added to each edge \(e\), where \(0 \leq \epsilon \leq 1/\alpha n\), \(n\) is the number of nodes in the network. Based on the fact that the original cost of links are all integers, the total cost of the path-pair is integer as well. On the other hand, the number of links used by the path-pair is smaller than \(n\), that is why the deviation of the total cost (e.g. \(\alpha \text{cost}(P_1) + \text{cost}(P_2)\)) is less than one, since \(\sum_{e \in E_1} \epsilon_e \alpha n < 1\), where \(E_1\) is the set of edges used by \(P_1\) or \(P_2\). Therefore, the result of the relaxation is not influenced by applying TAC, while the probability that more than one optimal path exist between source and destination can be significantly reduced.

Two methods based on TAC are tested to solve the reparable split problem. First, LPT is when TAC is performed before the relaxation. If the flow problem still occurs in the relaxation, it fails. Second, LPT+ is an enhancement of LPT when TAC is performed repeatedly on the original network until either the split-flow problem is solved (or an optimal solution is derived) or the time of iterations reaches \(\max_{LPT+}\), what is user defined.

\(\alpha\)-Shifting

Recall that ILP can solve the problem and guarantee the optimal solution. However, due to NP-completeness of the ILP, large networks (e.g. number of nodes > 200) cannot be handled in this way. To reduce the complexity, we propose a heuristic, \(\alpha\)-shifting, to yield a near-optimal solution within a polynomial computation time. The basic idea of \(\alpha\)-shifting is to search \(\alpha'\) that

\(^5\)unless \(P=NP\) the problem is \(NP\) complete (see Theorem 2) while LP can be solved in polynomial time [45]
is closest to $\alpha$, with which the relaxation process works with unsplit flows or at least reparable flows, i.e., a sub-optimal path pair is yielded.

In order to understand the method, we will analyze the cost of the path-pair plotted against $\alpha$, denoted by $t_{(s,d)}(\alpha)$. It can be proved that $t_{(s,d)}(\alpha)$ is continuous, monotone increasing, piecewise linear and concave. The piecewise linearity of the function comes from the fact that each path-pair has a cost of $\alpha \cdot \text{cost}(P_1) + \text{cost}(P_2)$ which is linear against $\alpha$. Also possible to prove that the solution of any LP relaxation plotted against $\alpha$ has the same properties (denoted with $t_{(s,d)}^{\text{relax}}(\alpha)$). Obviously, $t_{(s,d)}^{\text{relax}}(\alpha) \geq t_{(s,d)}(\alpha)$ and equal if the LPR or SFR gives integer solutions (see Figure 2.11). It can be proved that $t_{(s,d)}^{\text{relax}}(1) = t_{(s,d)}(1)$. Figure 2.11 shows an example

\begin{figure}[h]
  \centering
  \includegraphics[width=0.5\textwidth]{fig2.11}
  \caption{An example of the cost function plotted against $\alpha$}
\end{figure}

where the relaxation process suffers from the unrepairable split-flow problem when $\alpha = 4$. With the proposed $\alpha$-shifting scheme we are going to find $\alpha'$ closest to $\alpha$ such that a sub-optimal path pair can be derived. A binary search is suggested here to find $\alpha'$, in which $\alpha' = 6$ will be tried first, followed by $\alpha' = 5$, and then $\alpha' = 2$ and $\alpha' = 3$. The heuristic works as follows:

**Step 1:** Run relaxation SFR or LPR with $2\alpha$.

**Step 2:** If successful then find $\alpha'$ closest to $\alpha$ with binary search ($\alpha < \alpha' < 2\alpha$)

**Step 3:** If not successful find $\alpha'$ closest to $\alpha$ with binary search ($1 < \alpha' < \alpha$)

With this method in the worst case $\alpha'$ goes to 1 where the problem is degraded and gives the same result as Suurballe’s algorithm.

**Methods Summary**

The process of solving asymmetrically weighted disjoint path-pair is summarized by the flow chart shown in Figure 2.12. The approaches $0 \rightarrow 1 \rightarrow 2 \rightarrow (3) \rightarrow (4)$ can guarantee to derive an optimal solution (if there is any) at an expense of being non-polynomial during (4). On the other hand, the polynomial computation time can be achieved without a guarantee of optimality via the approach $0 \rightarrow 1 \rightarrow 2 \rightarrow (3)$.

**2.3.7 Numerical Results**

Figure 2.13 shows a typical function of cost against $\alpha$ in the case when there are different optimal path pairs at different values of $\alpha$. On the vertical coordinate the value 1 represents the
optimum cost (the cost was normalized). With the growth of $\alpha$, 2D gives better and better results compared to the optimum, i.e., it is a decreasing function. Note that for 2D the existence of the solution is not guaranteed (see also Table 2.9, row 2D success). On the other hand Suurballe’s algorithm (SUURB) gives optimal solution at $\alpha = 1$ and moves away from it if $\alpha$ increases. In this example in the interval $4 < \alpha < 12$ both 2D and SUURB yields more than 1.5 times worse solution compared to the optimal one. It can be seen that our method is always very close to the optimum. However, one has to mention that in a number of cases not only our method but also 2D and SUURB give the optimal solution. That is the reason to get smaller differences between the average costs of all node-pairs as for the "problematic" node pairs in Tables 2.5, 2.6, 2.7.

![Flowchart for solving the asymmetrically weighted optimal disjoint path-pair problem](image)

Figure 2.12: Flowchart for solving the asymmetrically weighted optimal disjoint path-pair problem

![Typical function of cost against alpha normalized with the value of the optimal solution when the optimal solution differs from the Suurballe’s one](image)

Figure 2.13: A typical function of cost against alpha normalized with the value of the optimal solution, when the optimal solution differs from the Suurballe’s one.

Four methods are compared according to 3 criteria: running time, success ratio, and cost. Simulations have been carried out between each pair of nodes and average calculated for running time and cost. Success means the percentage of node-pairs for that the method gives any solution. The methods are: (1) ILP that yields the optimal solution, (2) Suurballe (SUURB) that minimizes the total cost of the two paths, (3) 2D that minimizes the cost of the first paths, (4) LPR, and (5) SFR: our proposed methods. The tests were carried out on an 500 MHz Intel
PentiumIII based computer with 64 Mbytes of memory. ILOG's CPLEX 6.5 and LP SOLVE has been used for ILP tasks that are widely used for solving linear programs.

Tables 2.5 and 2.6 show numerical results for a dense 30-node and a sparse 35-node network. The running times show the average computational time for one demand in milliseconds. The success value is always 100% except in case of the 2D method since it does not always yield solution. Average normalized cost of all demands are shown for $\alpha = 1, 5, 100$, and average normalized cost for the "problematic" path-pairs. "Problematic" path-pair means that ILP and SUURB does not yield the same quality of result for $\alpha > 1$. Table 2.7 shows numerical results for 10 demands of a 400-node network.

In smaller networks (n30, n35) SUURB and 2D have shorter running time but worse quality than the three other methods. In case of n35 2D did not find solution in 7% of the cases. LPR did not split any flows. LPR gives very similar running time compared to ILP. In case of n30 and n35 SFR gives always the optimal solution. The reason for that is that in networks of such size SFR runs ILP if the flow splits. The running time of SFR is a fraction of ILP while it gives result very near to optimum.

The running time of the algorithm for a network of 1000 nodes and 1900 edges is as follows: 19.2s for LPR, and 7.1 for SFR. The quality of the results are similar to the other networks.

Table 2.8 shows the percentage of optimally routed demands in eight networks: four of them are sparser (N5, N15, N25, N35) and seven are denser (N30A, N30B, N30C, N50, N60, N100). The network names refer to the number of nodes. Simulations have been carried out between all pairs of nodes and the percentage of optimal result is shown in case of SFR, LPR, LPR+SFR, LPT, LPT+. In case of LPR+SFR after LPR SFR is applied if and only if LPR did not yield optimal solution. LPR+SFR yields optimal solution in all sparse networks, and about 90% of all demands in dense networks, and nearly optimal in the remaining 1%. LPT+ yields optimal solution for almost all of the demands in all networks.

In Table 2.9 three methods are compared according to the increase of runtime on six different networks.

2.3.8 Summary

We have proposed methods for finding asymmetrically weighted minimum cost disjoint paths. It finds solution in polynomial time that is in 99.9% of all cases the optimal one. In the remaining cases (1) either a near-optimal solution is delivered, (2) or the optimal solution can be found in longer time. The obtained results can be used for configuring survivable WDM, SDH, ATM, MPLS, and other networks. The proposed methods deliver results even in large networks (1000 nodes or more).

In the following Section we deal with characterizing infocommunication networks by their minimum and average availability values and advance of these values over an aimed level.

2.4 Availability Evaluation and Advance of Infocommunication Networks

Availability is one of the most important performance criteria, which characterises telecommunication network quality. The advance of availability is an important step towards guaranteeing Quality of Service (QoS) in telecommunication networks. Significant efforts are being made to construct a network as reliable as possible. In [68] an integer programming based method is proposed for SONET/WDM ring networks to determine the number of redundant components
and wavelength units necessary to achieve a guaranteed level of survivability. Unfortunately, this integer programming based method has very long running time, and it is limited to smaller networks. A tool to generate networks and a tool to evaluate the reliability of multilayer networks are described in [66]. In [69] the availability to cost ratio is compared for three protection strategies: unprotected demand, diversity protected demand and hot-standby protection of transmission system (1+1).

Accordingly, in the literature methods are presented to evaluate the availability of an existing network, e.g. [67], and there are many known methods to increase availability, e.g. [C2, 65]. But there is no known general and fast method that advances the availability of the network over an aimed level. We propose a three phase, iterative heuristic method which has been worked out during the cooperation between Hungarian Telecommunication Operator PanTel, and the Budapest University of Technology and Economics. The tool includes availability evaluation of networks (i.e., minimum or average of point-point availabilities) consisting of single or multiple layers. However, network availability is treated in a way that is different from the traditional terminology since in our model protected paths have point-point availabilities. Connection availability is easy to calculate while can be extended to the whole network by minimum and average calculations. In this Section single layer algorithms are dealt with, while in Section 3.6 they are applied for multilayer networks. The methods proved to be very efficient even for large, 100-node SDH networks and can be applied for ATM and WDM networks as well.

High-speed networks nowadays relay on fiber optics. Therefore, in case of a network failure the operator should react as fast as possible to minimise the loss of data. For this reason the network fault tolerance becomes more and more important. Many papers deal with designing fault tolerant networks [C2, 24, 63, 68]. These papers focus on establishing networks or configure them in an optimal way. Here a different approach is proposed. What should be done if connections between nodes do not reach a required level of availability? The focus of our work is how to extend existing links and nodes of telecommunication networks to achieve this level of availability. A three-phase method, Iterative Availability Enhancement (IAE), is proposed: (1) first, try to find redundant paths within the existing link capacities; (2) second, extend the capacities of existing links if needed; (3) third, extend the topology of the network, that is to install new links if needed and possible. The running time of this polynomial algorithm is less than a minute for the Pan-European transport network (29 nodes) and less than two hours even for a heavy loaded Hungarian network (90 nodes) [C4].

### 2.4.1 Component and Connection Availability Modelling

In this section, we describe the availability analysis of SDH and WDM network components and connections [8]. The value of availability ($A$) gives the probability that an item will carry out its required mission. Mean time to repair (MTTR) is the total corrective maintenance time by the event of a failure to the total restore of the item. Mean time to failure (MTTF) means the average elapsed time between two failures of the item. The availability of the system can be expressed by the term $A = \frac{MTTF - MTTR}{MTTF}$.

In case of **Series Configuration** the system is composed of $m$ independent subsystems in series. If any one of these subsystems fails, the entire system fails. The expressions for this configuration is:

$$A_s = \prod_{j=1}^{m} A_j$$  \hspace{1cm} (2.20)
where \( A_s \) denotes the system availability and \( A_j \) the availability of the \( j^{th} \) subsystem. The **Parallel Configuration** is composed of \( n \) independent subsystems in parallel. At least one subsystem must function for the success of the system:

\[
A_s = 1 - \prod_{j=1}^{n}(1 - A_j) \tag{2.21}
\]

The **\( k \)-out-of-\( n \) Configuration** is good if at least \( k \) of its \( n \) units are working properly:

\[
A_{k/n} = \sum_{i=k}^{n} \binom{n}{i} A^i (1 - A)^{n-i} \tag{2.22}
\]

**\( N+1 \) Configuration** gives the availability of one unit that shares one redundant unit with \( N-1 \) other identical working units:

\[
A_{N+1} = \sum_{i=1}^{N+1} \min(1, \frac{i}{N}) \binom{N+1}{i} A^i (1 - A)^{N+1-i} \tag{2.23}
\]

The node of a telecommunication network is divided into two parts from the availability point of view (Fig. 2.14): the equipment and the traffic cards. If the equipment is disrupted then all links adjacent to it break down. If one of the traffic cards get in failure state then only the one link that is connected to this card is broken. Consequently, the availability of an equipment is constant, while the availability of a link consists of a fix (traffic card), and a length-dependent part. The length-dependent part is proportional to the length of the link.

![Node and link availability model](image)

Figure 2.14: Node and link availability model

Availability of a single path - between two distant nodes - is the product of all link and equipment availabilities that are used by the path (1). Availability of a protected path (i.e., two edge or node disjoint paths) is calculated using the parallel configuration for the two single paths (2). Nodes that are connected to the ring by merely one link are called leaves. Leaves cannot be fully protected. Availability of a partially protected path (e.g., if source or destination node is a leaf) is the product of the common and the separated availability:

\[
A = A_{\text{common}} A_{\text{separated}} \tag{2.24}
\]

Where common availability \( A_{\text{common}} \) is the product of the availabilities of the elements that are used by both the working and the protection path:

\[
A_{\text{common}} = \prod_{\text{\( j^{th} \) element is common}} A_j \tag{2.25}
\]
and the separated availability is the parallel configuration of the working and protection availability

\[
A_{\text{separated}} = 1 - (1 - A_{\text{work}})(1 - A_{\text{prot}})
\]  

\[
A_{\text{work}} = \prod_{\text{\textit{j}th element in}} A_j \quad \text{working path only}
\]

\[
A_{\text{prot}} = \prod_{\text{\textit{j}th element in}} A_j \quad \text{prot path only}
\]

The term \textit{path availability} is also common for the availability of a path. The term \textit{network availability} is common for the probability that certain number of paths are in up-state, i.e., is working. Here and in Section 3.6 two main measures of the network are considered, which characterise telecommunication network quality: first, the \textit{average availability} \( (A_{\text{ave}}) \) is the mean of all connection availabilities, while the point-point availabilities may be weighted by the capacities of the demands; second, the \textit{minimum availability} \( (A_{\text{min}}) \) is the minimum of all connection availabilities.

\subsection{2.4.2 Problem Formulation}

The formulation of the problem is as follows. The topology of the existing network, link capacities, demand matrix, link and node availabilities are given. The availability value is constant for nodes, and it consists of constant and marginal value for links. Our aims are (1) to determine both the working and protection paths, (2) to determine for each link how much capacity should be added to it so that each working and protection path can be routed, (3) to determine the minimal number of new links to be installed. The objective is to use as few resources (link capacities) as possible. The additional constraint is the given degree of minimum availability \( A_{\text{min}} \) (e.g., 0.9999) to achieve.

\subsection{2.4.3 Methods for Solving the Problem}

This problem can be formulated and solved optimally by an Integer Linear Program (ILP). The problem is NP-hard, therefore it is computationally very intensive, i.e., it is nearly impossible to get a solution for a large or even a medium size telecommunication network. Therefore, it is reasonable to search for heuristic methods. Our new heuristic is based on two methods: first, on the Iterative Capacity Splitting (ICS) (see in Section 2.2.2 or [C2]), and second on a randomised method, which is based on Simulated Allocation (SA) (see Section 2.1.4, [5], or [C3]).

\subsection{2.4.4 The proposed heuristic Method: Iterative Availability Enhancement (IAE)}

The flowchart of the Iterative Availability Enhancement algorithm can be seen in Figure 2.15.

- \textbf{First phase.} The first step of the algorithms is to route as many protected paths as possible. This is done by the ICS method that gives approximate results for the NP-complete
problem in polynomial time. In the first phase of ICS working routes are determined so that each link capacity is to be decreased to the half \((w_l - c_l / 2)\). When not all commodities can be routed then increase the working capacity \(w_l\) for each link \(l\) e.g., by 10% until this step succeeds. In the second phase find as many protection routes as possible. The advantage of this method is that it finds working routes for all demands, and as many protection routes as possible.

- **Second phase.** In many practical situations finding a protection path violates capacity constraints. It means that the demands can not be accommodated by available capacities. The question of the second step is how to extend existing links of telecommunication networks in such a way that the extending cost is kept minimal? Costs of a capacity extension can be lower if there is enough fibre installed in the ground, and only the equipment at the ends of the span are to be installed. The cost of extending a link is higher if new cables are to be laid.

For capacity extension a greedy algorithm is proposed: Take the demand with the lowest current availability value \(A_d\). Find a protection route for that commodity without capacity constraints. Repeat this procedure for \(k\) commodities while there is any connection, which has availability \(A_d < A_{min}\) and an independent protection path can be found for it. Calculate new required capacity \((c_l')\) for each link affected by the new paths, and extend all links accordingly e.g., by a new STM-16 in case of an SDH network, but at least by \((c_l' - c_l)\). If some link capacities were increased in this phase, repeat the procedure from the first phase such that links are utilised smoother. Repeating the first phase also takes the advantage that link capacity extension can provide shorter paths.

- **Third phase.** The third step of the algorithm is to protect nodes or regions that are connected to the network by a single link. These links deteriorate the availability value because in case of a failure a whole region is cut from the network. So if it is needed, those regions are connected to the network with an additional link. This step depends strongly on the geographical capability of the region, e.g., location of rivers, railways, and mountains. Many research activities pay considerable attention to the problem of topological design [24] and connectivity augmentation [9]. It can be solved either by algorithmical tools or by human intervention or by combining them.

### 2.4.5 Numerical Example

The performance of the IAE algorithm is investigated on two more or less realistic networks. The first one consists of 26 nodes, which is an extended topology of the European Optical Network (EON). Seven more cities, Budapest, Bratislava, Warsaw, Helsinki, Vilnius, Riga, and Tallinn are added to the network introduced in [64]. The topology can be seen in Figure 2.16. For simplicity reasons demands between all node pairs are of one unit. There are two "leaf" cities: Warsaw and Helsinki that can not be protected. The second network is the modified backbone network of PanTel that is a ring-based SDH one with 90 nodes of which 9 are "leaves".

Considering EON network in the first phase for all 325 commodities can be found a working path but only for 276 of them a protection path. In the second phase \(k=5\) is a good compromise (see Section 2.4.4, Second Phase) and if capacity extension is necessary then it is increased by \(c=10\) units. Six cycles were necessary to obtain protection paths for all commodities except the unprotected links Helsinki-Tallinn and Bratislava-Warsaw. Commodities with end points in
Figure 2.15: Flowchart of the Iterative Availability Enhancement (IAE)
Helsinki or Warsaw are partially protected, i.e., protection routes are searched with end point Tallinn or Bratislava instead.

In the third (topology extension) phase Warsaw has been connected to Berlin while Helsinki to Stockholm. The new configuration was successful for all commodities. The changed link capacities can be seen in Table 2.10. It has been assumed that both the equipment availability and the constant link availability are equal to 99.9999%. Link cuts happen in 300 km once a year and are repaired within at most 4 hours. The increase of minimum (MIN) and average (AVE) availability can be seen in Figure 2.17. In the first four steps the existing edge capacities have been extended. In this phase the minimum availability has been enhanced from 99.7062% to 99.7292% and the average availability from 99.9763% to 99.9805%. The minimal availability has improved significantly in the topology extension phase (Step 5) to 99.9972%. The average availability reached 99.9992%. The algorithm has been applied with different traffic loads. Load1 means that the amount of commodities is increased by 10% and load2 means it is increased by 25% relative to the fully protected instance. The running time (on Pentium 200MHz, 80Mb memory, Windows NT) for EON-29 was 58 seconds for load1 and 66 seconds for load2, while in case of the Hungarian network 83 and 104 minutes for load1 and load2 respectively.

<table>
<thead>
<tr>
<th>Link</th>
<th>Old Cap</th>
<th>New Cap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paris-Zurich</td>
<td>80</td>
<td>130</td>
</tr>
<tr>
<td>Paris-Milan</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>Zurich-Milan</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>Zagreb-Vienna</td>
<td>80</td>
<td>110</td>
</tr>
<tr>
<td>Vienna-Berlin</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>Berlin-Warsaw</td>
<td>0</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>80</td>
</tr>
<tr>
<td>----------------</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>Moscow-Tallinn</td>
<td>80</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 2.10: Changed link capacities on the EON network

Figure 2.17: The increase of minimum (MIN) and average (AVE) availability.

2.4.6 Summary

The availability analysis of SDH and WDM network components and connections has been described. A three-phase, iterative heuristic method has been proposed, that advances the availability of networks to a certain level. The obtained results can be used efficiently for extension of even large-scale networks, consisting of 100 or more nodes. The algorithm runs in polynomial time. The demonstrations on two real-life networks of practical interest have shown that the IAE algorithm guarantees high level of availability while keeping costs as low as possible.
Table 2.4: The Numerical Results: Performance-Comparison of Algorithms for 1+1 and TCA Allocation Strategies

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N5</td>
<td>Time</td>
<td>0.740</td>
<td>0.617</td>
<td>0.021</td>
<td>0.017</td>
<td>0.000:07.31</td>
<td>0.000:03.31</td>
</tr>
<tr>
<td></td>
<td>Total Alloc [%]</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>60.75</td>
<td>60.75</td>
</tr>
<tr>
<td></td>
<td>Working Alloc [%]</td>
<td>28.3</td>
<td>28.3</td>
<td>28.3</td>
<td>28.3</td>
<td>28.3</td>
<td>28.3</td>
</tr>
<tr>
<td>N5 + 11</td>
<td>Time</td>
<td>0.870</td>
<td>0.672</td>
<td>0.028</td>
<td>0.021</td>
<td>0.000:08.95</td>
<td>0.000:03.55</td>
</tr>
<tr>
<td></td>
<td>Total Alloc [%]</td>
<td>91.2</td>
<td>91.2</td>
<td>94.0</td>
<td>94.1</td>
<td>58.24</td>
<td>61.54</td>
</tr>
<tr>
<td></td>
<td>Working Alloc [%]</td>
<td>26.7</td>
<td>26.7</td>
<td>26.9</td>
<td>27.3</td>
<td>26.74</td>
<td>27.47</td>
</tr>
<tr>
<td>N15</td>
<td>Time</td>
<td>0.740</td>
<td>10.8</td>
<td>3.1</td>
<td>2.2</td>
<td>0.000:51.63</td>
<td>0.000:05.46</td>
</tr>
<tr>
<td></td>
<td>Total Alloc [%]</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>53.43</td>
<td>53.43</td>
</tr>
<tr>
<td>N15 + 11</td>
<td>Time</td>
<td>0.870</td>
<td>12.4</td>
<td>3.3</td>
<td>2.4</td>
<td>0.000:58.37</td>
<td>0.000:06.42</td>
</tr>
<tr>
<td></td>
<td>Total Alloc [%]</td>
<td>98.0</td>
<td>98.2</td>
<td>98.6</td>
<td>98.4</td>
<td>51.43</td>
<td>52.85</td>
</tr>
<tr>
<td></td>
<td>Working Alloc [%]</td>
<td>25.3</td>
<td>25.7</td>
<td>25.9</td>
<td>25.8</td>
<td>25.60</td>
<td>26.09</td>
</tr>
<tr>
<td>N25</td>
<td>Time</td>
<td>0.740</td>
<td>11:21</td>
<td>15:31</td>
<td>0.3:50:59</td>
<td>03:13:27</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total Alloc [%]</td>
<td>100</td>
<td>97.9</td>
<td>98.2</td>
<td>98.4</td>
<td>65.23</td>
<td>65.55</td>
</tr>
<tr>
<td></td>
<td>Working Alloc [%]</td>
<td>32.15</td>
<td>31.9</td>
<td>31.9</td>
<td>31.9</td>
<td>31.76</td>
<td>31.76</td>
</tr>
<tr>
<td>N25 + 41</td>
<td>Time</td>
<td>0.870</td>
<td>11:22</td>
<td>1:14</td>
<td>0.4:8:55</td>
<td>03:30:46</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total Alloc [%]</td>
<td>100</td>
<td>88.2</td>
<td>87.4</td>
<td>87.7</td>
<td>53.21</td>
<td>56.54</td>
</tr>
<tr>
<td></td>
<td>Working Alloc [%]</td>
<td>25.7</td>
<td>26.8</td>
<td>29.7</td>
<td>29.6</td>
<td>27.48</td>
<td>28.71</td>
</tr>
<tr>
<td>N25 + 10%</td>
<td>Time</td>
<td>0.740</td>
<td>14:12</td>
<td>2:43</td>
<td>2:14</td>
<td>0.00:56:54</td>
<td>03:05:47</td>
</tr>
<tr>
<td></td>
<td>Total Alloc [%]</td>
<td>88.0</td>
<td>94.2</td>
<td>89.7</td>
<td>89.7</td>
<td>59.28</td>
<td>59.43</td>
</tr>
<tr>
<td></td>
<td>Working Alloc [%]</td>
<td>28.2</td>
<td>28.3</td>
<td>29.16</td>
<td>29.1</td>
<td>29.26</td>
<td>28.92</td>
</tr>
<tr>
<td>N35</td>
<td>Time</td>
<td>0.740</td>
<td>94.7</td>
<td>6.31</td>
<td>5.26</td>
<td>07:28:26</td>
<td>40:50:40</td>
</tr>
<tr>
<td></td>
<td>Total Alloc [%]</td>
<td>-</td>
<td>94.5</td>
<td>93.3</td>
<td>93.3</td>
<td>62.93</td>
<td>59.09</td>
</tr>
<tr>
<td></td>
<td>Working Alloc [%]</td>
<td>36.3</td>
<td>33.5</td>
<td>33.3</td>
<td>33.3</td>
<td>35.85</td>
<td>32.60</td>
</tr>
<tr>
<td>N35 + 11</td>
<td>Time</td>
<td>0.870</td>
<td>25:09</td>
<td>5:45</td>
<td>6:02</td>
<td>07:44:2</td>
<td>38:44:52</td>
</tr>
<tr>
<td></td>
<td>Total Alloc [%]</td>
<td>-</td>
<td>92.8</td>
<td>93.7</td>
<td>92.8</td>
<td>61.93</td>
<td>58.90</td>
</tr>
<tr>
<td></td>
<td>Working Alloc [%]</td>
<td>35.8</td>
<td>33.1</td>
<td>33.76</td>
<td>33.76</td>
<td>35.85</td>
<td>32.43</td>
</tr>
<tr>
<td>N35 + 10%</td>
<td>Time</td>
<td>0.740</td>
<td>24:51</td>
<td>3:15</td>
<td>5:41</td>
<td>08:28:16</td>
<td>38:24:53</td>
</tr>
<tr>
<td></td>
<td>Total Alloc [%]</td>
<td>-</td>
<td>83.5</td>
<td>83.5</td>
<td>83.1</td>
<td>56.16</td>
<td>53.95</td>
</tr>
<tr>
<td></td>
<td>Working Alloc [%]</td>
<td>32.6</td>
<td>30.1</td>
<td>30.1</td>
<td>32.31</td>
<td>29.62</td>
<td>29.62</td>
</tr>
</tbody>
</table>

Table 2.5: Numerical results of a (denser) network with 30 nodes and 63 links

<table>
<thead>
<tr>
<th>n30</th>
<th>Running time [ms]</th>
<th>Success</th>
<th>Cost of all path-pairs [%]</th>
<th>Cost &quot;problematic&quot; [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LP, SOLVE</td>
<td>CPLEX</td>
<td>𝛼 = 1</td>
<td>𝛼 = 5</td>
</tr>
<tr>
<td>Surr</td>
<td>4.02</td>
<td>100</td>
<td>100</td>
<td>102.60</td>
</tr>
<tr>
<td>2D</td>
<td>4.00</td>
<td>100</td>
<td>102.80</td>
<td>100.74</td>
</tr>
<tr>
<td>ILP</td>
<td>46.25</td>
<td>35.08</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>LPR</td>
<td>47.82</td>
<td>32.07</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>SFR</td>
<td>13.79</td>
<td>10.85</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>n35 methods</td>
<td>Running time [ms]</td>
<td>Success [%]</td>
<td>Cost of all path-pairs [%]</td>
<td>Cost &quot;problematic&quot; [%]</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------</td>
<td>-------------</td>
<td>---------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>LP_SOLVE</td>
<td>CPLEX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smurb</td>
<td>3.33</td>
<td>100</td>
<td>100</td>
<td>101.42</td>
</tr>
<tr>
<td>2D</td>
<td>3.11</td>
<td>93</td>
<td>93</td>
<td>101.42</td>
</tr>
<tr>
<td>ILP</td>
<td>25.48</td>
<td>100</td>
<td>100</td>
<td>100.91</td>
</tr>
<tr>
<td>LPR</td>
<td>25.51</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>SFR</td>
<td>9.24</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 2.6: Numerical results of a (sparser) network with 35 nodes and 51 links

<table>
<thead>
<tr>
<th>n400 methods</th>
<th>Running time [ms]</th>
<th>Success [%]</th>
<th>Cost [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP_SOLVE</td>
<td>CPLEX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smurb</td>
<td>721</td>
<td>100</td>
<td>105.41</td>
</tr>
<tr>
<td>2D</td>
<td>720</td>
<td>100</td>
<td>105.41</td>
</tr>
<tr>
<td>ILP</td>
<td>2873</td>
<td>1030</td>
<td>100</td>
</tr>
<tr>
<td>LPR</td>
<td>2855</td>
<td>1053</td>
<td>100</td>
</tr>
<tr>
<td>SFR</td>
<td>787</td>
<td>167</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 2.7: Numerical results of 10 relevant demands in a network with 400 nodes and 1378 links

<table>
<thead>
<tr>
<th>Network</th>
<th>No. Node</th>
<th>No. of edge</th>
<th>Avg. length of working path</th>
<th>Avg. nodal degree</th>
<th>2D success</th>
<th>Normalized time per path-pair</th>
<th>Cost [%]</th>
<th>Cost [%]</th>
<th>Cost [%]</th>
<th>Cost [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILP</td>
<td>14</td>
<td>27</td>
<td>2.43</td>
<td>3.37</td>
<td>93.03</td>
<td>5.32</td>
<td>60.36</td>
<td>46.7</td>
<td>60.36</td>
<td>46.7</td>
</tr>
<tr>
<td>LPR</td>
<td>14</td>
<td>27</td>
<td>2.27</td>
<td>3.37</td>
<td>93.03</td>
<td>5.22</td>
<td>64.80</td>
<td>47.67</td>
<td>64.80</td>
<td>47.67</td>
</tr>
<tr>
<td>SFR</td>
<td>14</td>
<td>27</td>
<td>1.10</td>
<td>3.37</td>
<td>93.03</td>
<td>5.10</td>
<td>5.25</td>
<td>48.0</td>
<td>5.25</td>
<td>48.0</td>
</tr>
</tbody>
</table>

Table 2.8: Percentage of optimal routed demands in the 8 examined networks

44
Chapter 3

Configuration and Protection of Multilayer Networks

Different algorithms for configuration of networks consisting of multiple layers are investigated for both dedicated and shared protection. An iterative algorithm called Multilayer Network Configuration Algorithm (MNCA) is proposed for configuring survivable IP over WDM networks. It is applicable to three protection schemes: Protection at the Lower (e.g., optical) Layer, Protection at the Upper (e.g., IP) Layer, and Protection at Both Layers. Disjoint routing in the upper layer is hard to manage because of two reasons: first, two paths should be disjoint in the lower layer as well; second, they should fit into available lower layer capacities. These problems are investigated and solved by mapping and routing strategies that improve availability by allowing disjoint routing. Throughput and utilization are further improved by a special iteration method. We show that a significant amount of overall capacity (and therefore the number of wavelength channels) can be saved (up to 71% in our experiments) while the network availability is negligibly deteriorated. The algorithms are compared according to their running time, resource (capacity) requirements, connection availability and throughput.

3.1 Introduction

The design, configuration and protection issues of Wavelength Division Multiplexing (WDM) networks are being discussed extensively in the literature, e.g., in [C2, C3, 46, 47, 48]. However, most existing networks are integrated in a multilayer network architecture, that means several combination of the architecture IP-over-ATM-over-SDH-over-WDM. This four layer architecture makes today core network architecture ineffective. The vision "IP directly over WDM" promises the elimination of unnecessary network layers leading to network cost and complexity reduction [6]. Such technology is Multiprotocol Lambda Switching (MPLS) that defines common control plane for the IP and the WDM layer by extending the MPLS concepts to include wavelength-switched lightpaths [49]. This allows sharing of information between the WDM and IP layers while enabling IP traffic to directly access the WDM channels. Therefore, such an IP-over-WDM architecture can effectively coordinate label switched path (LSP) determination and protection to deliver optimal performance [6].

Joint protection requires both layers to be able to handle failures. In MPLS each Label Switching Router (LSR) maintains an entry (next-hop label forwarding entry - NHLFE) for
incoming packets that determines the next hop of the packets. The entries can be changed to reroute traffic from failed path to an alternative path. In the WDM layer fiber- or channel-protection can be implemented by modifying the entries in the wavelength-forwarding table.

The standard bodies work on new IP layer connection oriented technologies for the control of optical networks. Examples areBesides the aforementioned MPAS [49], Generalized MPLS (GMPLS) [50, 51, 52] by IETF\(^1\), G.asn [53] and G.astn [54] by ITU-T, Optical UNI by OIF\(^2\) and UNI by ODSI\(^3\). ANSI\(^4\) and ETSI\(^5\) have similar activities as well. All these activities aim to achieve a uniform control plane for two or multiple network layers. Consequently, the protection tasks can be effectively coordinated between the lower (optical) and the upper (IP) layer [55]; however, cooperation of different types of network layers raises many questions. Some of these questions are related to protection: how to appoint for each failure the layer responsible for its healing; how to make protection mechanisms of different layers interwork; how to route the demands (commodities) in the upper and lower network layers; how to map the links of the upper layer into the lower layer; how to minimize the used resources while providing as high availability as possible [7].

In the literature several problems arising in multilayer networks are discussed. In [56] Bhardwaj constructs an algorithm for the shortest pair of physically-disjoint paths between a given pair of nodes. He takes two special cases of the problem (fork configuration and express link) into account. However, other configurations occur that are not covered by the paper. In [57] a tabu search based algorithm is proposed that maps upper layer links into lower layer channels and prevents three types of failure propagation. However, this method requires to be aware of the routing in the upper layer while our heuristic needs only the topology of the upper layer but not the routing. The reconfiguration problem of multilayer networks is formulated in [58] as an integer linear program (ILP) and is solved by Lagrangian Relaxation. In [59] the ILP formulation of protection at the upper (electrical) and at the lower (optical) layer are given. Three types of interconnection models are described in [60] and [61]: peer model, overlay model and interdomain model. Peer model allows seamless interconnection of IP and optical networks by using common control plane (Figure 3.1). We assume peer model in this Chapter; nevertheless, the algorithms are applicable to the overlay model as well. IP connections are realized by MPLS label switched paths (LSPs) that are mapped into \(\lambda\) paths of the WDM layer.

We investigate three multilayer recovery approaches: Protection at the Upper Layer (PUL), Protection at the Lower Layer (PLL), Protection at Both Layers (PBL). PBL is also known as Common Pool Survivability in case of Shared Protection. We formulate the problems as Integer Linear Programs (ILP) and propose heuristic algorithms based on those we have developed earlier for single-layer networks (see Chapter 2 or [C2, C3]). We propose a simple algorithm for mapping upper layer links into the lower layer (WDM) and show that with the help of this a higher number of disjoint paths can be found in the upper layer. We also show a smart routing method in the upper layer in order to increase the availability between node pairs.

We do not distinguish fibers and cables, i.e., we assume that each link contains one fiber. We assume that each node has wavelength conversion capability, i.e., we can consider all wavelength links of a physical link as one, i.e., we can sum up the capacities of separate wavelength channels of

---

\(^1\)Internet Engineering Task Force, http://www.ietf.org/
\(^2\)Optical Interworking Forum
\(^3\)Optical Domain Service Interconnect
\(^4\)American National Standards Institute
\(^5\)European Telecommunications Standards Institute
each link. Although OXCs with optical core (MEMS\textsuperscript{6}, Bubble Switches\textsuperscript{7}) attract more attention, OXCs with electrical core are cheaper, faster and wider used solution with natural full wavelength conversion capability. The ILP formulation and the MNCA algorithm is applicable to networks without wavelength conversion but this is beyond the scope of this Chapter. All our methods work for both Dedicated and Shared Protection, denoted as DP and SP, respectively. In the literature various terms are used for the IP layer (upper, client and second layer) and for the WDM layer (lower, physical, service, server and first layer). In this Chapter we use the terms upper and lower layer, respectively.

3.2 On Protection

In this section some protection strategies of interest are discussed: Should it be done at the upper or at the lower layer or at both layers? Should it be Dedicated or Shared?

3.2.1 Protection at Different Layers

Protection at the Lower Layer (PLL). In this case the network is protected in the lowest possible layer by equipment of this layer, e.g., the resilience is carried out in the layer as near to the origin of the failure as possible. An IP-over-WDM network scenario of six nodes can be seen in Figure 3.2. In four nodes \((A,B,C,E)\) both WDM and IP equipment can be found while in two nodes (D and F) only WDM equipment. The demand \(a-c\) is routed through nodes \(a-b-c\). The IP link \(a-b\) is realized by the WDM link \(A-B\). Link failures happen always in the lower

\textsuperscript{6}Micro Electro-Mechanical Systems, details at http://www.lucent.com

\textsuperscript{7}Liquid Bubbles, details at http://www.agilent.com
layer: for example if link A-B fails, it is detected by the WDM layer and protected by the path A-D-E-B. Consequently, the WDM layer supplies protected paths for the links of the IP layer.

**Protection at the Upper Layer (PUL).** The second multilayer protection method, called "Protection at the Upper Layer", recovers after failures in the layer closest to the origin of the traffic. Applying this approach failures happening in the lower layer are recovered in the upper layer. In Figure 3.3 IP path a-b-c is protected by IP path a-e-c, for this reason there is no need to protect it in the WDM layer. In case of overlay model the IP node detects that the signal does not arrive within a certain time from the WDM layer. In case of peer model IP node receives a control signal from the common control plane after a failure occurs. Obviously, the working and protection paths should be independent of each other in the lower layer as well.

An exhaustive comparison of the two approaches, according to resilience time and installation costs can be found in [7]. PLL is cheaper and faster, while PUL ensures better granularity that means:

- For demands having different reliability requirements connections of different reliability can be established.
- The uppermost layer protects against all failures in lower layers, thus the cooperation of different layers can be avoided, which ensures simpler functionality.
- Even thousands of LSPs can use a single 2.5 Gb/s WDM link. In case of protection in the upper layer they have to be restored one by one. Although this causes slower operation, these paths utilize the network more efficiently [7].

**Common Pool Survivability (CPS).** Even if failures are protected in the WDM layers the break down of an IP node is to be protected in the IP layer, i.e., spare resource is to be allocated not only in the WDM (lower), but also in the IP (upper) layer in order to protect transit IP routers. These protection paths are realized by protected WDM layer paths. This ensures exaggerated reliability but wastes capacity. This problem is managed by a method called "Common Pool Survivability" [7]. The basic idea of Common Pool Survivability is to treat spare capacity of the upper layer as an extra traffic in the lower layer. The lower layer spare capacity is reused by an upper layer recovery scheme.

One of the most important problems in the field of protection interoperability arises in case of upper layer protection. This is called failure propagation which means the failure of a single span or node may cause the failure of several components in the upper layer (the problem is also discussed in [57]). It may often happen in case of an inconvenient mapping strategy. Consider the mapping of Figure 3.4. It has been configured by simply running a shortest path algorithm: path A-B-E for link A-E and path C-B-E for link C-E. It is clear that none of A-B, B-C or B-E span failures can be recovered in the upper layer. This problem is addressed in this Chapter.

The other group of problems comes from the capacity constraints. The routing of demands is often done in the upper layer while the lower layer capacities should be taken into consideration.

### 3.2.2 Shared vs. Dedicated Protection

As described in Section 1.2 Dedicated Protection (DP) is advantageous since it practically does not need any signalling and it is extremely fast. Unfortunately, it needs a significant amount of resources, almost twice as much as required for working paths. On the other hand, assuming that only a single failure occurs at the moment the protection resources can be shared (SP). In
this case protection paths can share resources (typically span capacities) if their working paths
do not have any common span or node. This leads to resource allocation reduction of around
40% applied in a single layer (Section 2.2.5 or [C2]) and even more (up to 71%) applied in two
layers (Section 3.7.2). Unfortunately, in this case the protection mechanism needs signaling and
it is slower. Both DP and SP work for PLL, PUL and PBL.

The SP method is also known as TCA, i.e., Thrifty Capacity Allocation [C2]. The idea of
this method is that assuming a single span or node failure at a time only, we simulate failures
and calculate the amount of capacity allocation increment for each demand on all spans, before
routing it. This influences not only the allocations but the routing as well. Similar to Chapter
2 we assume the employment of Automatic Protection Switching (APS) to switch from working
to the protection path immediately after a failure.

3.3 Problem Formulation

Here the problem is formulated as a graph-theoretical task here, while in the next Section as
a more exact formulation by an Integer Linear Program (ILP).

In Section 2.2.2 and in [C2] we have presented the Integer Linear Program for a single layer
and here it is extended for the PLL, PUL and PBL approaches.

We consider two network layers: the lower layer and the upper layer but the formulations
and methods could be generalized for more layers and we could use layer $l + 1$ and $l + 2$ as well.
Let $L_1(V_1, E_1)$ and $L_2(V_2, E_2)$ denote the directed graphs of lower and upper layers, respectively, where $V_1$ and $V_2 \subseteq V_1$ are sets of nodes (vertices) while $E_1 \subseteq \{V_1 \times V_1\}$ and $E_2 \subseteq \{V_2 \times V_2\}$ are sets of directed edges (arcs).

Demands are given at the upper layer only, and we have to map them into edges of the lower layer which are disjoint at the lower layer. These demands, denoted by $\alpha_2(i, j, b_{v_2}) \in O_2$ between nodes $i$ and $j$ require bandwidth $b_{v_2}$ along their routes in the upper layer. Analogously, demands of lower layer $\alpha_1(i, j, b_{v_1}) \in O_1$ require $b_{v_1}$ units of capacity from the lower layer capacities.

Although we do know the end-points of lower layer paths (upper layer links) in advance, we do not know the required capacities ($b_{v_1}$) since these depend on the demand routing in the upper layer, i.e., on the optimization process itself.

The objective is to minimize the cost of network resource usage and the output of the algorithm is as follows. In case of PLL the task is to find a working path in the upper layer while guaranteeing a working and a protection paths in the lower layer. In case of PUL the task is to find two paths in the upper layer while in the lower layer we need only working paths. Finally, in the PBL case in both layers two paths are to be found. We use the approach Common Pool Survivability [7] therefore allocating lower layer protection path for the upper layer protection path is not needed, thus the amount of total spare capacity on edge $ij$ is bounded from below by both the lower and the upper layer spare capacity.

### 3.4 Integer Linear Programming (ILP)

We introduce the formulations for the PLL, PUL and PBL approaches, for simplicity reasons for two layers only, but they can be extended for more layers as well. The formulations are given for the case of Dedicated Protection only, while the algorithms in Section 3.5 are proposed for both Dedicated and Shared Protection. For simplicity reasons, span disjointness is required only.

Let $L_1(V_1, E_1)$ and $L_2(V_2, E_2)$ denote the directed graphs of lower and upper layers, respectively, where $V_1$ and $V_2 \subseteq V_1$ are sets of nodes (vertices) while $E_1 \subseteq \{V_1 \times V_1\}$ and $E_2 \subseteq \{V_2 \times V_2\}$ are sets of directed edges (arcs). Let $O_1$ and $O_2$ denote the lower and upper layer demands, respectively.

**Constants:**

- $B_{ij}$ is the capacity of lower layer edge $ij \in E_1$
- $c_{ij}$ is the cost of using lower layer edge $ij \in E_1$
- $c_{kl}$ is the cost of using upper layer edge $kl \in E_2$
- $\alpha_1(k, l) \in O_1$ denotes lower layer demand realizing upper layer edge between nodes $k$ and $l$
- $\alpha_2(k, l) \in O_2$ denotes upper layer demand between nodes $k$ and $l$
- $b_{v_2}$ denotes the bandwidth required for demand $\alpha_2$

Although we do know the end-points of lower layer paths (upper layer links) in advance, we do not know the required capacities since these depend on the demand routing in the upper layer, i.e., on the optimization process itself.

The flow conservation indicator is defined for all nodes $i \in V_1 \cup V_2$ and demands $o \in O_1 \cup O_2$ as follows:

\[
 f(i, o) = \begin{cases} 
 1 & \text{if } i \text{ is the source of } o \\
 -1 & \text{if } i \text{ is the sink of } o \\
 0 & \text{otherwise} 
\end{cases}
\] (3.1)

50
Variables:

The binary flow indicators take value 1 if flow $o$ uses edge $ij$, and 0 otherwise.

- $x_{ij}^o$ working path of lower layer using edge $ij$ for demand $o \in O_1$
- $y_{ij}^o$ protection path of lower layer using edge $ij$ for demand $o \in O_1$
- $x_{kl}^o$ working path of upper layer using edge $kl$ for demand $o \in O_2$
- $y_{kl}^o$ protection path of upper layer using edge $kl$ for demand $o \in O_2$

Objective:

The objective is to minimize the linear combination of the costs using upper and lower layer edges:

Objective:

$$
\text{minimize} \quad (\alpha C_1 + (1 - \alpha) C_2) \quad (3.2)
$$

where

$$
0 \leq \alpha \leq 1 \quad (3.3)
$$

while

$$
C_2 = \sum_{(k,l) \in E_2} \sum_{o \in O_2} c_{kl}^o b_o (x_{kl}^o + y_{kl}^o) \quad (3.4)
$$

and

$$
C_1 = \sum_{(i,j) \in E_1} \sum_{o \in O_1} c_{ij}^o (x_{ij}^o + y_{ij}^o) \quad (3.5)
$$

According to the appropriate protection scheme, $y_{ij}^o$ or $y_{kl}^o$ are omitted. Note, that while $b_o$ is present in equation (3.4) it is not present in (3.5). The reason is that we do not know it in advance. Fortunately, it is not needed at all, since equation (3.4) minimizes the total volume of traffic within the network. If we want to optimize in a fair way, i.e., to ensure the same chances for a traffic flow having lower bandwidth demand as for a traffic demand having higher bandwidth requirement, we will avoid using $b_o$ in equation (3.4) as well.

Constraints for Protection at the Lower Layer (PLL):

- Flow-conservation constraints for working and protection paths in the lower layer

$$
\sum_{m : (i,m) \in E_1} x_{im}^o - \sum_{n : (n,i) \in E_1} x_{nmi}^o = f(i,o_1) \quad (3.6)
$$

$$
\sum_{m : (i,m) \in E_1} y_{im}^o - \sum_{n : (n,i) \in E_1} y_{nmi}^o = f(i,o_1) \quad (3.7)
$$

for all nodes $i \in V_1$ and commodities $o \in O_1$

- Flow-conservation constraints for working paths in the upper layer

$$
\sum_{m : (i,m) \in E_2} x_{im}^o - \sum_{n : (n,i) \in E_2} x_{nmi}^o = f(i,o_2) \quad (3.8)
$$

for all nodes $i \in V_2$ and commodities $o \in O_2$

- Capacity constraints in the lower layer

$$
\sum_{m \in (i,m) \in E_2} \sum_{o \in O_2 (k,l) \in E_2} x_{im}^o (x_{ij}^o (k,l) + y_{ij}^o (k,l)) b_o \leq B_{ij} \quad (3.9)
$$

51
for all links \((i, j) \in E_1\)

- Constraints to ensure path-diversity in the lower layer

\[ x_{1_{ij}} + y_{1_{ij}} \leq 1 \]  \hspace{1cm} (3.10) 

for all links \((i, j) \in E_1\) and commodities \(o \in O_1\)

**Constraints for Protection at the Upper Layer (PUL):**

In this case in the lower layer we need only working paths, therefore variable \(y_{1_{im}}\) will not be used here. Thus, the objective is changed accordingly, and some of the constraints are reformulated as well as follows:

- Flow-conservation constraints for working paths in the lower layer: Equation (3.6).

\[
\sum_{m: (i, m) \in E_2} y_{2_{im}} - \sum_{m: (n, i) \in E_2} y_{2_{ni}} = f(i, o_2) \]  \hspace{1cm} (3.11) 

for all nodes \(i \in V_2\) and commodities \(o_2 \in O_2\)

- Capacity constraints in the lower layer

\[
\sum_{o_2 \in O_2} \sum_{(k, l) \in E_2} (x_{2_{kl}} + y_{2_{kl}}) x_{1_{ij}}^{o_2(k,l)} b_o \leq B_{ij} \]  \hspace{1cm} (3.12) 

for all links \((i, j) \in E_1\)

- Constraints to ensure path-diversity in the lower layer

\[
\sum_{(k, l) \in E_2} (x_{2_{kl}} + y_{2_{kl}}) x_{1_{ij}}^{o_2(k,l)} \leq 1 \]  \hspace{1cm} (3.13) 

for all links \((i, j) \in E_1\) and commodities \(o_2 \in O_2\)

**Constraints for Protection at Both Layers (PBL):**

In this case we use the approach Common Pool Survivability \([7]\) therefore allocating lower layer protection path for the upper layer protection path is not needed. In addition to Equations (3.6, 3.7, 3.8, 3.10, 3.11, 3.13) the following formulations are needed:

\[
\sum_{o_2 \in O_2} \sum_{(k, l) \in E_2} x_{2_{kl}} x_{1_{ij}}^{o_2(k,l)} b_o + P_{ij} \leq B_{ij} \]  \hspace{1cm} (3.14) 

for all links \((i, j) \in E_1\)

\[
\sum_{o_2 \in O_2} \sum_{(k, l) \in E_2} x_{2_{kl}} y_{1_{ij}}^{o_2(k,l)} b_o \leq R_{ij} \]  \hspace{1cm} (3.15) 

52
for all links \((i, j) \in E_1\)

\[
\sum_{e_2 \in C_2} \sum_{(k, l) \in E_2} y_{e_2}^{(k, l, i, j)} x_{1, i, j}^{(k, l)} b_{0, 2} \leq R_{ij}
\]  

(3.16)

for all links \((i, j) \in E_1\)

\(R_{ij}\) denotes the amount of total spare capacity on edge \(ij\) that is bounded from below by both the lower and the upper layer spare capacity.

**Comments:**

For simplicity reason the problems are formulated for Dedicated Protection. For all three approaches the equations of Shared Protection can be formulated as presented in Section 2.2.3 and [C2]. In all three formulation (PLL, PUL, PBL) additional lower layer demands could be added by introducing new variables and adding them to the capacity constraints and flow conservation constraints.

Unfortunately, PLL, PUL and PBL formulations are non-linear. This is because of the AND operation, e.g., upper layer demand \(o_2\) uses upper layer edge \(k\) which is a lower layer path which uses lower layer edge \(ij\) (see Equation 3.12). Now both variables must have value 1 to be sure that demand \(o_2\) is really using lower layer edge \(ij\). It can be written as \(x_{e_2}^{(k, l, i, j)} \cdot x_{1, i, j}^{(k, l)}\). This AND operation (binary product) can be substituted by four Bool-algebraic inequalities as described in Section 2.2.3 and [C2]. This increases the number of constraints significantly, but after this conversion the problems are linear and can be solved by any available solver, e.g., CPLEX, which gives exact solution for small networks.

## 3.5 Multilayer Network Configuration Algorithm (MNCA)

Since the problem is very complex we have to decompose it to simpler subproblems to be solved separately and refined iteratively.

### 3.5.1 Problem Decomposition

Here we describe two decomposition approaches for PLL, PUL and PBL problems.

- **Bottom Up** - Protection at the Lower Layer (PLL ↑), Protection at the Upper Layer (PUL ↑), Protection at Both Layers (PBL ↑): The lower layer is first configured and the upper layer thereafter.

- **Top Down** - Protection at the Lower Layer (PLL ↓), Protection at the Upper Layer (PUL ↓), Protection at Both Layers (PBL ↓): The upper layer is first configured and the lower layer thereafter.

For the Bottom-Up (↑) approach we start with configuring the WDM layer assuming that the demands that are to be mapped to that layer are known but the sizes of the demands are not known. After this, when the embedding (mapping) is determined the IP layer demands can
be routed within the virtual topology obeying lower layer capacity constraints. This is refined iteratively.

For the Top-Down approach (↓) we start with routing the IP layer demands over the virtual topology without knowing the capacity constraints at all, and then try to embed it into the WDM network - repeating all that iteratively.

Most of the above subproblems can be solved by single layer methods [C2, C3, 46, 47, 48]. We have applied SA++ (Section 2.1.4 and [C2]) that worked very well in single layer networks. Complications occur in case of routing in the upper layer while taking mapping and capacity constraints into account.

The proposed algorithm, Multilayer Network Configuration Algorithm (MNCA) uses the top-down method in case of Protection at the Lower Layer (PLL ↓) and the bottom-up method in case of Protection at the Upper Layer (PUL ↑). The reason for this is that in such a way the determination of single paths is done first and the determination of path-pairs later, that is we let more freedom for the algorithm in the second ("harder") part. Let us take the case when we determine lower layer working paths after upper layer working and protection paths. The method for that is to construct a stumped graph from the lower layer network that does not contain links that are used by both the upper layer working and protection paths. It can easily happen that the stamped graph does not contain any path for mapping. Therefore, the bottom-up method is preferred.

Protection at Both Layers can also be implemented by the above methods. The difference to PUL is that two disjoint paths should be allocated in the lower layer as well. Common Pool Survivability [?] is solved by allocating the maximum of the upper and lower layer protection bandwidth. For PBL the Bottom Up approach proved to be more efficient (PBL ↑) in sense of throughput and utilization.

### 3.5.2 Mapping the Upper Layer into the Underlying one (Smart-Mapping)

An algorithm called Smart-Mapping is proposed that constructs a mapping in the lower layer without any knowledge of upper layer routing. This mapping can be refined by the iterative method. The capacities of all demands (representing upper layer links) are set to one unit and the capacities of the lower layer links are increased in each step.

The mapping algorithm is as follows:

- **Step 1.** Set the size of all demand- and link-capacities to 1.
- **Step 2.** Run a single layer configuration algorithm. If it succeeds then STOP otherwise continue.
- **Step 3.** Increase all link capacities by 1, and goto Step 2.

This algorithm minimizes the maximum number of upper layer links that are disconnected due to a single lower layer span failure (since it minimizes the maximum number of demands that use a single lower layer link). It solves the problem depicted in Figure 3.4 and gives a mapping depicted in Figure 3.3. In order to decrease running time link capacities are set to \(|D|/|L|\) \(|D|\) and \(|L|\) denote the number of demands and links in the network, respectively since at least so much capacity is needed on each link even if all demands are realized on a single span. After determining the mapping, upper layer demands are to be routed taking this mapping into consideration.
3.5.3 Disjoint Routing in the Upper Layer

One efficient way of enhancing availability is to find two physically disjoint paths. In this Subsection we deal with disjoint routing of one upper layer demand. This is not as simple as in case of one layer because both the mapping and the lower layer capacity constraints are to be taken into consideration.

Routing in the upper layer can fail because of two reasons: (1) Due to capacity constraints: In this case the routing fails because it cannot be routed within available lower layer capacities. (2) Due to disjointness: In this case the routing fails because there exists no pair of paths which are disjoint in the lower layer. Capacity constraints are always to be respected but disjointness constraints can be avoided in order to achieve higher throughput. In the later case the price for higher throughput is the deterioration of availability.

First, all edges are temporarily deleted that would be overloaded if they would be used by the demand. It can be proved that this problem is algorithmically very complex (NP-hard). We show four methods that solve this problem. Typically, an algorithm that finds solution more frequently has longer running time. ILP gives optimal solution but its running time can be very long (exponential). The other three methods (LR, 2D, 2D+) give solution in shorter (polynomial) time but the existence of a solution is not guaranteed even if it exists.

**Integer Linear Programming (ILP)**

The ILP method always finds a solution if it exists. In the formulation for a single commodity \( o \) the following notation is used. Let \( b_i \) denote the size (required bandwidth) of demand \( o \) and let \( m_{ij}^k \) indicate whether the mapping of upper layer edge \( (k,l) \in E \) uses lower layer edge \( (i,j) \in E_1 \) (0/1). The binary flow indicators \( x_{ik}^o \), \( y_{ik}^o \) and \( f(i,o) \) has been defined in Section 3.4.

The ILP formulation is as follows:

\[
\sum_{m: (i,m) \in E_2} x_{mk}^o - \sum_{n: (n,i) \in E_2} x_{nm}^o = f(i,o) \tag{3.17}
\]

\[
\sum_{m: (i,m) \in E_2} y_{mk}^o - \sum_{n: (n,i) \in E_2} y_{nm}^o = f(i,o) \tag{3.18}
\]

for all nodes \( i \in V_2 \)

\[
\sum_{(k,l) \in E_2} (x_{ik}^o + y_{ik}^o) m_{ij}^k \leq 1 \tag{3.19}
\]

for all links \( (i,j) \in E_1 \)

Equations (3.17) and (3.18) are the well-known flow-conservation constraints for working and protection paths. Constraint (3.19) ensures ensure path-diversity in the lower layer. Capacity constraints are not needed because all edges \( e \in E_2 \) have been temporarily deleted that would be overloaded if they would be used by the demand.

**Linear Relaxation (LR)**

In this case the integer condition of the ILP formulation is relaxed, i.e., a linear program is solved. It has shorter (polynomial) running time but some variables are to be rounded.
Two Shortest Path Algorithms (2D)

This method uses Dijkstra's shortest path algorithm two times. After finding the working path using Dijkstra's algorithm (or one of the other shortest path algorithms), delete all upper layer links temporarily that use same physical span(s) as the working paths (this can be decided on the basis of the mapping). Finally, run the shortest path algorithm again on the stamped graph. Although this method is very fast, it may happen that it does not yield solution even if there is one, namely if the routing of the first path blocks the routing of the second one.

Two Shortest Path Algorithms with Improvements (2D+)

Some heuristics can be applied to improve 2D described in the previous subsection. First, for each edge $e \in E_2$ set length $l_e$ to the number of links that are physically dependent to edge $e$ and run Dijkstra's algorithm with these lengths. By this way the number of blocked edges is minimized in the second step. The second improvement is the following. If the second path is blocked by the first one then the first path is changed so that the blocking in the second step will be avoided as far as possible. Accordingly, running the shortest path algorithm the second time will succeed with a higher probability.

Routing in the upper layer can fail because of two reasons: (1) Due to capacity constraints: In this case the routing fails because it cannot be routed within available lower layer capacities. (2) Due to disjointness: In this case the routing fails because there exists no pair of paths which are disjoint in the lower layer. MNCA solves these problems by using Smart-Mapping, Smart-Routing (ILP) and iterations. Capacity constraints are always to be respected but disjointness constraints can be avoided in order to achieve higher throughput. In the later case the price for higher throughput is the deterioration of availability.

MNCA is summarized for the three protection strategies in the following two subsections.

3.5.4 Iterating the PLL Top-Down (↓) Method

The iteration of PLL (Protection at the Lower Layer) top-down (↓) method is as follows:

- **Step 1.** First, set the cost of all upper layer edges to one unit.

- **Step 2.** In each iteration after running the algorithm described in Section 3.5.1 (PLL ↓) two cases can occur: (1) All lower layer demands could be allocated or (2) some lower layer demands could not be allocated. In the first case the algorithm stops with success while in the second case it proceeds to Step 3.

- **Step 3.** Allocate all unallocated lower layer demands waving capacity constraints. Multiply cost of all upper layer edges $e_2$ by $i_{e_2}$ defined as follows. Let $load_{e_1}$ denote the load or utilization of edge $e_1$ relative to the capacity of edge $e_1$. Let $load_{e_2}$ be the maximum of $load_{e_1}$ for all $e_1$ that is used by the upper layer edge $e_2$. In this way the load of an upper layer edge $e_2$ has been defined. We propose $i_{e_2} = \tanh(a load_{e_2} - b) + c$ (See Figure 3.6) where $\tanh$ is the hyperbolic tangent function. For the numerical study we have used $a=b=3$ and $c=1$.

This function has the following properties: its value is near zero when $load_{e_2}$ is near to zero, it tends to 2 when the edge load tends to infinity, it is continuous, its differential is also continuous, and it decreases the cost of edges that have low utilization and increases the cost of over-utilized edges in order to force paths to low-utilized edges in the next iteration.
3.5.5  Iterating the PUL and PBL Bottom-Up (↑) Method

The iteration of PUL (Protection at the Upper Layer) bottom-up (↑) method is very similar to that of PLL↓ but in this case the algorithm performs the refinement of a possibly bad mapping as well.

• **Step 1.** First, set the cost of all lower layer edges to one unit.

• **Step 2.** In each iteration after running the algorithm described in Section 3.5.1 (PUL ↑) two cases can happen: (1) All upper layer demands could be allocated or (2) some upper layer demands could not be allocated. In the first case the algorithm stops with success while in the second case it proceeds to Step 3.

• **Step 3.** Allocate all unallocated upper layer demands waving capacity or disjointness constraints. Multiply cost of all lower layer edges e1 by \( i_{e1} \) that is defined as follows.

\[
i_{e1} = \tanh(a \cdot \text{load}_{e1} - b) + c \cdot \text{bad mapping of } e1 \text{ where } \text{bad mapping of } e1 \text{ is the number of upper layer demands that have no other choice than to use } e1 \text{ more than once (i.e., both the working and the protection path use } e1). \text{ Consequently, in the next iteration low-utilized and "less-used" edges are preferred.}
\]

• **Step 4.** Go to Step 2.

The iteration of the PBL (Protection at Both Layers) method is the same as that of PUL.

3.5.6  Time Complexity

The running time of MNCA depends on the time complexity of the applied single layer method. We have applied SA++ (Section 2.1.4, [C2]) that has a time complexity of \( O(|V|^2 |D|^2) \), where \(|V|\) represents the number of nodes and \(|D|\) the number of demands in the network. The method of Smart-Mapping (Section 3.5.2) calls the one layer method with \(|E|^2\) demands at most \(|E|^2\) times, i.e., its time complexity is \( O(|V|^2 |E|^2) \). In the 2D algorithm (Section 3.5.3) after running the Dijkstra's algorithm (\( O(|V|^2 |E|) \)), all lower-upper layer link pairs are to be examined.
\(O([E_2||E_1]|), \text{ and Dijkstra's algorithm is to be run again, thus the total time complexity is}\ O([V_2]^2 + [E_2||E_1]).\)

Running PLL top-down method (Section 3.5.4) in each iteration two single layer problems are solved. The number of iterations is a small constant number (e.g., 5 or 10), i.e., the time complexity is the same as that of the used single layer method \(O([V_1]^2|D|^2)\) in this case. Considering PUL and PBL bottom-up (Section 3.5.5) methods in each iteration a single layer problem is solved for the lower layer, and one in the upper layer (by ILP or 2D). Using 2D, the time complexity will be \(O([V_2]^2 + [E_2||E_1]|D|^2)\).

### 3.6 Availability Evaluation and Advance of Multilayer Networks

In Section 2.4 we have proposed a general and fast heuristic method that advances the availability of a network over an aimed level. In this Section the evaluation and advance of multilayer networks availability is addressed. Protection schemes of multilayer networks are introduced and methods are proposed that evaluate the availability of a multilayer network and advance it over an aimed level (e.g., 99.9999\%). We propose methods to evaluate the availability of multilayer networks in case of PLL, PUL and PBL. An iterative method is proposed that advances the availability of multilayer networks over an aimed level.

#### 3.6.1 Availability Analysis

We give a short overview how the availability of a multilayer network can be calculated. This is an extension of the analysis described in the previous Section for single layer networks. Availability analysis of PLL, PUL and PBL multilayer networks are discussed.

**Availability of PLL**

In this subsection all equipment and paths are projected into the lower layer and the calculations done at one layer (Figure 3.7). Three (partly independent) paths will occur: the working path (WP) the lower layer protection path (PLL path) and the upper layer protection path (PUL path).

![Figure 3.7: Project two layers into one](image)

In case of PLL there exist both lower and upper layer equipment that are common in the working and protection paths. (See Figure 3.2: paths a-b-c and A-D-E-B-E-F-C are common in equipment b and end-points a and e) The common equipment are the end-points of upper layer
links. This can be calculated by Equation (2.24) where $A_c$ is the the product of availabilities of all common equipment and $A_i$ is the separate availability of WP and PLL path described above.  

The following remark simplifies the calculation: in case of PLL the availability is mostly influenced by the unprotected equipment (e.g., node b in Figure 3.2), i.e., it can be approximated by their product.

**Availability of PUL**

In this case the working and protection paths are independent of each other. That means parallel configuration (Equation (2.26)) can be applied. (See Figure 3.3: paths a-b-c and a-e-c)

**Availability of PBL**

In this case the difficulty is that there are three paths that can be common in several equipment and spans: the working path, the lower layer protection path (PLL) and the upper layer protection path (PUL). See Figure 3.8 for an illustrative example. The availability calculation is decomposed into two parts. Let the set of non-end equipment and spans used by both the working and any PLL path called “common elements” (node marked with ‘X’ in Figure 3.8). The source and destination equipment are not counted into this set. Let $A^{WP+PLL}$ denote the probability that every “common element” is in ‘up’ state.

$$A^{WP+PLL} = \prod_{i \in WP \text{ and } i \in PLL \text{ and } i \text{ endpoints}} A_i$$

In the first case we assume that at least one of the ”common elements” is in ‘down’ state while in the second one we assume that all of them are in ‘up’ state.

- If any of the “common elements” are in ‘down’ state ($1 - A^{WP+PLL}$) then both the working and the lower layer protection paths are affected by the failure, consequently, merely the PUL path can be alive ($A^{PUL}$). Consequently, the availability of this case is: $(1 - A^{WP+PLL}) A^{PUL}$.

- In the other case all equipment used by both the Working and any PLL paths are in ‘up’ state ($A^{WP+PLL}$). Let $A_1^{WP}$ denote availability of the WP assuming that all ”common elements” are in ‘up’ state (i.e., the availability of the ”common elements” are 1). Similarly, let $A_1^{PLL}$ denote the availability of PLL path assuming that ”common elements” are in ‘up’ state. Let $A_1^{PLL+PUL}$ denote the common availability of $A_1^{PLL}$ and $A^{PUL}$. This can be calculated by the parallel-series method, described in the previous section (Equation (2.24)). In this case WP is independent both from PLL and PUL paths. Consequently, the availability assuming that all ”common elements” are in ‘up’ state is a parallel configuration of the working and the union of the PLL and PUL paths: $\text{PARALLEL}(A_1^{WP}, A_1^{PLL+PUL})$.

According to the above considerations, the availability of PBL can be calculated by this decomposition:

$$A^{PBL} = (1 - A^{PLL+PUL}) A^{PUL} + (A^{PLL+PUL}) \text{PARALLEL}(A_1^{WP}, A_1^{PLL+PUL})$$

(3.21)
3.6.2 Availability Enhancement

A general and fast heuristic method is proposed that advances the availability of a multilayer network over an aimed level\textsuperscript{8}, $A_{aim}$. The proposed method, Multilayer Availability Enhancement (MAE) is a three phase, iterative heuristic method that is based on the Iterative Availability Enhancement (IAE) proposed for single layer networks in Subsection 2.4.4.

- **First phase.** Similar to the single layer case, the first step of the algorithm is to route as many protected paths as possible within existing capacities. However, in each step of the ICS algorithm the infeasible links, i.e., that would cause overload in the lower layer network, are to be temporarily erased. Furthermore, disjoint routing is carried out in the upper layer with using one of the methods described in Subsection 3.5.3.

- **Second phase.** This phase, as explained in Subsection 2.4.4, is for extending the available links of the network. For capacity extension a greedy algorithm is proposed: Take the demand in the upper layer with the lowest current availability value ($A_d$). Find a protection route (see Subsection 3.5.3) for that commodity without considering the capacity constraints of the lower layer. Repeat this procedure for $k$ commodities while there is any connection that has availability $A_d < A_{aim}$ and an independent protection path can be found for it. Map the protection paths into the lower layer and calculate new required capacity ($c'_e$) for each lower layer link used by the new paths, and extend all links by at least $c'_e - c_e$. Finally, go to the first phase in order to utilize links smoother.

- **Third phase.** In the third phase new links are to be installed (if needed). Naturally, the new links are set up in the lower layer, which causes that the upper layer graph will be also denser. We propose to install new links where the disjoint algorithm (see Subsection 3.5.3) has failed, for example, if two regions are connected by a single link. The disjoint routing algorithm succeeds with higher probability if the graph is denser.

\textsuperscript{8}e.g., 99.99% or 99.999999%

---

Figure 3.8: In case of PBL there exist three "partly" disjoint paths: working path, upper layer protection path (PUL path) and lower layer protection path (PLL path)


3.7 Performance of the Algorithms

3.7.1 Networks

We have investigated four networks N5, N15, N25 and N35 consisting of 5, 15, 25 and 35 WDM nodes respectively (solid and broken lines in Figure 3.9). The WDM layer of these networks were optimally designed and configured by a method developed earlier. The IP layer of N5 consists of three more links compared to the WDM layer. N15 has a centralized IP network, while in N25 and N35 some nodes has been deleted and some links added randomly (solid and dashed lines in Figure 3.9). As mentioned earlier we consider demands in the IP layer only because WDM demands have to be routed and protected in the WDM layer anyway. Some additional details of the networks are shown in Table 3.1.

![Diagram of networks N5, N15, N25 and N35]

Figure 3.9: The examined networks: N5, N15, N25 and N35

3.7.2 Numerical Results

The methods are compared according to 6 criteria:

- computational time [Time]
<table>
<thead>
<tr>
<th>Networks</th>
<th>No. of WDM nodes</th>
<th>No. of WDM links</th>
<th>No. of IP nodes</th>
<th>No. of IP links</th>
<th>No. of IP demands</th>
</tr>
</thead>
<tbody>
<tr>
<td>N5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>N15</td>
<td>15</td>
<td>15</td>
<td>14</td>
<td>26</td>
<td>91</td>
</tr>
<tr>
<td>N25</td>
<td>25</td>
<td>31</td>
<td>20</td>
<td>40</td>
<td>190</td>
</tr>
<tr>
<td>N35</td>
<td>35</td>
<td>51</td>
<td>30</td>
<td>59</td>
<td>435</td>
</tr>
</tbody>
</table>

Table 3.1: The 4 examined networks: Number of nodes, links and IP demands

- network utilization in total [Utilization]
- network utilization introduced by working paths only [Working Util.]
- network throughput [Throughput]
- Average of all path availabilities [Avail (Ave)]
- Minimum of all path availabilities [Avail (Min)]

The tests were carried out on an 500 MHz Intel PentiumIII based computer with 64 Mbytes of memory. ILOG’s CPLEX 6.5 has been used for ILP tasks.

Figure 3.10 shows that the Smart-Map algorithm (See Section 3.5.2) works better than SPOO that maps the demands on the shortest paths one-by-one. The improvement is significant on larger mesh networks (85.34% versus 58.6% in our N35 test network) since in smaller and sparser networks the mapping is nearly trivial.

Figure 3.10: Percentage of node-pairs with physically disjoint path-pairs after one-by-one mapping (SPOO) and Smart-Mapping (SMART)

In Figure 3.11 diverse routing at the upper layer is considered (See Section 3.5.3). It is shown that taking mapping into consideration (SMART) yields higher throughput than the traditional 2D (SPOO) algorithm (see Section 3.5.3), e.g., 85.34% versus 43.89% in N35.
Figure 3.11: Percentage of node-pairs where 2D (SPOO) and Smart-Routing (SMART) can find physically disjoint pair of paths.

In Figure 3.12 it is shown that iteration increases the number of allocated demands (See Section 3.5.4). It shows in case of PLL how for N15 the throughput increases. It is 75% after the first while 100% after the third iteration.

Figure 3.12: Throughput after iteration 1, 2, 3 in case of PLL for the 15-node network (N15)

In Figure 3.13 the network normalized utilization of different protection strategies are compared: PLL, PUL and PBL; and in each case both DP and SP. In this case the networks are assumed not to have capacity constraints. In case of Dedicated Protection PBL handles capacity in a very inefficient way; PUL is the most efficient strategy. In case of Shared Protection PUL yields best utilization while PLL and PBL needs slightly more capacity. It can be seen that the reservation of common space for protection for WDM and IP layer proved to be very efficient; it needs hardly more capacity than protecting only in the upper layer.
The detailed results are presented in Tables 3.2, 3.3, and 3.4. Two algorithms Shortest Path One-by-One (SPOO) and Multilayer Network Configuration Algorithm (MNCA) were compared in every combination of two protection strategies (DP and SP), three multilayer protection schemes (PLL, PUL, PBL), and four test networks (N5, N15, N25, N35). In case of PUL and PBL the case is also considered where the physical jointness of the two paths is enabled (and minimized) if physical disjointness is not possible (for N25 and N35).

**Time**

PLL has always shorter running time than PUL. While for PLL MNCA has about 2 times longer running time than SPOO, it has about 10 times longer running in case of PUL and PBL. SP has longer running time than DP. However, in some cases MNCA is faster than SPOO and SP faster than DP (e.g., Table 3.2 for N35). This occurs if the throughput is 100% since in this case MNCA stops to continue iterating. In case of PUL and PBL the algorithm spends most of the time solving the ILP in order to achieve higher throughput. Faster combinatorial algorithms are under development that will decrease the running time significantly.

**Utilization**

The working utilization has very similar values for Dedicated and Shared Protection, except when the throughput is significantly less than 100%. For PBL the working utilization contains the working path of both the upper layer working and protection path in case of DP.

Applying MNCA with SP saves 57-71% capacity in case of PBL, 15-40% in case of PUL, and 30-46% in case of PLL opposed to DP. The large difference between total utilizations for DP and SP is due to the large difference in capacity requirements of their protection paths. In cases where the throughput is 100%, i.e., all demands could be allocated, the utilization of PUL is less than that of PLL, especially in case of larger networks. However, if throughput is 100% then the utilization of MNCA can be worse than that of SPOO since MNCA enhances the availability.
using disjoint paths that can be longer than the joint paths. MNCA improves throughput that
necessarily requires more capacity (e.g., Table 3.3, third row, N25).

**Throughput**

MNCA gives always higher throughput than SPOO: up to 33% better in case of PUL and
PBL while more than two times better in case of PLL. For Shared Protection particularly for
larger mesh networks PLL gives better throughput. For ring networks PUL seems to be slightly
better. For Dedicated Protection PUL is always better than PLL, particularly for larger mesh
networks (because of the granularity of PUL method).

The ratio of utilization over throughput characterizes the algorithm well. The reason for less
than 100% throughput is either a capacity or a disjointness constraint or both. It can be seen in
the tables that SP always increases throughput.

**Availability**

The values of availabilities are shown just from the fifth digit in order to avoid redundant
9-digits. For example 99.959 in the Tables 3.2, 3.3, 3.4 means 0.9999959 or 99.999959%. In case
the throughput is less than 100% we put the availability values into parentheses because it can
be misleading since worse algorithms allocate merely the most reliable connections, accordingly
the availabilities will be higher. It is suggested to compare availabilities of 100% throughput.
In our experiments the following availabilities were assumed for comparing the methods: link
availabilities: 99.9999%, WDM nodes: 99.99999%, IP-over-WDM nodes: 99.999981%. Please
note that the upper limit of the availabilities is 9992 which is the availability of the two end-points.
The network availability improves when applying any of the protection schemes but enabling
physically disjointness (Tables 3.3 and 3.4, two lowermost rows) slightly deteriorates availability.
The availability values of PBL are excellent because they are always larger than 996193, i.e.,
dunted to the upper bound. Otherwise, comparing PUL and PLL according to availability, it can
be stated that if there exist physically disjoint paths then PUL is better, otherwise PLL is more
advantageous.

**Availability Enhancement**

To demonstrate the Multilayer Availability Enhancement (MAE) we use the network N15A
that is a modification of N15 and depicted on Figure 3.14. We assume that the aimed level of
availability is $A_{aim} = 99.9992\%$, i.e., 920000 in Table 3.5. However, the current value is 520011
(Table 3.5, first row). The availability values after the Second Phase are the same as after the
First (Table 3.5). In the Third Phase a new link is installed (thick line in Figure 3.14), and
the average value increases from 741432 to 888610, while the minimum value increases from
520011 to 730003. After repeating the First and Second Phases, link capacities are increased,
and consequently, the availability is increasing till it exceeds the aimed value and reaches 939990
(i.e., 99.99939990%).

**3.8 Summary**

This Chapter jointly considers routing and protection issues of IP over WDM networks.
The proposed methods, Integer Linear Programming and Multilayer Network Configuration Al-
gorithm (MNCA) were presented for two layers of GMPLS but they can be applied to other
<table>
<thead>
<tr>
<th>Networks</th>
<th>Performance Measure</th>
<th>Dedicated Protection</th>
<th>Shared Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SPOO</td>
<td>MNCA</td>
</tr>
<tr>
<td>N5</td>
<td>Time [s]</td>
<td>0.03</td>
<td>0.06</td>
</tr>
<tr>
<td></td>
<td>Utilization [%]</td>
<td>94.3</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Working Util [%]</td>
<td>22.3</td>
<td>22.6</td>
</tr>
<tr>
<td></td>
<td>Throughput [%]</td>
<td>79.4</td>
<td>84.1</td>
</tr>
<tr>
<td></td>
<td>Avail (Ave) [%]</td>
<td>(9959)</td>
<td>(9959)</td>
</tr>
<tr>
<td></td>
<td>Avail (Min) [%]</td>
<td>(9943)</td>
<td>(9943)</td>
</tr>
<tr>
<td>N15</td>
<td>Time [s]</td>
<td>0.15</td>
<td>0.27</td>
</tr>
<tr>
<td></td>
<td>Utilization [%]</td>
<td>98.1</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Working Util [%]</td>
<td>22.2</td>
<td>24.1</td>
</tr>
<tr>
<td></td>
<td>Throughput [%]</td>
<td>68.4</td>
<td>69.7</td>
</tr>
<tr>
<td></td>
<td>Avail (Ave) [%]</td>
<td>(9951)</td>
<td>(9948)</td>
</tr>
<tr>
<td></td>
<td>Avail (Min) [%]</td>
<td>(9943)</td>
<td>(9943)</td>
</tr>
<tr>
<td>N25</td>
<td>Time [s]</td>
<td>0.4</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>Utilization [%]</td>
<td>77.0</td>
<td>96.9</td>
</tr>
<tr>
<td></td>
<td>Working Util [%]</td>
<td>19.1</td>
<td>20.2</td>
</tr>
<tr>
<td></td>
<td>Throughput [%]</td>
<td>59.3</td>
<td>70.5</td>
</tr>
<tr>
<td></td>
<td>Avail (Ave) [%]</td>
<td>(9943)</td>
<td>(9942)</td>
</tr>
<tr>
<td></td>
<td>Avail (Min) [%]</td>
<td>(9905)</td>
<td>(9886)</td>
</tr>
<tr>
<td>N35</td>
<td>Time [s]</td>
<td>1.7</td>
<td>3.6</td>
</tr>
<tr>
<td></td>
<td>Utilization [%]</td>
<td>89.3</td>
<td>86.1</td>
</tr>
<tr>
<td></td>
<td>Working Util [%]</td>
<td>21.6</td>
<td>18.1</td>
</tr>
<tr>
<td></td>
<td>Throughput [%]</td>
<td>13.9</td>
<td>33.1</td>
</tr>
<tr>
<td></td>
<td>Avail (Ave) [%]</td>
<td>(9942)</td>
<td>(9941)</td>
</tr>
<tr>
<td></td>
<td>Avail (Min) [%]</td>
<td>(9905)</td>
<td>(9886)</td>
</tr>
</tbody>
</table>

Table 3.2: Numerical results for Protection at the Lower Layer
<table>
<thead>
<tr>
<th>Networks</th>
<th>Performance Measure</th>
<th>Dedicated Protection</th>
<th>Shared Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time [s]</td>
<td>SPOO</td>
<td>MNCA</td>
</tr>
<tr>
<td>N5</td>
<td></td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>Utilization [%]</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Working Util [%]</td>
<td>28.3</td>
<td>28.3</td>
</tr>
<tr>
<td></td>
<td>Throughput [%]</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Avail (Ave) [%]</td>
<td>99619993</td>
<td>99619993</td>
</tr>
<tr>
<td></td>
<td>Avail (Min) [%]</td>
<td>99619993</td>
<td>99619993</td>
</tr>
<tr>
<td>N15</td>
<td>Time [s]</td>
<td>0.6</td>
<td>12.1</td>
</tr>
<tr>
<td></td>
<td>Utilization [%]</td>
<td>63.25</td>
<td>84.33</td>
</tr>
<tr>
<td></td>
<td>Working Util [%]</td>
<td>22.9</td>
<td>30.0</td>
</tr>
<tr>
<td></td>
<td>Throughput [%]</td>
<td>75</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Avail (Ave) [%]</td>
<td>(99619926)</td>
<td>(99619926)</td>
</tr>
<tr>
<td></td>
<td>Avail (Min) [%]</td>
<td>(99619926)</td>
<td>(99619926)</td>
</tr>
<tr>
<td>N25</td>
<td>Time [s]</td>
<td>2.2</td>
<td>226</td>
</tr>
<tr>
<td></td>
<td>Utilization [%]</td>
<td>34.7</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>Working Util [%]</td>
<td>11.9</td>
<td>16.9</td>
</tr>
<tr>
<td></td>
<td>Throughput [%]</td>
<td>60.1</td>
<td>77.1</td>
</tr>
<tr>
<td></td>
<td>Avail (Ave) [%]</td>
<td>(9944)</td>
<td>(9948)</td>
</tr>
<tr>
<td></td>
<td>Avail (Min) [%]</td>
<td>(9843)</td>
<td>(9843)</td>
</tr>
<tr>
<td>N35</td>
<td>Time [s]</td>
<td>9.6</td>
<td>861</td>
</tr>
<tr>
<td></td>
<td>Utilization [%]</td>
<td>41.2</td>
<td>48.9</td>
</tr>
<tr>
<td></td>
<td>Working Util [%]</td>
<td>13.5</td>
<td>19.5</td>
</tr>
<tr>
<td></td>
<td>Throughput [%]</td>
<td>49.6</td>
<td>58.6</td>
</tr>
<tr>
<td></td>
<td>Avail (Ave) [%]</td>
<td>(9949)</td>
<td>(9951)</td>
</tr>
<tr>
<td></td>
<td>Avail (Min) [%]</td>
<td>(9843)</td>
<td>(9843)</td>
</tr>
</tbody>
</table>

Jointness enabled

<table>
<thead>
<tr>
<th>Networks</th>
<th>Performance Measure</th>
<th>Dedicated Protection</th>
<th>Shared Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time [s]</td>
<td>1.292</td>
<td>20.159</td>
</tr>
<tr>
<td>N25</td>
<td>Utilization [%]</td>
<td>56.12</td>
<td>56.49</td>
</tr>
<tr>
<td></td>
<td>Working Util [%]</td>
<td>20.30</td>
<td>19.58</td>
</tr>
<tr>
<td></td>
<td>Throughput [%]</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Avail (Ave) [%]</td>
<td>9834</td>
<td>9915</td>
</tr>
<tr>
<td></td>
<td>Avail (Min) [%]</td>
<td>9029</td>
<td>9386</td>
</tr>
<tr>
<td>N35</td>
<td>Time [s]</td>
<td>6.079</td>
<td>246.33</td>
</tr>
<tr>
<td></td>
<td>Utilization [%]</td>
<td>88.94</td>
<td>82.00</td>
</tr>
<tr>
<td></td>
<td>Working Util [%]</td>
<td>36.19</td>
<td>33.43</td>
</tr>
<tr>
<td></td>
<td>Throughput [%]</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>Avail (Ave) [%]</td>
<td>9620</td>
<td>9676</td>
</tr>
<tr>
<td></td>
<td>Avail (Min) [%]</td>
<td>8058</td>
<td>7958</td>
</tr>
</tbody>
</table>

Table 3.3: Numerical results for Protection at the Upper Layer (in the two rows below physically jointness of the two paths is enabled)
<table>
<thead>
<tr>
<th>Networks</th>
<th>Performance Measure</th>
<th>Dedicated Protection</th>
<th>Shared Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SPOO</td>
<td>MNCA</td>
</tr>
<tr>
<td>N5</td>
<td>Time [s]</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>Utilization [%]</td>
<td>78.87</td>
<td>97.74</td>
</tr>
<tr>
<td></td>
<td>Working Util [%]</td>
<td>33.96</td>
<td>43.40</td>
</tr>
<tr>
<td></td>
<td>Throughput [%]</td>
<td>33.96</td>
<td>43.40</td>
</tr>
<tr>
<td></td>
<td>Avail (Ave) [%]</td>
<td>(9962)</td>
<td>(99619999)</td>
</tr>
<tr>
<td></td>
<td>Avail (Min) [%]</td>
<td>(9962)</td>
<td>(99619993)</td>
</tr>
<tr>
<td>N15</td>
<td>Time [s]</td>
<td>0.290</td>
<td>1.592</td>
</tr>
<tr>
<td></td>
<td>Utilization [%]</td>
<td>99.40</td>
<td>99.40</td>
</tr>
<tr>
<td></td>
<td>Working Util [%]</td>
<td>47.76</td>
<td>47.76</td>
</tr>
<tr>
<td></td>
<td>Throughput [%]</td>
<td>56.64</td>
<td>56.64</td>
</tr>
<tr>
<td></td>
<td>Avail (Ave) [%]</td>
<td>(99619954)</td>
<td>(99619964)</td>
</tr>
<tr>
<td></td>
<td>Avail (Min) [%]</td>
<td>(99619925)</td>
<td>(99619925)</td>
</tr>
<tr>
<td>N25</td>
<td>Time [s]</td>
<td>1.242</td>
<td>195.64</td>
</tr>
<tr>
<td></td>
<td>Utilization [%]</td>
<td>38.95</td>
<td>45.89</td>
</tr>
<tr>
<td></td>
<td>Working Util [%]</td>
<td>15.89</td>
<td>19.85</td>
</tr>
<tr>
<td></td>
<td>Throughput [%]</td>
<td>29.21</td>
<td>25.30</td>
</tr>
<tr>
<td></td>
<td>Avail (Ave) [%]</td>
<td>(99619977)</td>
<td>(99619988)</td>
</tr>
<tr>
<td></td>
<td>Avail (Min) [%]</td>
<td>(99619921)</td>
<td>(99619925)</td>
</tr>
<tr>
<td>N35</td>
<td>Time [s]</td>
<td>4.637</td>
<td>1389.488</td>
</tr>
<tr>
<td></td>
<td>Utilization [%]</td>
<td>48.66</td>
<td>44.33</td>
</tr>
<tr>
<td></td>
<td>Working Util [%]</td>
<td>19.3</td>
<td>18.24</td>
</tr>
<tr>
<td></td>
<td>Throughput [%]</td>
<td>28.38</td>
<td>43.19</td>
</tr>
<tr>
<td></td>
<td>Avail (Ave) [%]</td>
<td>(99619941)</td>
<td>(99619941)</td>
</tr>
<tr>
<td></td>
<td>Avail (Min) [%]</td>
<td>(99619743)</td>
<td>(99619801)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Networks</th>
<th>Performance Measure</th>
<th>Dedicated Protection</th>
<th>Shared Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>SPOO</td>
<td>MNCA</td>
</tr>
<tr>
<td></td>
<td>Time [s]</td>
<td>1.312</td>
<td>15.792</td>
</tr>
<tr>
<td></td>
<td>Utilization [%]</td>
<td>43.02</td>
<td>47.94</td>
</tr>
<tr>
<td></td>
<td>Working Util [%]</td>
<td>18.18</td>
<td>21.38</td>
</tr>
<tr>
<td></td>
<td>Throughput [%]</td>
<td>29.67</td>
<td>31.85</td>
</tr>
<tr>
<td></td>
<td>Avail (Ave) [%]</td>
<td>(99619888)</td>
<td>(99619964)</td>
</tr>
<tr>
<td></td>
<td>Avail (Min) [%]</td>
<td>(99619490)</td>
<td>(99619803)</td>
</tr>
<tr>
<td>N35</td>
<td>Time [s]</td>
<td>7.460</td>
<td>212.055</td>
</tr>
<tr>
<td></td>
<td>Utilization [%]</td>
<td>59.49</td>
<td>58.58</td>
</tr>
<tr>
<td></td>
<td>Working Util [%]</td>
<td>25.54</td>
<td>23.46</td>
</tr>
<tr>
<td></td>
<td>Throughput [%]</td>
<td>29.09</td>
<td>29.37</td>
</tr>
<tr>
<td></td>
<td>Avail (Ave) [%]</td>
<td>(99619888)</td>
<td>99619888</td>
</tr>
<tr>
<td></td>
<td>Avail (Min) [%]</td>
<td>(99619001)</td>
<td>99619541</td>
</tr>
</tbody>
</table>

Table 3.4: Numerical results for Protection at Both Layers (in the two rows below physically jointness of the two paths is enabled)
multilayer networks (e.g., SDH over WDM, ATM over WDM) and generalized for more than two layers as well. Applying MNCA with Shared Protection (SP) saves 30-46% capacity in case of Protection at the Lower Layer (PLL), 15-40% in case of Protection at the Upper Layer (PUL), and 57-71% in case of Protection at Both Layers (PBL) or increases the throughput by the same values while the availability is negligibly deteriorated. PLL yields higher throughput than PUL for SP in larger mesh networks, while PUL yields always higher throughput for Dedicated Protection. PBL approaches the optimal availability and needs slightly more capacity if applied with SP. The proposed three phase, iterative heuristic method, Multilayer Availability Enhancement, advances the availability of multilayer to a certain level, while Smart-Mapping and Smart-Routing algorithms increase the throughput of the network significantly.
Chapter 4

Dynamic Routing and Wavelength Assignment in Survivable WDM Networks

Wavelength division multiplexing (WDM) networks are very attractive candidates for next generation optical Internet and intelligent long-haul core networks. In this Chapter we consider WDM networks with wavelength routing switches enabling the dynamic establishment of light-paths between each pair of nodes. The dynamic routing and wavelength assignment (RWA) problem is studied in multifiber networks, assuming both protection strategies: dedicated and shared. We solve the two subproblems of RWA simultaneously, in a combined way using joint methods for the wavelength selection (WS) and wavelength routing (WR) tasks. For the WS problem in contrast to existing strategies we propose a new, network state based selection method, which tries to route the demand on each wavelength, and selects the best one according to different network metrics (such as available channels, wavelengths per fiber and network load). For the WR problem we propose several weight functions for using in routing algorithms (Dijkstra or Suurballe), adapting dynamically to the load of the links and to the length of the path. The combination of different wavelength selection and routing (WS&WR) methods enables wide configuration opportunities of our proposed algorithm allowing good adaptation to any network state. We also propose the extension of the RWA algorithm for Dedicated and Shared Protection and a new method for applying Shared Protection in dynamic WDM environment. The detailed analysis of the strategies demonstrate that our RWA algorithm provides significantly better performance than previous methods in terms of blocking probability whether with or without protection methods.

4.1 Introduction

Nowadays, it is clear that WDM (Wavelength Division Multiplexing) networks will play fundamental and determinant role in the near future telecommunication world. Since WDM can provide an optical transmission system with extremely large data rate (currently 160 wavelengths per fiber is available, each with 2.5 Gbit/s bit rate), it is very attractive for long distance, high transmission speed backbone or core networks. In [70] it is mentioned that all major long-distance carriers in the U.S. have already used a point-to-point WDM transmission technology, and soon
the wavelength routing will be introduced in most of the networks. The ultimate goal is the usage of so-called all-optical network, where a wavelength goes through the network without electronic/optical and optical/electronic conversion providing transparent data transmission.

Similarly to the introduction of the wavelength routed network, the point-to-point networks are replaced by wavelength routed networks, the dynamic establishing of traffic demands also becomes more and more important. Assumption of dynamic traffic arrivals can be explained by the following reasons. First, the continuously increasing traffic volume and the so-called "bandwidth hungry" applications yield that the offered traffic of Internet service providers will be very high. Furthermore, the demand for this kind of service will change from time to time since the demands of the customers are changing as well (the time scale can vary between minutes to hours). Second, it is possible to reconfigure the network in response to changing traffic patterns or link/node failures (in this case the time frame is larger).

Beside the importance of finding an optimal wavelength routing strategy in case of dynamic traffic demands, another critical issue of WDM networks is protection against failures. In fact, each optical channel operates at a rate of several Gbit/s, thus a failure can damage the performance of the network and can affect huge number of customers' demands. Although it is possible to use the recovery procedures of higher layer protocols (e.g., ATM or IP), it is well known that their recovery time is quite long. Since it has many advantages (e.g., speed and efficiency), the application of an optimal protection strategy in the optical layer is proposed.

This Chapter deals with the emerging on-line (on-demand) wavelength selection and wavelength routing (WS & WR) problems in WDM networks with dynamic traffic arrival model, considering different protection strategies. Our objective is to develop an algorithm that is able to solve the dynamic connection establishment problem (routing and wavelength assignment) with or without protection in an optimal way, where the network performance is measured in the blocking probability. In addition to the presentation of our proposed allocation strategies another main objective of the Chapter is to give a detailed analysis of them, applying different network configurations, traffic scenarios and types of protection. Nowadays, this subject is studied extensively in the literature. In the following we give a brief overview of published efforts and results in this area.

The above mentioned problem was first discussed in [71], where the authors assume fixed routing (the routes are given in advance) and a greedy heuristic, called First-Fit algorithm for solving the WS problem. In this case the wavelengths are assumed to be indexed arbitrarily and the new connection is established on the available wavelength that has the smallest index.

In [72] the authors proposed a layered graph model of the WDM network and developed two wavelength selection methods, called PACK and SPREAD, furthermore, they presented a faster version of Dijkstra's method.

In [73] an on-line wavelength assignment algorithm is proposed for multi-fiber WDM networks. They define relative capacity loss for each wavelength and choose the wavelengths with minimum relative capacity loss. For a given number of fibers per link and number of wavelengths per fiber, the algorithm aims to minimize the blocking probability. However, for each potential connection, fixed routing is used.

In [74] the First-Fit algorithm was compared to the Random Wavelength Allocation and simulations show that the blocking probability of the first fit algorithm with fixed routing is considerably lower than the blocking probability with the Random Wavelength Selection algorithm. Furthermore, the authors propose the so-called Least Loaded Routing algorithm for mesh type networks with and without wavelength conversion.

In [75] the idea of packing wavelengths is further extended in the Most-Used algorithm where
the highest loaded and still available wavelength is selected. The authors adopt a general adaptive routing approach, where all paths are considered between a source-destination pair. The authors propose some mechanisms for the wavelength search sequence and evaluate their blocking performance using both single- and multi-fiber networks.

In [76] it is shown that the Most-Used algorithm performs slightly better than the First-Fit one for ring topology. Besides, an algorithm is proposed where the wavelength is selected that maximizes the total network capacity after the connection is established. The authors pointed out that their new algorithm can also be extended to alternate routing using k shortest paths between sources and sinks.

In [77] the authors calculated the approximate blocking probabilities for fixed routing, while in [78] this technique has been extended to alternate routing with random wavelength allocation.

In [79] dynamic routing algorithms based on path and neighborhood link congestion are proposed assuming fixed path.

In [80] the authors propose some sophisticated, link state dependent weight functions for the Dijkstra algorithm and they evaluate the performance of them.

For more details we refer to [1] that contains a detailed overview of the activities on WDM transport network related questions.

On the basis of the literature survey, we have some comments on these papers. At first, one group of the previously presented algorithms use fixed routing, or alternate routing, which means that the available path(s) for each demand are pre-defined, so the main problem is to find an appropriate wavelength for the connection. This yields suboptimal solution since the blocking probability is lower with dynamic route selection. Secondly, the wavelength selection and the routing problem were solved in one phase, using some layered-extension of the network graph as it is presented in [72]. Furthermore, other group of the papers concentrate on finding an adequate path for a demand using a sophisticated version of Dijkstra’s algorithm, but the WS problem is solved in a very simple way, namely using random or simple sorted wavelength selection processes. Protection issues have not been considered in dynamic environment, which is important for less sensitive traffic.

The main focus of the Chapter is to fill these gaps of existing algorithms. We propose different adaptive wavelength selection methods and several sophisticated weight functions for the path selection algorithms (e.g., Dijkstra or Suurballe [12]). It means the selection of both the appropriate wavelength and path based on the current network status. Since the wavelength selection and routing problem is known to be NP-hard [1], heuristic is necessary. For the WS problem we overview our own interpretation of two well-known approaches [74] (called SORT and RAND) and we propose a new approach (called EXHAUST). In case of SORT the wavelengths are examined in a pre-defined order and the first available wavelength is selected. Using RAND, the wavelengths are examined in random order and the first available one is selected. In case of EXHAUST all wavelengths are examined according to the pre-defined network state metric(s) (e.g., load of the current wavelength, path length on the current wavelength using an adequate weight function) and the best one is selected, which provides lower blocking probability in long time frame. We outline the EXHAUST method (see 4.3.4) with different weight functions (see 4.3.2), and compare it with the existing WS algorithms (SORT and RAND). We also examine the effect of 1+1 and shared protection on the WS&WR algorithm and it is analyzed how the network resources are occupied in case of different protection types.

We consider “all-optical” two-connected core or backbone networks with Optical Cross Connects (OXC) and Optical Add-Drop Multiplexers (OADM) in each node. It means that all nodes have switching capability, and any node is able to generate a demand towards any other one.
We assume transmitters and receivers are tunable in all OADMs, which means a demand can be established on any available wavelength.

The nodes are assumed not to have wavelength conversion capability. It is evident that the blocking probability of this kind of network is larger than the network with wavelength conversion, but in [81] it is shown that the difference is very small. If we can pay the small cost of a little bit higher blocking probability, we can save the cost of very expensive wavelength converters. In summary, the cost of the network without wavelength conversion capability is much less than with wavelength conversion, and the performance degradation is absolutely not significant.

Furthermore, the links are assumed multifibers, that means, several fibers are installed between the nodes and each fiber contains the same number of wavelengths.

The rest of the Chapter is organized as follows. In Section 4.2 we present our network model. In Section 4.3 the existing WS methods are overviewed and our proposed WS strategies and weight functions are outlined, as well as the frame of the new WS&WR algorithm. In Section 4.4 the different kinds of protection method are surveyed and a shared protection implementation is proposed. In Section 4.5 the performance evaluation of the proposed strategies is presented.

### 4.2 The Network Model

In a general case we can model any optical network by a directed graph \( G(\mathcal{N}, \mathcal{L}, \Lambda_Q) \), where \( \mathcal{N} \) represents the set of nodes, \( \mathcal{L} \) represents the set of links, and \( \Lambda \) represents the available wavelengths in the network, while \( Q \) denotes the number of different wavelengths. Since we assume multi-fiber network, an integer \( C(l) \) is defined for all links \( l \in \mathcal{L} \), representing the number of fibers installed in this physical connection. Summarizing, our network representation requires unique wavelength number \( Q \) per fiber, but enables different number of fibers on different links. Furthermore, we assume the demands are also directed. The reason of directed physical links is that the link capacity can be different in the opposite directions. On the other hand, the directed demand enables that the magnitude of a demand from \( a \) to \( b \) can be different from the magnitude of a demand from \( b \) to \( a \).

In the algorithm, instead of unique \( G \), a set of graphs \( g_{\lambda(i)}(\mathcal{N}, \mathcal{L}, \lambda) \) is used, where \( \lambda(i) \) is the \( i^{th} \) wavelength, \( i = 1 \ldots Q \). Each \( g_{\lambda(i)} \) graph represents the available network resources on the adequate wavelength \( \lambda(i) \). For each \( g_{\lambda(i)} \) graph, for each link \( l \) a so-called state (column) vector \( \chi(i, l) = (x(i, l, 1), \ldots, x(i, l, z), \ldots, x(i, l, C(l))) \) is defined. If \( x(i, l, z) = 1 \), then fiber \( z \) of link \( l \) in graph \( i \) is occupied, otherwise \( x(i, l, z) = 0 \).

From the viewpoint of dynamic arriving of connection establishments (and tear-downs), we consider a discrete system, i.e., the demand establishing requests or demand tear down requests arrive in arbitrary times \( t_1 < t_2 < \ldots < t_k < t_{k+1} < \ldots < t_K \) (these events are called actions in the rest of the Chapter, we assume only one action at a time). For any time \( t_k \) we may introduce the step - dependent version of state vector \( \chi(i, l) \) using the following notations: \( \chi^k(i, l) = (x^k(i, l, 1), \ldots, x^k(i, l, z), \ldots, x^k(i, l, C(l))) \). This vector shows the load of the adequate \( \lambda(i) \) in step \( k \). We have to note that the condition of the appropriate operation of the algorithm is that \( \min_k(t_{k+1} - t_k) \) must be greater than the required path computation time; we assume that this condition is fulfilled in our network and simulation model.

We introduce some network performance metrics, which will be used in the WS strategies. Let \( N_d(k) \) mean the number of allocated demands in step \( k \). \( N_a^k \) means all of the allocated demands until step \( k \). It is obvious that \( N_a^k = \sum_{j=0}^{k} N_a(j) \). Let \( N_b^k \) the number of blocked demands until step \( k \). Based on the above the average blocking probability until step \( k \) is denoted and
computed in the following way: $p_k^i = \frac{N_i}{N_i^0 + N_i}$. 

Let $\omega_{\lambda(k)}$ denote the number of currently allocated demands on wavelength $\lambda(i)$ in step $k$ over all fibres. Using this $N_i(k) = \sum_{i=0}^{Q} \omega_{\lambda(i)}(k)$. We define the load of a wavelength in step $k$ by $\psi_{\lambda(i)}(k) = \frac{\sum_{i=0}^{Q} \sum_{j=0}^{c(i)} \psi_{\lambda(i)}^j(k)}{\sum_{i=0}^{Q} \psi_{\lambda(i)}^j(k)}$.

4.3 The Proposed Wavelength Selection and Routing Methods

In this section we outline the assumptions for our algorithm and the network environment, we overview the used weight functions of the routing algorithms and the wavelength allocation strategies, finally we give a brief pseudo-code of the proposed WS&WR algorithm.

4.3.1 Assumptions for the Proposed Algorithm and Network Environment

Before outlining our proposed methods, it is important to clarify the assumptions, condition and constraints for the network environment and the WS&WR algorithm.

1. The topology of the network is known and does not change during the working period of the algorithm (except link failures).

2. The source and sink nodes of the incoming demands are known, while the holding time (till the connection will be alive) is not, consequently this information is not usable during the WS&WR phase.

3. There is no information about future incoming requests. It means that the algorithm can only take the actual network status into account.

4. We assume that a demand can reserve one wavelength.

5. The algorithm has information on the current status of the network (e.g., the paths of the established demands). Normally the algorithm operates as a centralized tool, but it is also able to work in the OXC's in a distributed way. In the first case the demand establish or tear down requests are forwarded to the queue of a central management unit. Using this model, the requests are processed in a FIFO mode and it is obvious that the central management tool has recent and precise information about the state of the network. If the WS&WR problem of the current demand is solved, the management tool can handle the required modifications in the adequate nodes and links to establish or tear down the demand. Consequently, our algorithm is suitable to operate in a centralized mode without any restriction.

In case of distributed operation we can realize some difficulties. First of all, the most important problem is the propagation delay of the link state advertisement messages. If $m_{\lambda(k)}(t_{k+1} - t_k)$ is smaller than the required time for all nodes to get information about the latest network state then some nodes have not up to date information about the used network resources, resulting bad wavelength selection and route computation causing demand blocking in the resource allocation phase. We can eliminate the problem if we provide that $m_{\lambda(k)}(t_{k+1} - t_k)$ is larger than the sum of the computation time of the algorithm and the
largest propagation delay. Summarizing, the distributed operation is not impossible, but we have to consider that some demands can be blocked in the establishing phase.

4.3.2 The Proposed Weight Functions

During the wavelength selection (Subsections 4.3.3 and 4.3.4) we have to solve the routing problem of the incoming demands. In our model the WS and WR problems are strongly connected to each other, since the wavelength selection is influenced by the applied weight function of the routing algorithm (see $w_{\text{func}(l)}$ in the pseudo-codes in the next sections). We apply 8 different types of weight functions (from the fixed routing version to the adaptive version, which is able to react to the current status of the network). The first function does not take the link load into account, the central links of the network will be overloaded causing high blocking probability. Therefore we propose 7 new weight functions, trying to share the load among the links. The functions are listed below:

1. $w_{\text{func}(l)} = 1$ for all links. This is the simple fixed, shortest path routing, which is used as measure or reference for advanced weight functions.

2. $w_{\text{func}(l)} = \sum_{z=0}^{C(l)} x(l, 1, z)$ in wavelength $\lambda(l)$. Using this function the demand is routed onto those links, on which the number of used fibers is the smallest.

3. $w_{\text{func}(l)} = \frac{1}{c(l) \cdot \left( \sum_{i=0}^{C(l)} x(l, i, z) \right) + \varepsilon}$ in wavelength $\lambda(l)$; $\varepsilon$ is a small number (e.g., $10^{-4}$) and it is required to avoid the division by zero. In case of this function the weight of a heavy loaded link will be very large (the denominator of the formula is close to zero), therefore these links will not be loaded further if there is at least one less loaded path for the demand.

4. $w_{\text{func}(l)} = \frac{1}{c(l) \cdot \left( \sum_{i=0}^{C(l)} x(l, i, z) \right) + \varepsilon}$. This function is very similar type to the previous one, but tests show that the difference between them is quite significant.

5. $w_{\text{func}(l)} = \frac{1}{c(l) \cdot \sum_{i=0}^{C(l)} x(l, i, z) + \varepsilon}$. This is a little bit simplified function, but in spite of this it seems to be quite effective in case of heavy loaded networks.

6. $w_{\text{func}(l)} = \frac{\sum_{i=0}^{C(l)} x(l, i, z)}{c(l) + 1}$. In this case the least loaded path is selected, independently from its length. Although, using this function we can achieve very good load balance between the wavelength, but the paths often will be significantly longer than the shortest path, causing some non-desirable extra load in the network.

7. $w_{\text{func}(l)} = \frac{\sum_{i=0}^{C(l)} x(l, i, z)}{c(l)}$. This function increases linearly with the load of the link, which may cause that the difference between the load of the wavelengths can be significant.

8. $w_{\text{func}(l)} = \delta + \frac{\sum_{i=0}^{C(l)} x(l, i, z)}{c(l)}$. This weight system considers a correction part ($\delta$), which forces the routing algorithm to select a shorter, more loaded path if the least loaded path is too long.

4.3.3 Existing Wavelength Selection Strategies

In this subsection we describe the two well-known wavelength selection methods SORT and RAND. Both of these methods are based on first-fit wavelength selection strategy, which provides
simple and relative fast operation, but the cost of them is the performance degradation of the methods.

- **SORT** strategy: Using this type of wavelength allocation each wavelength is examined in a predefined order and the first one is selected, where the demand allocation is possible.

```
FOR i=1 to Q DO {
    routable = CheckDemand(a, g\_λ(i), w\_func(t))
    IF (routable == "YES") {
        AllocDemand(a, g\_λ(i), w\_func(t))
        DEMAND ALLOCATION IS POSSIBLE; Exit loop;
    } end IF
} end FOR
IF (routable == "NO") THEN BLOCKING!
```

Function CheckDemand(a, g\_λ(i), w\_func(t)) tries to allocate demand a into wavelength \(\lambda(i)\) with weight function \(w\_func(t)\). If the allocation attempt was successful, the demand is allocated on the current wavelength (indexed by \(i\)) on the shortest path determined by \(w\_func(t)\) (AllocDemand). If all wavelengths have already checked and there is not enough free resource for the allocation then the demand request is blocked. The advantage of this greedy method is its simplicity and fast operation (the demand is allocated on the first suitable wavelength), but its drawback derives from the greedy property (the first fit wavelength does not mean the best fit wavelength). A sophisticated searching and selection between the wavelengths which are able to carry the new demand can result much better long-term network performance and load balancing (using SORT, the wavelengths with smaller index are higher loaded, while others lower).

- **RANDOM** strategy: In this case a wavelength is selected randomly and it is checked whether the demand can be allocated on it or not.

```
FOR j=1 to Q DO {
    i = RANDOM(Q)
    routable = CheckDemand(a, g\_λ(i), w\_func(t))
    IF (routable == "YES") {
        AllocDemand(a, g\_λ(i), w\_func(t))
        DEMAND ALLOCATION IS POSSIBLE; Exit loop;
    } end IF
} end FOR
IF (routable == "NO") THEN BLOCKING!
```

The WS process in this case is very similar to the previous one, the difference is that the sorting of examined wavelengths is different (random) for each demand request. There are \(Q\) attempts to find an available wavelength for the new demand, so each wavelength is examined only once, but the sorting of wavelengths is different for the demands. In long term this method requires about the same computation time as SORT strategy, but because of the randomization, the load balancing between the wavelengths is more uniform than in case of the SORT method.
4.3.4 Our proposed Wavelength Selection Strategies

EXHAUST strategy is the collection of sophisticated wavelength selection methods, which are able to perform better overall network utilization and lower blocking probability than the above widely used processes. Using any version of this method all wavelengths are examined and the best one is selected according to some pre-defined network measure (see below).

\[
\text{best} \_\text{wl} = -1 \\
\text{FOR } i = 1 \text{ to } Q \text{ DO } \\
\text{ routable } = \text{CheckDemand}(a, g_{\lambda(i)}, w_{func}) \\
\text{ IF (routable == “YES”) } \\
\text{ best} \_\text{wl} = \text{Comparison}(i, \text{best} \_\text{wl}, a, w_{func(i)}, WS_{mode}) \\
\text{ END IF } \\
\text{ END FOR } \\
\text{ IF (best} \_\text{wl} != -1) \\
\text{ AllocDemand(a, best} \_\text{wl}, w_{func(i)}) \\
\text{ DEMAND ALLOCATION IS POSSIBLE! } \\
\text{ END IF } \\
\text{ IF (best} \_\text{wl} == -1) \text{ THEN BLOCKING! }
\]

As we can see, the WS process is a bit more complicated than in the previous two cases. All wavelengths (indexed by \(i\)) are examined one by one, and the most suitable one (this is \(\text{best} \_\text{wl}\)) is selected based on \(WS_{mode}\) for demand allocation. The comparison is done using the following performance measures, defined by \(WS_{mode}\) attribute.

- If \(WS_{mode} = 0\), then that wavelength is selected first, where the demand can be realized on the shortest path according to \(w_{func}\). If there are more than one such wavelengths, then in the next phase, from these wavelengths that one is selected, on which \(\omega_{\lambda(i)}\) is the smallest one. This process provides that always the less loaded wavelength is selected among those on which the path is the shortest, consequently the load in the network is kept in equilibrium between the wavelengths.

- If \(WS_{mode} = 1\), the first phase of selection is the same, but in the second phase that wavelength is selected, where \(\omega_{\lambda(i)}\) is the largest one. In this case always the most loaded wavelength is selected among those on which the paths are shortest, resulting maximum utilization of the available wavelengths. Although this strategy does not provide as uniform load distribution between the different wavelengths as the previous one, but it is expected that a new demand can be established in one of the less loaded wavelengths with larger chance.

- If \(WS_{mode} = 2\), the first phase is also the same as in \(WS_{mode} = 0\), but in the second phase that wavelength is selected, where \(\psi_{\lambda(i)}\) is the smallest one. In this case the load of a wavelength is measured better than in the previous cases because in addition to the number of allocated demands \(\omega_{\lambda(i)}\), the length of the paths are also considered in the computation of the load \(\psi_{\lambda(i)}\).

- If \(WS_{mode} = 3\), the first phase is also the same as in \(WS_{mode} = 0\), but in the second one that wavelength is selected, where \(\psi_{\lambda(i)}\) is the largest one.

- If \(WS_{mode} = 4\), then that wavelength is selected, where \(\psi_{\lambda(i)}\) is the smallest (the wavelength is the less loaded) independently from the length of the path of the demand to be

77
established. In this case the least loaded path is selected, but it can cause some additional load increasing, if the path of the demand is too long.

In the further part of the Chapter we will refer to the different settings of $WS_{mode}$ attribute as $EXHAUST(WS_{mode})$; for example $EXHAUST(0)$ means $WS_{mode} = 0$ strategy.

We would like to take some notes on running time of $EXHAUST$ method compared to $SORT$ and $RAND$. If we apply $EXHAUST$, all the wavelengths are examined, which requires $Q$ iterations. In long term, in case of a less loaded network, the average required iterations of $SORT$ and $RAND$ is $\frac{Q}{2}$. Otherwise, if the network load is high the number of required iterations is very close to $Q$ in case of both $SORT$ and $RAND$. It causes that in a network where the average load is near 70-80% there is no significant running time increasing if we use the $EXHAUST$ instead of simple $SORT$ and $RAND$ methods.

4.3.5 Pseudo-Code of the Algorithm Frame

In summary, the user can select from two main types of existing wavelength allocation strategies ($SORT$, $RAND$) and our proposed method ($EXHAUST$). Within the $EXHAUST$, 5 selection strategies may be selected. It means that 7 different strategies may be selected and they are combined with 8 weight functions, resulting 56 different configuration possibilities of the WS&WR algorithm. The following pseudo-code helps to understand the operation of the algorithm:

```plaintext
DEFINE WL_Sel_Strat (1-7)
DEFINE Weight_Type (1-8)

IF(ACTION) CHECK(ACTION)
    IF(ACTION = Demand A Setup Request) {
        FLAG = WS&WR PROCESS(a, WL_Sel_Strat, Weight_Type)
        IF (FLAG = SUCCESS) RESOURCE ALLOCATION(a, Sel_Path, Sel_WL)
        IF (FLAG = FALSE) Demand Request Refused
    }
    IF(ACTION = Demand A Tear Down Request) {
        RESOURCE FREEING(a)
    }

Before the beginning of the operation the user sets the selected WS and WR strategies. The algorithm needs the following information about the network: the nodes, the physical links connected to them and the number of wavelengths (this is graph $G$), as well as the number of fibers per link $(C(l)$ for each link $l$). From these data the algorithm will build the $g_{\lambda(i)}$ graphs.

As we have mentioned, in our context action means a demand establishment request or a demand tear down request. If an action is recognized, the algorithm first checks the type of it. If it is a demand setup request then a Wavelength Selection Process starts (Subsections 4.3.3 and 4.3.4) according to the pre-defined selection strategy and weight function type ($WL_{Sel_{Strat}}$, Weight_Type). If this process was successful (Flag=1), the required modifications are executed in the adequate $g_{\lambda(i)}$ graph of the algorithm and the data of the computed route (Sel_Path, Sel_WL; the selected wavelength and path) are forwarded to the network management to reserve required resources for the demand in the real network. If the computation was unsuccessful (Flag=0) then the demand request is rejected. If the action is demand tear down request then the only task is to release the resources in the adequate $g_{\lambda(i)}$ graph.
Please note that the structure of the algorithm is very flexible and easy to complete with further WS methods or weight functions. It is not a complicated task to apply different kinds of protection/restoration strategy as we will see in the following section.

4.4 Shared Protection in Dynamic WDM Environment

In this section we propose a fast method for applying Shared Protection in dynamic WDM environment. To our best knowledge the behaviour of these methods has not been studied in the framework of dynamic WS&WR.

We have classified the different restoration techniques in Sections 2.1 and 2.2. We have seen that Shared Protection - in contrast to Dedicated Protection - does not allocate bandwidth for each protection path separately but it uses a more sophisticated method to economize on capacity: sharing resources among protection paths. The probability of having two independent failures in the network at the same time is extremely low. For this reason when configuring the system of backup paths, we allocate fewer capacity on each link that still guarantees to survive every single failure.

For simultaneous determination of two diverse paths the method proposed by Sururulate [12] is used because it is able to find a pair of link- or node-disjoint paths with minimal total cost.

Before proposing a fast method for applying Shared Protection in dynamic WDM environment assuming distributed control, we extend the model of Section 2.2.3 for WDM, explained on two illustrative examples. Figure 4.1(a) shows a simple network of one wavelength where three connections are to be set up: two from node a to d and one from node c to f. Each connection has a working (solid line) and a protection (dotted line) lightpath as shown in Figure 4.1(a). We consider two links which can fail: (a,d) and (c,f). All node-pairs are using paths denoted by solid lines as their working paths. If link (a,d) fails then the backup path of (a,d) [a-b-e-d] should overtake the traffic of two lightpaths. But if (c,f) fails then the backup path of it should overtake the traffic of only one lightpath. For this reason (b,e) has to ensure enough capacity for the maximum of the two connections, i.e., two lightpaths for protection purposes. In this case link (b,e) has to have capacity for two lightpaths, since working paths do not use this link. This is enough capacity for full protection under assumption that only one link can fail at a time, therefore two units of capacity are enough instead of three. This method of determining less backup capacity is called capacity compression.

![Figure 4.1: Ideas for shared protection: (a) Capacity compression, (b) Capacity reuse](image)

Figure 4.1(b) shows another case where a working path using links (a,b), (b,c) and (c,f) is depicted. Now consider the case when link (a,b) fails. After the failure the protection lightpath is used and the capacity of the whole working path can be used for protection purposes of other
demands. Consequently, in case of link failure (a,b) the backup capacity of links (b,c) and (c,f) can be extended by the capacity of this demand. This phenomenon is called capacity reuse.

To capitalize the gain of capacity compression and capacity reuse the backup matrix $B^{\lambda(i)}$ for each wavelength $\lambda(i)$ is defined. These matrices are indexed by two links $l$ and $f$, the number indexed by links $l$ and $f$ is denoted by $b_{l,f}^{\lambda(i)}$. This number $(b_{l,f}^{\lambda(i)})$ expresses that at least so many backup channels are to be allocated on link $l$ in order to protect link $f$, i.e., link $l$ provides so many channels "on the account of" link $f$. Obviously, the capacity that is to be allocated for backup purposes on link $l$ is the maximum of $b_{l,f}^{\lambda(i)}$ for each link $f$ ($\max_f b_{l,f}^{\lambda(i)}$).

The values of the backup matrix are the following in every moment. First, due to capacity compression $b_{l,f}^{\lambda(i)}$ is the number of connections with working path using link $f$ and protection path using link $l$ on wavelength $\lambda(i)$. Second, due to capacity reuse we can extract the number of connections with working path using both links $l$ and $f$ on wavelength $\lambda(i)$.

$$b_{l,f}^{\lambda(i)} = \sum_{d \in \text{Active Connections}} \left( (f \in WP_d^{\lambda(i)}) \cdot (l \in PP_d^{\lambda(i)}) - (f \in WP_d^{\lambda(i)}) \cdot (l \in WP_d^{\lambda(i)}) \right) \tag{4.1}$$

Where $f \in WP_d^{\lambda(i)}$ equals to 1 if the $t^k$ wavelength of link $f$ is used by the working path of demand $d$, otherwise it is 0. The meaning of $l \in PP_d^{\lambda(i)}$ analogously indicates the usage of the protection path.

The application of this method to static routing and centralized management is obvious: $Q$ backup matrices $(B^{\lambda(i)}, i = 1...Q)$ are stored in the management system or derived after each protection path allocation. It is introduced in Section 2.2.3 and [C2] without the specialities for WDM. Applying this method guarantees that the network survives any single link or node failure and gives resource savings around 30-40% compared to Dedicated Protection.

Here we propose a very efficient method (in terms of time and memory consumption) to apply Shared Protection in dynamic WDM environment. The difficulty arises that the matrix is very large. If each OXC would store the whole $B^{\lambda(i)}$ matrices, this would cause huge memory consumption in the OXCs and what is more important, causing high administrative traffic in the network since each OXC should be notified about each protection path allocation.

The idea is that each OXC stores only one row of the matrix for each outgoing wavelength. For each link $l$ the starting OXC stores $b_{l,f}^{\lambda(i)}$ for each link $f$ and wavelength $\lambda(i)$, i.e., the number of wavelength channels that are rerouted from $f$ to $l$ if link $f$ fails. Thus, the protection capacity on link $l$, wavelength $\lambda(i)$ can be simply calculated ($\max_f b_{l,f}^{\lambda(i)}$) by the source OXC of link $l$.

The following management operations are to be defined for building up and tearing down working and protection paths:

1. **Build up a working lightpath on wavelength $\lambda(i)$**: When the working path WP is built up, in the starting OXC of each link $l$ do the following:
   (1) Path allocation: Find $z$ such that $x(i, l, z) = 0$ and set $x(i, l, z) = 1$.
   (2) Capacity reuse implementation: For each link $f$ that uses also WP and $f \neq l$: $b_{l,f}^{\lambda(i)} := b_{l,f}^{\lambda(i)} - 1$.

2. **Tear down a working lightpath on wavelength $\lambda(i)$**: When the path WP is torn down, in the starting OXC of each link $l$ do the following:
   (1) Path deallocation: Set the appropriate $x(i, l, z) = 0$.
   (2) Capacity reuse implementation: For each link $f$ that uses also WP and $(f \neq l)$: $b_{l,f}^{\lambda(i)} := b_{l,f}^{\lambda(i)} + 1$.  

80
3. **Build up a protection lightpath on wavelength** $\lambda(i)$: When the path PP is built up, in the starting OXC of each link $l$ do the following:
Path allocation with capacity compression: For each link $f$ that uses WP: $b_{l,f}^{\lambda(i)} := b_{l,f}^{\lambda(0)} + 1$.

4. **Tear down a protection lightpath on wavelength** $\lambda$: When the path is torn down, in the starting OXC of each link $l$ do the following:
Path deallocation with capacity compression: For each link $f$ that uses WP: $b_{l,f}^{\lambda(i)} := b_{l,f}^{\lambda(i)} - 1$.

After every step the number of used fibers on $\lambda(i)$ wavelength on link $l$ can be calculated as follows: $\sum_{i=0}^{C(l)} x(i, l, z) + \max_{l} b_{l,f}^{\lambda(i)}$. In case of a failure the protection path is activated and the above method guarantees that on each link $l$ of the active protection path there is a free channel for the path, i.e., a fiber $z$ can be found such that $x(i, l, z) = 0$.

In summary, for taking the advantage of Shared Protection in a dynamic WDM system the following extensions are needed: (1) Each OXC should store for each outgoing link and each wavelength a vector of $[L-1]$ number ($b_{l,f}^{\lambda(i)}$). (2) At the protection path allocation (build up) and deallocation (tear down) not only the protection path but the working path should be carried as well (since it is needed for steps 3 and 4). (3) Some additional calculations are needed during building up of the protection path. However, for the prize of that minimal cost the gain is to have more available wavelength channels in the network and lower blocking probability (see Section 4.5.3).

### 4.5 Numerical Results

We have conducted numerical investigations in order to show the real improvements resulting from using our algorithms. In Section 4.5.1 the simulation environment and methodology is described while in the following sections the simulation experiments will be introduced. The following areas are surveyed using simulations:

- Our aim was to get insight into the performance of the proposed WS&WR algorithm in cases of different adjustment, traffic load and networks.
- The effect of using different kinds of protection strategy is also investigated deeply.
- Finally, we briefly touch some relevant network configuration/dimensioning questions as well.

#### 4.5.1 Simulation Environment

We used three networks for simulations: the topology of Net_A&T is the same as the national backbone network of AT&T in the USA, Net_P is the national backbone network of Poland, while Net_50 is a random generated network with a dense mesh topology. For simplicity reasons, we assume that the number of fibers per links ($C(l)$) is equal for all links. Table 4.1 contains the most important data of the networks (N-number of nodes; E-number of edges; M-number of demands; C-number of fibers per links; Q-number of wavelengths).

For the description of the demand allocation/deallocation request process we have some mathematical tools in our hand. The most widespread demand arrival model is Poisson-process where the requests arrive with a given $\alpha$ average intensity or the interarrival time between two requests is exponentially distributed with parameter $\tau$. Another possible model is when
The requests arrive at equal time intervals while the number of active intervals is distributed exponentially. Although the above models are suitable for our problem, we applied another demand request arrival model instead of them, which helps to describe and compute the network performance parameters better.

We use a list of actions (referred to as action list), which contains information about the demand request actions and is generated using the following methodology. At first, we have to define the number of actions during the simulation, denoted by \( N_{\text{act}} \). Then the pre-defined average load of the network \( (L_{\text{goal}}) \) to be achieved, and the variance of this value \( (V_{\text{goal}}) \) is to be specified. In our context \( L_{\text{goal}} \) means about how many percent of the demands should be active in the network during a long time frame. (Please consider that in some cases \( L_{\text{goal}} \) can be only a theoretical load because it is possible that the network cannot carry this amount of demands that can result this "set aside" load.) Then an initial process is started, in which as many demand requests are generated that provide load near to \( L_{\text{goal}} \) in the network. Let this current load value denoted by \( L_{\text{curr}} \) (in the simplest case as many demand setup requests are generated which provides that \( L_{\text{curr}} \) is equal to \( L_{\text{goal}} \). After this initial phase the following process is repeated \( N_{\text{act}} \) times:

- If \( |L_{\text{curr}} - L_{\text{goal}}| \leq V_{\text{goal}} \), a demand will be selected randomly using uniform distribution, and if currently it is active then it will be deallocated, while if it is not established then a demand setup request will be generated.

- If \( L_{\text{curr}} - L_{\text{goal}} > V_{\text{goal}} \), then it means that the network seems to become overloaded compared to \( L_{\text{goal}} \) therefore a demand will be deallocated.

- If \( L_{\text{goal}} - L_{\text{curr}} > V_{\text{goal}} \), then it means that the network seems to become underloaded compared to \( L_{\text{goal}} \) therefore a demand setup request will be generated.

This generation method provides that the average network utilization will be between well-definable bounds, and the test pattern will not contain uncontrolled parts where the average load or demand request intensity changes locally. On the other hand, using \( V_{\text{goal}} \) we can set an interval where the demand setup and tear down requests are generated randomly with independent, identical distribution. The aforementioned action list is used as an input file of the algorithm during the different simulation scenarios.

### 4.5.2 Simulations on Blocking Probability

In this section we investigate the performance of our algorithm using different adjustments of wavelength selection strategies and weight functions.

First of all, we would like to prove that sophisticated WS strategies (the different settings of the EXHAUST method) can provide better long-term, overall network performance than that of the widely applied methods, SORT and RAND. In the simulation all of the test networks are used to
examine how the algorithm performs in different network sizes. In our simulations for Net_{AT&T} and Net_T 15000, while for Net_{56} 20000 demand requests are generated to ensure that steady state has been reached. In this test all possible settings of the WS methods are examined, but for the sake of simplicity (to reduce the number of test scenarios) we applied only weight function type 3. Since in this test we do not consider any fault protection we use the simple Dijkstra algorithm for demand routing. In the first test we started from such a demand request arrival intensity which causes that about 50% of the demands are active at a time, then we increase the average number of active demands by 5% test by test until it reached 95%. The results are summarized in Figure 4.2, for Net_{FI}, Net_{AT&T} and Net_{56}, respectively.

![Blocking Probability Curves](image)

**Figure 4.2:** Blocking probability using different WS strategies in Net_{FI}, Net_{AT&T} and Net_{56}

Considering the figures, in all of them two main groups of blocking curves are sharply distinguished and the groups contain the curves of the same wavelength selection strategies in all networks. As it is expected, SORT and RANDOM WS strategies resulted noticeably higher
blocking probability in case of any offered load value than the sophisticated methods. On the other hand, it is very interesting that EXHAUST(4) produces poor quality solution. This is because EXHAUST(4) takes only the residual fiber information into account during the WS process resulting quite long paths on the selected wavelength, and later this long paths cause blocking of other demands. The other four types of EXHAUST strategy result about the same blocking probability curves, using any of them results good network performance.

We can introduce the so-called wavelength selection method gain denoted by $\gamma$, that is a proportion number, which can measure the quality of the wavelength selection methods. It can be show the steady state (it means that the traffic has a fixed intensity and the blocking probability goes to an average value in the long term) difference between the blocking probabilities resulting two different WS strategies. It can be computed in the following way. $\gamma \_2 = 1 - \frac{p_1[WS\_method(1)]}{p_1[WS\_method(2)]}$ (in percent), where in both methods the routing algorithm uses the same weight function. In the same way we introduce a proportion to measure the quality of a weight function, denoted by $\delta$. The computation is very similar to the previous case $\delta \_2 = 1 - \frac{p_1[weight\_func\_type(1)]}{p_1[weight\_func\_type(2)]}$ (in percent) and we assume that the WS methods are the same in case of both weight functions. Thus $\delta \_2$ gives us information how much is the steady state difference between the blocking probabilities using different weight functions. Tables 4.2 and 4.3 contain information about the wavelength selection method gain ($\gamma$) in case of 55% and 75% offered load using NetAFT.

<table>
<thead>
<tr>
<th>Load</th>
<th>$\gamma_{SORT_SORT}$</th>
<th>$\gamma_{RANDOM_SORT}$</th>
<th>$\gamma_{EXHAUST_0_SORT}$</th>
<th>$\gamma_{EXHAUST_1_SORT}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.55</td>
<td>-</td>
<td>-2.64%</td>
<td>27.829%</td>
<td>27.255%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Load</th>
<th>$\gamma_{EXHAUST_2_SORT}$</th>
<th>$\gamma_{EXHAUST_3_SORT}$</th>
<th>$\gamma_{EXHAUST_4_SORT}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.55</td>
<td>27.57%</td>
<td>24.903%</td>
<td>3.82%</td>
</tr>
</tbody>
</table>

Table 4.2: Gain of WS strategies - 55% offered traffic

<table>
<thead>
<tr>
<th>Load</th>
<th>$\gamma_{SORT_SORT}$</th>
<th>$\gamma_{RANDOM_SORT}$</th>
<th>$\gamma_{EXHAUST_0_SORT}$</th>
<th>$\gamma_{EXHAUST_1_SORT}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.75</td>
<td>-</td>
<td>-3.47%</td>
<td>9.505%</td>
<td>7.86%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Load</th>
<th>$\gamma_{EXHAUST_2_SORT}$</th>
<th>$\gamma_{EXHAUST_3_SORT}$</th>
<th>$\gamma_{EXHAUST_4_SORT}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.75</td>
<td>8.91%</td>
<td>8.717%</td>
<td>1.93%</td>
</tr>
</tbody>
</table>

Table 4.3: Gain of WS strategies - 75% offered traffic

On the basis of the $\gamma$ values we can see that in case of 55% offered traffic, EXHAUST(0) strategy produced the best result (it produces 27.829% less blocking probability than the SORT method). EXHAUST(1) and EXHAUST(2) also perform very impressive results while EXHAUST(3) produces a little bit worse values.

In case of larger offered traffic we can realize the same tendency between performance of the WS strategies than in the previous case. Furthermore, EXHAUST(0) produces the best result, but the difference between this and the other three (1, 2, 3) EXHAUST type methods becomes a little bit larger. On the other hand, we can realize that the $\gamma$ values are smaller than in the previous case, which is caused by the increased traffic to be carried by the network.

The effectiveness of the operation in cases of different weight functions is also very important measure of the WS&WR method. Therefore it is tested how the blocking probability varies with different kinds of weight function and our aim was also to determine which is the most
recommended weight function, which can perform a good network operation for both moderate and heavy load. We used the same action lists as in the previous tests, and for wavelength routing strategy we selected EXHAUST(0) since it guarantees the best WS solution. Figure 4.3 shows the resulted blocking probability functions. Tables 4.4 and 4.5 contain information about the wavelength selection method gain (δ), in case of 55% and 75% offered load using Net_{AT&T}.

![Blocking Probability vs Offered Traffic](image1)
![Blocking Probability vs Offered Traffic](image2)

Figure 4.3: Blocking probability using different weight functions in Net_{P}, Net_{AT&T} and Net_{56}

If we consider these three figures together, we can see how the effectiveness of the different weight functions depend on the size of the network and number of the traffic demands. First of all, it can be seen that weight type 6 performs significantly worse blocking probability than the other methods. This is because weight type 6 does not takes any path length information into account. It results that although the least loaded path is selected for the demand to be routed, but this path can be much longer than the shortest path and this causes higher \( P_b \) for other demands, resulting large average blocking probability. We do not expect that weight type
1 produces good results, because this does not take any load value (number of free fibers on the adequate wavelength) into account. In case of $Net_{T'}$ there is no significant difference between the weight functions except of 6, especially when the offered traffic is near to 80-90%. Between 0.5 and 0.65 offered traffic, weight type 3 performs the best blocking probability, while in case of greater traffic intensity weight type 2 is the best, but 7 and 8 also provide a good overall blocking probability. Weight types 4 and 5 do not work optimally in case of a less loaded network, but its performance is increasing with the traffic intensity as can be realized in the 0.75 - 0.95 offered traffic interval. We have to deal with weight type 4 because it produces a not very favorable curve. Considering the formula of weight types 3 and 4 we can realize that the difference between them is almost negligible, but the curves resulted from these formulas significantly differ. In case of $Net_{ATK,T}$ we meet similar situation than in the previous network. Weight type 2 is the best, but weight types 3, 5, 7 and 8 also perform reasonable and stable blocking in case of any offered traffic value. Using $Net_{56}$ almost all curves can be distinguished inside the whole examined offered traffic area, but there is no absolutely optimal weight type. When the offered traffic is between 0.5 and 0.65 weight type 5, when 0.7 weight type 3, in case of 0.75 weight type 5 again and finally between 0.8 and 0.95 weight type 2 produces the best blocking probability values. Similarly, weight types 7 and 8 also perform quite good results. Summarizing, weight types 2, 3, 5, 7 and 8 can guarantee a reasonable blocking probability in case of any network size and traffic intensity. A further, fruitful research direction could be how to combine these weight functions (using different function in case of different traffic intensity and network state) to find the best overall solution.

Table 4.4: Gain of weight functions - 55% offered traffic

<table>
<thead>
<tr>
<th>Load</th>
<th>$\delta_{1,3}$</th>
<th>$\delta_{2,3}$</th>
<th>$\delta_{3,3}$</th>
<th>$\delta_{4,3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.55</td>
<td>-</td>
<td>43.25%</td>
<td>44.40%</td>
<td>13.72%</td>
</tr>
<tr>
<td>0.55</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.5: Gain of weight functions - 75% offered traffic

<table>
<thead>
<tr>
<th>Load</th>
<th>$\delta_{1,3}$</th>
<th>$\delta_{2,3}$</th>
<th>$\delta_{3,3}$</th>
<th>$\delta_{4,3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.75</td>
<td>-</td>
<td>21.53%</td>
<td>15.61%</td>
<td>8.96%</td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We have two notes on the above tables. At first, we can realize how $\delta$ decreases as the offered traffic intensity increases. On the other hand, it seems that in case of 55% offered traffic weight type 5 is the best, but in case of 75% weight type 2 is the winner. It is also important that if we use sophisticated methods, we can achieve 40-50% less blocking probability than using simple shortest path and $\delta$ remains about 10% even in a heavy loaded network.

Beside the blocking probability, it is also important how the network load is changing with the increasing of offered traffic. If the network load remains the same while the blocking probability is increasing, it means that the used WS&WR algorithm is able to handle the demand requests well. On the other hand, decreasing of the network load proves that the WS&WR algorithm
does not work effectively when the requests intensity becomes large. That is why we calculate the network load using $Net_{ATKT}$ in case of different WS processes combined with weight type 3, and different weight types with EXHAUST(0) WS method to examine the performance of our algorithm from this viewpoint, too (Figure 4.4).

![Network utilization using different WS strategies and weight functions in $Net_{P1}$, $Net_{ATKT}$ and $Net_{56}$](image)

The curves on the two figures seem to be very different, however we can find the connection between them. On the first figure we can find the two groups of curves as in Figure 4.2. The sophisticated methods result less network load when the offered traffic are relatively small, but the simpler WS strategies (SORT, RANDOM, EXHAUST(4)) result higher network load even in case of smaller traffic intensity. With the increase of the offered traffic, the curves get closer to each other, finally the utilization begins to decrease caused by an overloaded network situation when the complicated WS strategies can not already produce an optimal load balancing in the network, neither. On the other figure weight types 1, 4 and 6 result low network utilization because of their high blocking probability. Weight types 3, 5 and 7 produce very similar curves as on the other figure while weight types 2 and 8 result a "middle" level utilization. Since the largest value of the utilization using weight type 2 is about 80% it is sure that this type of function provides reasonable blocking probability even in case of larger traffic intensity. Weight types 3, 5 and 7 result utilization close to 90%, consequently they cannot handle significant greater traffic demands to be established.

It is also important to know how the blocking probability is changing with the number of wavelengths or the number of fibers in case of a fixed demand request intensity. We also investigate this question using an average intensity (75 percent of the demand are active in the steady state) and EXHAUST(0) with weight function 3. The results can be seen in Figure 4.5 for $Net_{P1}$ and $Net_{56}$. For each fiber number (1..10) the respective curve shows the blocking probability with each available wavelength number (1..10).
4.5.3 Simulations with Protected Connections

The aim of this section is to overview some relevant questions that appear when we try to solve the WS&WR problem taking different types of protection into account.

At first, we would like to examine how the blocking probability is changing if we apply 1+1 Dedicated and Shared Protection in our networks. During the tests the same network configuration and action lists are used than in Section 4.5.2, the EXHAUST(0) wavelength selection method is applied with weight function 3. The results can be seen in Figure 4.6 for $Net_{AT&T}$ and $Net_{56}$ networks (the results were similar in $Net_{P1}$).

In case of Dedicated Protection at least twice as much resource is needed as in the unprotected case. It corresponds to higher blocking probabilities on these charts. The gain of Shared Protection is significant against Dedicated Protection: it yields about 11-16% less blocking probability than Dedicated Protection. It is a tendency that in larger networks the gain is larger.

Figure 4.5: Blocking probability with different number of fibers and wavelengths in $Net_{P1}$ and $Net_{56}$
Figure 4.6 does not take the effect of a link failure into consideration when the advantages of protected paths appear. Obviously, after a failure the blocking probability is greater than in nominal network state, but the measure of the increase is a very important factor of the effectiveness of WS&WR algorithm.

Another factor is the number of broken connections after failure. This value is zero in case of Dedicated and Shared Protection and will be shown for the unprotected case using restoration. We simulate a single link failure and investigate (1) the number of demands affected by the failure, (2) the number of unrecoverable demands and (3) how the blocking probability is increasing until the link is repaired and we examine the ratio of the increased blocking probability and the blocking probability using protection in the network. In case of 1+1 Dedicated/Shared Protection a single failure does not cause changing of the blocking probability. In this test we use Net_{AT&T} EXHAUST(2), and weight functions 2 and 3, the rest of the configuration data is the same as in the previous tests. We ran 50 different test scenarios to simulate a link failure and calculate the average number of re-routable and blocked demands caused by the failure. Using EXHAUST(2) and weight function 2, five demands were re-routable from the average active 33 affected demands, the rest of the demands were lost (about 85% of those demands were blocked, which used the failed link). In case of EXHAUST(2) and weight function 3 the average number of re-routable demands was 4 and the average number of blocked demands was 28 (it means that 87.5% of those active demands were blocked, which used the failed link).

Another important issue is the ratio of the blocking probabilities in case of different protection types. Using Net_{AT&T} and EXHAUST(2) with weight functions 2 and 3 the following comparison gives a picture about this question. The "normal state" means nominal network operation, "general failure" means that one of the less loaded links is failed while in case of "critical failure" that link is failed what is the most loaded in the long term. We measured how the blocking probability increased after these types of failures (we assumed that the network can achieve a new steady-state during the time of failure). For comparison we computed the blocking probabilities using Dedicated and Shared Protection for the normal network state as well. In the row of general and critical failure we note the blocking probability increasing compared to normal state in parentheses. Table 4.6 summarizes these results.

First of all, we have to note that it is of importance which link fails because the failure of different links causes quite different blocking probability increase. If an often used link fails, the
blocking probability increases by about 15%. Although the blocking probability is significantly higher if a protection method is used even after failure, however we have to note that in case of protection a failure does not cause loss of demands while in this unprotected case (restoration) a failure of a critical link can cause the loss of 15% of demands in long term. It means quality of service degradation and on the other hand significant income decreasing.

We have seen that application of protection methods increases blocking probability. It could be an important information for a network operator how he should extend his network in order to reach same blocking probability with protection as without protection. There are two obvious ways: increase the number of fibers or increase the number of wavelengths, but it is not trivial in which situation which one is better and more economical. Figure 4.7 has the same meaning as Figure 4.5 but these charts are for Dedicated and Shared Protection in Netp. In case of DP the curves are very similar to that of no protection, i.e., the same statements are true for this case, except that they need more fibers and wavelengths for the same blocking probability. Considering Shared Protection there is no gain in case of one fiber compared to DP, and the gain is more and more with the increase of fibers. The reason for this is that more fibers yield larger link capacity in each graph g1(i), which causes more connections in the graph. The consequence of more connections is the more effective use of capacity compression. Thus, using Shared Protection the fiber-extension is more effective than wavelength-extension.

![Figure 4.7: Blocking probability with different number of fibers and wavelengths in case of Dedicated and Shared Protection in Netp](image)

Finally, the question is investigated how to extend an existing WDM network to get the same
blocking probability values as without protection in the original network. All three networks used initially 8 wavelengths (white bars in Figure 4.8 on the left) and number of fibers shown in Table 4.1 (white bars in Figure 4.8 on the right). We have increased the number of used wavelengths one by one and did simulations using Dedicated and Shared Protection. For the new wavelength number the one has been chosen that gave the same or less blocking probability with any traffic load as the original network without protection (Figure 4.8, left chart). After this the wavelength number was fixed and the number of fiber has been extended until the same blocking probability has been achieved as originally (Figure 4.8, right chart).

![Figure 4.8: Number of wavelengths/fibers required for the same blocking probabilities in the three test networks](image)

Using Dedicated Protection the number of wavelengths or fibers should be extended by 120 to 150% of the original one while for Shared Protection it is only 50 to 90% (less in larger networks). The number of new wavelengths with SP is between 54% and 58% compared to DP and the number of new fibers with SP is between 43% and 57% compared to DP.

### 4.6 Summary

This Chapter studied the dynamic Wavelength Selection and Wavelength Routing (WS&WR) problem in multifiber optical networks. A suitable model has been developed for circuit-switched type WDM networks, where the main task was to find an appropriate wavelength (wavelength selection task - WS) and an optimal path (routing task - WR) for the incoming demands. As a part of this task we applied two types of protection methods in the network, namely dedicated and shared, when beside the working path, a protection path is to be found as well. Resource reservation for Dedicated Protection is trivial while a new strategy has been proposed for implementing Shared Protection. Sophisticated strategies for the WS problem have been proposed, which take the current network state into account, while in the routing problem different kinds of link state dependent weight functions have been compared. We have analyzed these methods in different real networks with and without protection, and prepared a detailed performance analysis of them in term of steady-state blocking probability. The tests of our WS&WR algorithm
indicate that the network state dependent WS strategies, as well as the link state dependent weight functions result better network performance (blocking probability) than the simple sort and random WS methods and the classical hop count and/or load based weight functions. For example, EXHAUST(0) WS strategy performs about 10-27% less blocking probability than the often used sorting and random methods. Similarly, the proposed best weight function (2) results 20-40% less blocking probability than the fixed or alternate routing. Furthermore, the gain of using the method with Shared Protection is about 11-16% less blocking probability compared to Dedicated Protection. Summarizing, by applying the proposed algorithms, the performance of optical WDM backbone networks improves significantly.
Chapter 5

Conclusions

Novel algorithms have been developed for configuration and protection of a wide range of infocommunication networks. The algorithms of Chapter 2 are able to configure single layer (e.g., ATM, SDH, MPLS) networks, while in Chapter 3 a Multilayer Network Configuration Algorithm has been worked out. We have proposed a new approach, the “asymmetrically weighted” disjoint pair of paths. These problems are formulated as an Integer Linear Programming task, and several efficient heuristic methods have been given. With methods in Chapter 4 shared protection can be easily applied in dynamic WDM context. The algorithms either yield optimal solution or a tradeoff between running time and quality.

This set of algorithms have been proved to work effectively in practice and are suitable for implementing a network configuration tool, or building them in into existing resource management tools. We can draw a conclusion that in most cases of configuration it is worth using heuristics. For the price of trilling quality degradation they yield results in acceptable time. Their second advantage is that they are able to handle even large scale networks.

The proposed methods assume the principle of virtual paths, while the realisation of them, i.e., whether these paths are realised physically by a packet switched network or a circuit switched network, is beyond the scope of this work. For example, Label Switched Paths of MPLS appear as packet flows on a lower layer while can be treated as virtual paths on a higher layer.

In this Dissertation we have laid emphasis on the reliability of the network infrastructure. In parallel with the protection and increasing availability of the connections, the active time of the corresponding applications increases as well. This is of crucial importance, e.g., for tele-medicine, on-line control systems, on-line banking; while it is more and more significant factor for winning satisfied and pleased “average” users of web, video-on-demand, video conference, etc.

The results of this work could be extended in several directions. In infocommunication networks the available bandwidth typically varies in time. The transmission rate of such elastic traffic can be tuned according to the actual network state. Assuming such elastic traffic there arises the problem how to allocate resource (bandwidth) of sources in a fair manner and how to protect them against failures. In several applications it is also meaningful to define minimum and maximum rate for sources. The research could be continued by developing efficient algorithms for treating elastic traffic.

Other direction is to assign priorities to the traffic demands. This would have the advantage that lower priority traffic could use the backup resources that would be pre-empted by higher priority traffic only in case of a failure. Furthermore, other advanced protection and restoration techniques could be investigated and compared.
As an improvement of Chapter 4 grooming of smaller demands at the switches could be considered as well.
Bibliography


96


[64] M. J. O'Mahony, M. C. Sinclair, B. Milic, “Ultra high capacity optical transmission networks: European research project COST 239”, CONTEL’93, Zagreb, July 1993


98


99
References to own Results

Journal papers


Conference papers


On-line book