Optimal logical topology design with WDM path protection.

Hong Guan
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000 ext. 3208.
Optimal Logical Topology Design with

WDM Path Protection

By

Hong Guan

A Thesis
Submitted to the Faculty of Graduate Studies and Research
Through the School of Computer Science
In Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor
Windsor, Ontario, Canada

June 2003

©2003 Hong Guan
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.
Abstract

Bandwidth usage on the Internet is doubling every six to twelve months according to some research result. The emergence of WDM technology provides more bandwidth and cost effective solution to upgrade the speed of data transmission. The future trend in backbone network design is to gradually migrate to IP over WDM network.

For this next generation optical network, there is one very important issue – design of robust networks that can survive faults. Protection is a common lower layer mechanism against common faults in optical network, such as fiber cuts. There are two protection mechanisms, dedicated path protection or shared path protection, respectively.

This thesis work will concentrate on shared path protection and try to find the optimal solution for logical topology design using integer linear program.

Keywords: WDM, Wavelength Routed Network, Physical Topology, Logical Topology, Single-hop, Multi-hop, Routing and Wavelength Assignment (RWA), Protection, Restoration, MILP, Heuristic.
To my daughter, my parents, and my love...
Acknowledgement

I like to take this opportunity to thank everyone who has offered their wisdom, guidance and support. This thesis work could not have been achieved without their help. A special appreciation to my advisor Dr. Arunita Jaekel and Dr. Subir Bandyopadhyay for their continuous support and valuable advice. Thank Dr. Majid Ahmadi and Dr. Alioune Ngom for their time and suggestions.

Thanks to the entire optical network group for their support and cooperation.
# Table of Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Dedication</td>
<td>iv</td>
</tr>
<tr>
<td>Acknowledgement</td>
<td>v</td>
</tr>
<tr>
<td>List of Figures</td>
<td>viii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
</tbody>
</table>

**Chapter 1: Introduction**

1.1 Physical and Logical Topologies                                 | 1    |
1.2 Faults in WDM networks                                          | 2    |
1.3 Data communication in a WDM network assuming no faults         | 4    |
1.4 Data communication in a WDM network using protection scheme    | 5    |
1.5 The main results of our research                                | 6    |
1.6 Thesis organization                                             | 7    |

**Chapter 2: Review**

2.1 WDM Optical Networks                                            | 8    |
2.1.1 Main WDM devices                                             | 8    |
2.1.2 Wavelength routing                                            | 10   |
2.2 Single-hop and Multi-hop networks                              | 11   |
2.3 Logical Topology Design in networks where the possibility of faults is not considered | 12   |
2.4 MILP approaches                                                 | 16   |
2.5 Heuristic approaches                                            | 17   |
2.6 Survivability and Fault Management                              | 18   |
2.6.1 Restoration Time                                              | 19   |
2.7 Restoration                                                     | 21   |
2.8 Protection                                                      | 21   |
2.8.1 Dedicated Path protection                                     | 23   |
2.8.2 Shared Path Protection                                        | 26   |
2.9 Introduction to MILP and CPLEX                                   | 27   |
2.9.1 Optimization problem                                          | 27   |
2.9.2 LP (Linear Programming)                                       | 27   |
2.9.3 ILP and MILP                                                   | 28   |
Chapter 3: Optimal Logical Topology Design 29

3.1 The MILP formulation 30
  3.1.1 Notation 30
  3.1.2 Binary variables 31
  3.1.3 Continuous variables 35
3.2 Constraints of the MILP formulation 37
  3.2.1 Path creation and channel allocation constraints 38
  3.2.2 Fiber disjoint constraints 40
  3.2.3 Constraints for routing primary lightpaths 41
  3.2.4 Constraints for backup multiplexing 42
  3.2.5 Constraints corresponding to transmitters and receivers 45
  3.2.6 Traffic flow constraints 47
3.3 A Heuristic approach 28

Chapter 4: Analysis and Results 53

4.1 Complexity analysis 53
4.2 Implementation 58
4.3 Experimental results 59
  4.3.2 Solution details 60

Chapter 5: Critical Summary 64

Bibliography 66

Appendix A: Glossary 79

Appendix B: Constraint creation program 80

Vita Auctoris 96
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1 – 1: Physical and logical topology</td>
<td>3</td>
</tr>
<tr>
<td>Figure 2 – 1 Optical Cross-Connect (OXC)</td>
<td>9</td>
</tr>
<tr>
<td>Figure 2 – 2 Multiplexer</td>
<td>9</td>
</tr>
<tr>
<td>Figure 2 – 3 Demultiplexer</td>
<td>9</td>
</tr>
<tr>
<td>Figure 2 – 4 Optical Add/Drop Multiplexer (OADM)</td>
<td>10</td>
</tr>
<tr>
<td>Figure 2 – 5: Physical topology and logical topology</td>
<td>12</td>
</tr>
<tr>
<td>Figure 2 – 6: other possible logical topology</td>
<td>13</td>
</tr>
<tr>
<td>Figure 2 – 7: An 8-Node ($p = 2$, $k = 2$) ShuffleNet</td>
<td>14</td>
</tr>
<tr>
<td>Figure 2 – 8: Physical topology and traffic matrix</td>
<td>14</td>
</tr>
<tr>
<td>Figure 2 – 9: Restoration time according to ITU-T M.495</td>
<td>19</td>
</tr>
<tr>
<td>Figure 2 – 10: Path protection and Link protection</td>
<td>23</td>
</tr>
<tr>
<td>Figure 2 – 11: 1+1 protection</td>
<td>24</td>
</tr>
<tr>
<td>Figure 2 – 12: dedicated path protection (1:1)</td>
<td>25</td>
</tr>
<tr>
<td>Figure 2 – 13: The WDM shared path protection (1:N)</td>
<td>26</td>
</tr>
<tr>
<td>Figure 3 – 1: A physical network with potential logical edges</td>
<td>33</td>
</tr>
<tr>
<td>Figure 3 – 2: (a) RWA for selected lightpaths and (b) final logical topology</td>
<td>34</td>
</tr>
<tr>
<td>Figure 3 – 3: An example illustrating violation of constraint 3.23</td>
<td>44</td>
</tr>
<tr>
<td>Figure 4 – 1: 3-node network with two logical edges</td>
<td>55</td>
</tr>
<tr>
<td>Figure 4 – 2: A 3-node physical topology</td>
<td>61</td>
</tr>
<tr>
<td>Figure 4 – 3: Routing for source destination pair (0, 2)</td>
<td>62</td>
</tr>
<tr>
<td>Figure 4 – 4: Some lightpaths for the 3-node network</td>
<td>63</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 4-1: Comparison of number of variables and constraints</td>
<td>57</td>
</tr>
<tr>
<td>Table 4-2: Some comparative results using our MILP and our heuristic</td>
<td>60</td>
</tr>
<tr>
<td>Table 4-3: Solution details for a 3-node network</td>
<td>61</td>
</tr>
<tr>
<td>Table 4-4: Routing over the logical topology for the 3-node network</td>
<td>62</td>
</tr>
</tbody>
</table>
Chapter 1
Introduction

The rapid increase of bandwidth intensive applications, such as high definition television, video and digital audio, has created an unprecedented demand for bandwidth on the Internet. By some estimates [Tu00], bandwidth usage on the Internet is doubling every six to twelve months, and the growing demand for network bandwidth is expected to continue in the coming years. With recent advances in optical technologies, especially the development of wavelength division multiplexing (WDM) techniques, the amount of raw bandwidth available on fiber optic links has increased by several orders of magnitude. Now a single fiber can have 100 or more wavelengths with possible bit rates of 2.5 to 10 gigabit per second on each wavelength [AQ02]. But this leads to a serious mismatch with current switching technologies, which are not yet capable of switching these high aggregate rates. IP over WDM has been proposed as a possible solution to this problem [Qi00]. IP over WDM is a new layer structure in which IP packets are directly mapped into wavelength channels, without going through intermediate layers such as ATM or SONET/SDH. Such architectures streamlines both network hardware and related software, and at the same time provide a flexible infrastructure. Running IP traffic directly over WDM network is considered to be the next generation optical network [HA00]. Wavelength division multiplexing (WDM) divides the tremendous bandwidth of a fiber into K non-overlapping carrier wavelengths, \( \lambda_1, \lambda_2, \ldots, \lambda_K \) called WDM channels [SRM02]. The value of K is determined by the technology used and a value of K = 200 have been reported [SRM02]. With WDM each physical fiber link can support many optical signals (as many as there are carrier wavelengths on the fiber).

The most frequent cause for failure in WDM networks is a fiber cut due to a break in any fiber in the network. In this thesis we have described novel and improved ways to design robust WDM networks which can withstand the failure due to a fiber cut.
1.1 Physical and Logical Topologies

The physical topology of an optical network consists of a set of end-nodes, router nodes and the optical fibers interconnecting these nodes. Each end-node in the network is capable of generating data for transmission, receiving data and is typically a computer having a number of optical transmitters and receivers. A router node is an optical device connected to a number of incoming fibers and a number of outgoing fibers. Optical signals on any incoming fiber may be communicated to any outgoing fiber depending on the way the router is programmed. The physical topology of an optical network may be conveniently represented by a digraph $G$ where

- the nodes in the graph $G$ are the end-nodes and router nodes in the physical network
- if the node pair $(i, j)$ are connected by a fiber in the physical network, in the graph $G$ there is a directed edge(also called link) from $i$ to $j$.

A lightpath in an optical network is a point-to-point communication path that connects a transmitter at a source node to a receiver at a destination node [SRM02] where no opto-electronic conversion is needed at any intermediate node. Such a path where there is no opto-electronic conversion is called an all-optical path. If we look at the graph model of the physical topology of the network given in Figure 1-1, a lightpath starts from an end-node (the source of the communication), traverses a number of links (optical fibers on the physical network) and intermediate nodes (optical routers) and ends at an end-node (the destination for the communication). In the simplest and the most widely used case which we also consider in this thesis, the lightpath uses the same carrier wavelength $\lambda_i$ from the source node to the destination node on all the fiber links from the source node through the intermediate nodes to the destination node. This is also called wavelength continuity constraint [RM99].

After the lightpaths have been routed on the physical topology, there are typically many lightpath that share the same physical link. If two lightpaths share the same fiber, then they must be assigned two different carrier wavelengths on that fiber [ABJ02].
If there is a lightpath from node $P$ to node $Q$, we can communicate data from $P$ to $Q$ by using a transmitter at $P$ which modulates some carrier wavelength $\lambda_i$ for some $i$, $1 \leq i \leq K$, using the data to be sent and injects it into the fiber connecting the source node to the first intermediate node in the path from $P$ to $Q$. This will take the signal at wavelength $\lambda_i$ to the first intermediate node, which is a router. This router is already preprogrammed to direct the incoming signal at wavelength $\lambda_i$ to the appropriate fiber that connects the first router node to the next router node in the path to the destination node. This process continues until the signal reaches the destination node. Such communication involving only one lightpath is also called single-hop.

![Diagram](image)

(a) Physical topology  
(b) Logical topology

**Figure 1–1: Physical and logical topology**

As shown in figure 1–1 (b), the communication from node1 to node3 involves single-hop even though in the physical topology (figure 1–1 (a)), there is no direct fiber link between node1 and node3.

To ensure that it is possible to have communication using single-hop only, a network must have lightpaths between all source-destination pairs. In an N node network this means that we need at least n (n-1) lightpaths and is generally not practical. In real-life networks, only some pairs of end-nodes are connected by a lightpath. If two end-nodes $X$
and Y are not connected by a lightpath, we need to determine a sequence of nodes $A_1, A_2, \ldots, A_k$ such that there is a lightpath from source node X to node $A_1$, a lightpath from node $A_1$ to node $A_2$, ..., a lightpath from node $A_k$ to the destination node Y. The communication from X to Y can be accomplished in a number of hops. The first hop is to send the data from source node X to node $A_1$. The next hop is to send the data from node $A_1$ to node $A_2$. This process continues until the final hop where the data is sent from node $A_k$ to the destination node Y. Such networks are often called multi-hop networks. Our objective is to design multi-hop networks.

In this connection, the notion of a logical topology represented by a graph $LG$ is useful where

- the nodes in the graph $LG$ are the end-nodes in the physical network
- if the node pair (i, j) are connected by a lightpath, in the graph $LG$ there is a directed edge from i to j

In terms of the graph $LG$, to find a way for a source node X to communicate with a destination node Y we have to find a path $X \to A_1 \to A_2 \to \ldots \to A_k \to Y$. Henceforth we will refer to an edge in the logical topology as a logical edge. For example, as shown in figure 1-1 (b), the edge (1→3) is a logical edge.

When developing a scheme to accommodate all requests for communication in an optical network, we have to decide which end-nodes need to be connected by a lightpath. In other words, we have to design the logical topology for the given physical topology.

### 1.2 Faults in WDM networks

The failure of a fiber link, as mentioned earlier, is the most frequent cause of network failure and may cause all lightpaths using the faulty fiber to fail. The net result is significant data loss. In order to have a reliable and survivable WDM network, when designing logical topology, we not only need to accommodate the traffic specified by the user, but also should be able to handle failures, e.g. have strategies for fault management in place. Since single fiber failure is the predominant form of failure in optical networks,
a lot of research has been concentrated in this area. Our focus is also on single fiber failures.

There are two main approaches for fault management: protection and restoration. In restoration, the spare capacity available after the fault's occurrence is utilized for rerouting the disrupted traffic. In protection, spare capacity is reserved during call setup. For every primary lightpath, a backup lightpath is created and kept in reserve at the time of designing the logical topology. In the absence of failures communication uses primary lightpath alone. When there is a failure due to a fiber cut, a number of primary lightpaths are affected since they use the failed fiber. Communication that uses these affected lightpaths is resumed using the corresponding backup lightpaths. Detailed discussions of restoration and protection techniques are given in chapter 2.

1.3 Data communication in a WDM network assuming no faults

Given the user requirements for data communication, our objective is to determine a strategy for data communication so that the amount of traffic on the most heavily loaded transmitter or receiver is as little as possible. In terms of the logical topology, this means minimizing the traffic on the logical edge carrying the maximum traffic. The traffic on the logical edge carrying the maximum traffic is often called the network congestion. Given the user requirements for data communication and a description of the physical topology, we have to solve the following subproblems:

i) Determine an optimum logical topology \( L_G \) for the network

ii) To have a valid lightpath corresponding to each logical edge in \( L_G \), determine an appropriate wavelength and a route through the physical topology. This is called the Route and wavelength assignment problem (RWA).

iii) For each request for data communication, say from source node \( i \) to destination node \( j \), find an optimum way to route the data from \( i \) to \( j \) using a multi-hop path in the logical topology so that the congestion is minimized.

These subproblems are interrelated since our objective is to minimize the congestion and subproblem 3 cannot be solved until subproblem 1 is solved. Subproblem 1 in turn
requires subproblem 2 to be solved in order to ensure that the logical topology is indeed realizable.

1.4 Data communication in a WDM network using protection scheme

Our objective in this case is the same as in the previous section - to determine a strategy for data communication so that we minimize the network congestion. The only difference is that we have a failure management scheme, which will take care of single fiber cuts in the network. Given the user requirements for data communication and a description of the physical topology, the subproblems we have to solve are similar to those we described in the previous section and are as follows:

i. Determine an optimum logical topology $LG$ for the network

ii. Corresponding to each logical edge in $LG$, determine an appropriate wavelength and a route through the physical topology for a primary lightpath and a backup lightpath. The fibers used in the primary lightpath must be distinct from the fibers used in the backup lightpath.

iii. For each request for data communication, say from source node $i$ to destination node $j$, find an optimum way to route the data from $i$ to $j$ using a multi-hop path in the logical topology so that the congestion is minimized.

These subproblems are again interrelated for the same reasons mentioned in the previous section.

1.5 The main results of our research

Our research is focused on the static logical topology design using shared path protection mechanism. We have made two major contributions to the state of the art in this subject. We have developed the logical topology design problem as a Mixed Integer Linear Programming (MILP) formulation. Our formulation is interesting because it minimizes the number of integer variables without resorting to any simplification of the problem. We have tested our formulation using CPLEX and have compared the complexity of our formulation to Mukherjee [SRM02], which uses a major problem simplification.
Our formulation is still too complicated for use in practical sized networks. However, it is still important, because

i) this will give us the absolute optimized solution;
ii) it can be used for small networks;
iii) the solution can be used as benchmark to evaluate other heuristics.

We have split the subproblems mentioned in section 1.4 into two parts. Part I deals with the logical topology design problem including the RWA for the primary and backup path. Part II deals with the problem of efficient routing in a predetermined logical topology in order to find a routing with minimum congestion. To solve part I we have developed very fast heuristic and have tested the heuristic with a number of networks both existing and synthetic.

The entire work was done jointly by this author and Ms Min Hou. In this thesis, we have emphasized the first approach even though in chapter 3, we have outlined both our approaches.

1.6 Thesis organization

The rest of the thesis is organized as follows:

Chapter 2 will give thesis related background review and a brief introduction to MILP (Mixed Integer Linear Programming).

Chapter 3 will give an overview of both the approaches mentioned in section 1.5. It also includes detailed explanations of our constraint formulae and some critical analysis of our approach.

Chapter 4 will include an analysis of our MILP formulation and present our experimental results for several small physical network topologies.

Chapter 5 will conclude the thesis with some critical summery and suggestions for future work.
Chapter 2
Review

2.1 WDM Optical Networks

Wavelength division multiplexing (WDM) divides the tremendous bandwidth of a fiber (potentially a few tens of terabits per second) into many non-overlapping carrier wavelengths (WDM channels) [RM99]. Transmissions on different wavelengths are coupled into a single fiber using wavelength multiplexers [SRM02]. The number of wavelengths that can be carried by a given fiber depends on fiber characteristics and other technological constraints such as the sensitivity of optical components, effect of cross talk on fibers. With the rapid advances in WDM technology, the transmission capacity over a single fiber has increased dramatically [MB98]. In addition, the numerous improvements made on optical components such as optical transmitters, receivers, filters, amplifiers, and add/drop multiplexers have now made it possible to have up to 200 wavelengths on a single fiber, each supporting a data rate of 2.5 to 10 gigabit per second [AQ00].

2.1.1 Main WDM devices

In this section, we briefly outline the main optical components used in WDM systems.

Optical cross-connect (OXC): This device provides the ability to route or switch individual optical signals on any of its input ports to any output port, without requiring the signal to undergo any optical-electronic conversion. If an OXC is equipped with wavelength converters, then it can also change the wavelength of an incoming optical signal as it passes through the switch [LES00] (figure 2–1). Using OXCs at intermediate nodes allows an all-optical communication channel (lightpath) to be established between two end-nodes.
Multiplexer: This device combines several optical signals at different carrier wavelengths so that they can be transmitted over the same fiber (figure 2-2).

Demultiplexer: This device splits several optical signals at different carrier wavelengths on one fiber into signals on a number of fibers where each fiber carries only one signal at distinct wavelength [RS98]. See figure 2-3.
**Optical add/drop multiplexer (OADM):** This device allows one or more optical signals having some specified carrier wavelengths to be selectively added or dropped at a node, and allows the remaining optical signals to pass through the node unaffected. [TS00] As shown in figure 2–4, through the OADM, $\lambda_3$ is added and $\lambda_4$ is dropped, $\lambda_1$ and $\lambda_8$ are optically passed through.

![Diagram of OADM](image)

Figure 2–4: Optical Add/Drop Multiplexer (OADM)

### 2.1.2 Wavelength routing

Wavelength routed networks are optical networks where the routing and switching of optical signals are carried out based on their wavelengths. According to [LES00], WDM networking has been launched by the concept of wavelength routing. The principle is that each high-speed data flow is associated with a specific optical carrier wavelength directly routed through the network, based on its wavelength, from the end-node which is the source of the data flow to the end-node which is the destination of the data flow, without being converted to the electronic domain at the intermediate nodes which are all router nodes. This is made possible by the development of all-optical routers, which are able to handle many WDM channels simultaneously.

An important feature of wavelength routing optical network is transparency of the signal format. Each lightpath can carry different types of data flows simultaneously at a variety of bit rates. Furthermore, the wavelengths can be spatially reused in the network. This means that the same wavelength can be allocated to two or more separate lightpaths, as
long as these lightpaths do not share a common fiber. Therefore, although the number of wavelengths available may be limited, the number of lightpaths that the network can support is typically much higher. Finally, the network can be configured in such a way that when failure occurs, the lightpaths can be rerouted over alternative paths automatically. This provides a high degree of reliability in the network. Considerable work has been done in this area, in recent years [RM99a][RM99b][SRM02][HSSY00][HYC00][Ma00][MSC00][CB98].

2.2 Single hop and multi-hop networks

An optical network topology can be single-hop or multihop (a hop is defined as a logical connection between two nodes without any opto-electronic processing at intermediate devices, if any). In a single-hop topology, optical signals carry information from a source node to the destination node without undergoing any opto-electronic conversion [RM99a][RM99b]. Note that the information from the source to the destination node may be routed via multiple optical devices (e.g., passive star coupler, all-optical multiplexer, all-optical router), however, no electronic processing takes place in these devices. In a multiple-hop topology, traffic from a source node to a destination node might travel via some intermediate nodes where it undergoes opto-electronic conversions, gets processed, and is sent out via another link, possibly on a different wavelength [RM99b][LDS98].

For example, in figure 2–5, Node 1 can reach Node 2 in one hop. Thus, whatever Node 1 transmits on wavelength $\lambda_1$ reaches Node 2 at the speed of light (in fiber). On the other hand, Node 1 can reach Node 3 in two hops. After the first hop, the signal from Node 1 undergoes opto-electronic conversion at Node 2. Then, based on the destination address in the message supplied by Node 1, Node 2 decides that the message has to be forwarded to Node 3. Node 2 re-encodes the message in the optical domain and forwards it to Node 3 on wavelength $\lambda_1$. Similarly, Node 2 can reach Node 5 via three hops.
A WDM network with N end-nodes where all communication uses single-hop requires n (n-1) lightpaths. Since this is not feasible for nontrivial networks most WDM networks use multi-hop.

2.3 Logical Topology Design in networks where the possibility of faults is not considered

On a given physical topology, different logical topologies can be embedded by employing multiple parallel WDM channels and by properly configuring the network elements. In a wavelength-routed network, different switching configurations of the wavelength routers will result in different logical topologies. For example, for the same physical topology in figure 2–5 (a), if we create lightpaths as shown in figure 2–6 (a), we will get a logical topology in figure 2–6 (b), in which each node has a lightpath to every other node.

Note that, if the network topology is relatively static, then fixed wavelength transceivers can be used instead of tunable ones. Also, the wavelength-routed networks, can exploit the advantages of wavelength reuse. [BJS99].
The above discussion shows that a large number of logical topologies may be mapped to the same physical topology. A very important question is the mechanism through which a good logical topology may be selected. It is also important to decide exactly what parameter we want to optimize when designing such networks. Some pertinent parameters are the diameter of the logical topology and the amount of electronic processing that have to be done for traffic grooming [AQ00] at an end-node. The proposed logical topologies can be broadly classified into two categories: arbitrary and regular. The nodal connectivity patterns in regular topologies are very systematic and well defined, which simplifies routing and management operations. Another important advantage is that the diameter of such logical topologies is known to be very small. Over the past few years, several regular logical topologies have been proposed for optical networks [CGK92][Gi90][MA98][BSSB95][Wu92]. However, it is difficult to add an arbitrary number of nodes to a regular topology and still maintain its well-defined structure. Another major disadvantage is that such a structure does not take into account known static traffic pattern.

Examples of regular multihop network topologies include ShuffleNet [Gi90][MA98], 2-Dimensional Torus (Manhattan Street Network (MSN)), de Bruijn graph [BSSB95],
Hypercube [Wu92] and TreeNet [Wu95]. Comprehensive surveys on regular and multihop network architectures can be found in [RM99b][LDS98][IMG96][GRS97]. Figure 2–7 shows one of these topologies.

![Diagram of a network topology](image)

**Figure 2–7: An 8-Node \( p = 2, k = 2 \) ShuffleNet.**

In multihop networks, which do not have any predetermined topology, there is no discernable connectivity pattern. Based on the traffic between all node pairs, the design objective is to develop a logical topology that handles the traffic in a most efficient manner.

![Diagram showing physical topology and traffic matrix](image)

<table>
<thead>
<tr>
<th></th>
<th>(1)</th>
<th>(2)</th>
<th>(3)</th>
<th>(4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>0.00</td>
<td>0.20</td>
<td>0.80</td>
<td>0.40</td>
</tr>
<tr>
<td>(2)</td>
<td>0.30</td>
<td>0.00</td>
<td>0.10</td>
<td>0.70</td>
</tr>
<tr>
<td>(3)</td>
<td>0.90</td>
<td>0.40</td>
<td>0.00</td>
<td>0.10</td>
</tr>
<tr>
<td>(4)</td>
<td>0.60</td>
<td>0.20</td>
<td>0.40</td>
<td>0.00</td>
</tr>
</tbody>
</table>

**Figure 2–8: Physical topology and traffic matrix**
One convenient way to represent the traffic between different node pairs in an n-node network is to use an \( n \times n \) matrix. Such a matrix is called a traffic matrix. The entries in the traffic matrix are typically obtained by observing long-term traffic patterns over the network. A network with four nodes is shown in figure 2–8 (a), and a possible traffic matrix for this network is shown in figure 2–8 (b). The entry \( t_{ij} \) represents the amount of traffic from node \( i \) to node \( j \). It is convenient to normalize the traffic between any node pair and represent it as a fraction of the capacity of a lightpath. For example, the amount of traffic from node 1 to node 3 is given as 0.80 in the traffic matrix shown in figure 2–8 (b). If a lightpath in the network can carry 2.5 Gb/sec, this means that the actual traffic is 2.0Gb/sec. This entry \( t_{ij} \) represents the long-term average (or peak) traffic between the node pair \((i, j)\).

From the traffic matrix \( A \) in figure 2–8 (b), we can see that there is a lot of traffic between node 1 and node 3 \( (t_{13} = 0.80 \text{ and } t_{31} = 0.90) \), but as shown in figure 2–8 (a), there is no lightpath from node 1 to node 3, so the traffic \( t_{13} \) reaches node 3 in two hops. It has to undergo opto-electronic conversion at Node 2. Besides, from node 1 to node 2, it uses lightpath \( p_{12} \) and from node 2 to node 3 it uses lightpath \( p_{23} \), the traffic on these two paths will be \( p_{12}: t_{12} + t_{13} = 1.00 \) and \( p_{12}: t_{23} + t_{13} = 0.90 \), which makes these two lightpaths very congested while other lightpaths such as \( p_{41} \) only carry less than half of the lightpath capacity. For the traffic \( t_{31} \), the two possible route are \((3 \rightarrow 4 \rightarrow 1)\) and \((3 \rightarrow 4 \rightarrow 2 \rightarrow 1)\), but none of them can carry \( t_{31} \), which is 0.90, because although lightpath \( p_{34} \) have 0.90 spare capacity, but lightpath \( p_{41} \) only has 0.40 spare capacity, and for lightpath \( p_{42} \) and \( p_{21} \), no one has enough spare capacity to carry \( t_{31} \). Therefore, the above logical design is not very good for this traffic matrix.

The end-nodes have to convert the optical signal to electronic signal and vice-versa and a network with a lower value of congestion means less conversion at the end-nodes, which translate to less complexity of the electronic circuits for the conversion. For a given physical network topology minimizing the congestion level, based on the traffic matrix, is a good metric for designing logical topology.
For a WDM optical network, if we could establish a lightpath between every source destination pair we immediately get a network having a minimum value of congestion. In such a solution there is no opto-electronic conversion within the network except at the end nodes. If such a solutions is feasible for a network, it alleviates the electronic bottleneck and is called an all-optical network For an all-optical network having \( n \) nodes, the fully connected logical topology needs \( n(n-1) \) lightpaths, and the number of wavelengths necessary to support the \( n(n-1) \) lightpaths can be extremely large if \( n \) is large. So all-optical logical topology is generally not feasible.

In order to best exploit the capacity of the WDM network infrastructure, a crucial task is the identification of the best feasible logical topology. For a given physical network, the best logical topology is determined by the pattern of traffic flows over the network. Conventional design methods of the logical topology have been focusing on maximizing the total amount of traffic on the network or minimizing the congestion on the network [BJS99]. For many years this problem was extensively studied by many researchers and many approaches were proposed [MBRM96] [RS96] [SL95] [LMM00][DR00][GL01].

These approaches can be classified into two types: MILP (Mixed Integer Linear Programming) approaches and heuristic approaches.

### 2.4 MILP approaches

Many researchers have considered the logical topology design problem as an optimization problem. They formulate a set of efficient mixed integer linear programming, and use Integer linear programming tools to solve it. For a limited number of transmitters and receivers at each node, connected by optical fiber links that support a limited number of wavelengths, and given a traffic matrix whose elements correspond to the (average) traffic exchanged by all source destination nodes pair, some optimization targets used by different researchers are:

— Minimize the average number of hops [RS96][LMM00].
— Minimize the maximum congestion level in the network [SRM02]
— Minimize the average packet delay at the nodes [Mu97] [RS96].

The number of hops, the maximum congestion level and the average packet delay are all interrelated. For instance, minimizing the average number of hops also leads to reducing the average traffic flow on the links, and hence reduces the congestion level. This will indirectly minimize the queuing delay. Directly or indirectly, most logical topology design aim to minimize the maximum congestion level in the network [Ag02].

Using the MILP approach, some researchers combine the logical topology design problem and routing problem together [RS96]. Since the combined problem is computationallly hard to solve, especially for large network, they split it into two subproblems: the logical topology design problem and the routing problem, and solve those two problems independently [KAM00]. The logical topology design problem is formulated as MILP problem, and the routing problem is formulated as LP problem.

2.5 Heuristic approaches

The MILP solutions are computationallly intractable for practical networks, as they generate a large number of constraints and integer variables. Many heuristic approaches have been proposed to address this problem. Generally, there are four types of heuristic approaches [Ag02]:

1. Heuristic solutions of the MILP problem — Genetic algorithms, simulated annealing, variable depth local search techniques, LP-relaxation, etc. [RS98] [LLP00] [SPM99] [Ka00] [LJB02]. The drawback, quoting [Ag02], is that "the solution quality is not satisfying".

2. Maximizations of the single-hop traffic flows — Single-hop maximization logical topology design algorithm (SMLTDA). But if in/out degree of nodes is small and the traffic matrix is small, it may result in non-connected logical topology. Greedy logical
topology design algorithm (GLTDA), it uses multiple parallel lightpaths and attempts to introduce logical links between nodes that exchange large amounts of traffic [RS96].

3. Heuristic maximizations of the single-hop and multi-hop traffic flows – there are two types:

- AL (Adding lightpaths) – Increasing multi-hop logical topology design algorithm (I-MLTDA). It starts from a non-connected topology and sequentially adds lightpaths, trying at each step to minimize congestion level in the network.

- RL (Remove lightpaths) – Decreasing multi-hop logical topology design algorithm (D-MLTDA). It starts from a fully connected topology and sequentially removes lightpaths, selecting those that carry the smallest traffic flows [DR00].

4. Algorithms based on the adoption of a pre-established regular logical topology – Pre-established logical topology design algorithm (PLTDA) and Random logical topology design algorithm (RLTDA) [Du01].

### 2.6 Survivability and Fault Management

Today’s WDM networks are capable of providing very high bandwidths, with bit rates of 2.5 to 10 Gigabit per second on each carrier wavelength and 100 or more wavelengths on each fiber [AQ00]. Therefore, if the network is down, due to a fault in the network, even for a few seconds, it typically leads to huge data losses. So a highly available network, which is very resilient to network failures, is a very important requirement for WDM networks. The term *survivability* refers to a network’s ability to continue providing services in the presence of failures [MVWY99]. Survivability can be addressed within many different layers in the network hierarchy, with each layer implementing its own protection mechanisms.

In this thesis, we are focusing on IP over WDM networks. In such networks, the protection and restoration mechanisms may be implemented within either optical (WDM)
layer or the IP layer [SRM02]. At the WDM layer, for every primary lightpath, a backup path and a corresponding wavelength is reserved in advance at the time of lightpath setup. At the IP layer, over-provisioning of network capacity may be provided so that after a failure, alternate routes through the network can be dynamically discovered, by utilizing the spare capacity [VCDJ00][Wu95][Ra02][La02].

2.6.1 Restoration Time

Restoration speed i.e. how quickly a network can be recovered from a failure is a key criteria when developing a reliable and survivable optical network. The ITU-T (International Telecommunications Union on Telecommunications) specifies how the restoration time is calculated [TS02]. Fig 2-9.

![Diagram of Restoration Time]

Figure 2–9: Restoration time according to ITU-T M.495.
In the case of a failure, it takes some time for the node next to the faulty component in the network to detect that failure. This time interval is called Detection Time (T1).

After some time, the failure is confirmed, and the restoration procedure is initiated. The time interval between this point of time and the failure recognition is called Waiting Time (T2).

During the restoration, the control signals are transmitted and the received signals are processed. This time interval is called Restoration Procedure Time (T3).

The amount of time required for processing the last received control signal is called Restoration Transfer Time (T4).

In the last step, a verification of the protection switching operation or some resynchronization might be completed. This time interval is called Recovery Time (T5).

For all of these five time intervals, the time between the failure occurrence and the fault confirmation is also called Confirmation Time (Tc). The time between the fault confirmation and until the last restoration operation is finished is also called Transfer Time (Tt). The overall time interval from the failure occurrence to the full restoration is done is called Restoration Time (Tr).

The telecommunication industry has traditionally required a recovery time in the order of 50 ms. [VCDJ00]

Restoration techniques at the IP layer usually result in recovery times, which are significantly larger (of the order of seconds). While the recovery time using optical layer protection techniques are of the order of milliseconds, which can largely minimize data losses. Besides, the optical layer can efficiently multiplex protection resources (such as spare wavelengths and fibers) among several higher layer network applications and also the survivability at the optical layer provides protection to higher layer protocols that may not have built-in protection [RM99]. Generally, restoration schemes are more efficient in
utilizing capacity due to the sharing of the spare capacities, while protection schemes have a faster recovery time and provide guarantees on the recovery.

Since single fiber failures are the predominant form of failures in optical networks, a lot of research has been done, on designing optical networks that provide fault-management techniques to combat such fiber failures [AQ00][ABJ02][SRM02][GR00][GRS97]. In the following section, we will briefly highlight some of these techniques.

2.7 Restoration

Restoration can be used to provide either more efficient routes after the protection is completed, or additional resilience against further faults before the first fault is fixed [GR00]. Usually, restoration mechanism is quite slow (seconds to minutes) and can be computed by a centralized management system [GR00]. Under the current technology, a restoration of a ring based optical network takes about 50 ms [GR00][Me00] and that of an end-to-end mesh based optical network needs a few hundreds ms.

Restoration provides a second step of protection against network failures. Restoration can handle both link failures and node or multiple concurrent failures. It's typically applied in mesh topologies. In dynamic restoration, the spare capacity available within the network is utilized to restore the services that are affected by a failure. Restoration can be implemented in a centralized [GR00] or distributed [SS00] fashion. Since our work did not use this technique, we will not review this technique in detail.

2.8 Protection

The physical layer is close to most of the common failures on the network, such as a fiber cut. Survivability mechanisms in the optical layer involve detecting this and performing a simple switch to divert the traffic through an alternate path. This is called protection [EL96].
Protection as the lower layer mechanism provides a first step of defense against common faults in optical network. Protection is topology and technology-specific and offers fast recovery, but protection may be unable to protect against node failure or multiple faults. Usually, a fixed amount of capacity is dedicated for protection purposes in order to make a fast transfer of traffic from failed facilities to good ones. Those resources may be dedicated for each failure scenario, or may be shared among different failure scenarios [RM99]. In other words, depending on how the dedicated pre-assigned capacity is used, there are two protection mechanisms, dedicated or shared protection, respectively. Also, depending on how to provide the protection, there are two types of protection approaches, path protection [WMM00] [AQ00] [RM99] and link protection [El96] [RM99], respectively.

In path protection, a backup path, which is also an end-to-end restoration path, is reserved during a call setup. In case of a failure, the traffic restoration is handled by the source and the destination nodes along the backup path between the nodes.

In link protection (also called line protection), backup paths are reserved around each link of the primary path. The traffic restoration is handled by the nodes adjacent to the failure, e.g. the end-nodes of the failed link dynamically discover a backup path around the link, all the connections that traversing the failed link are rerouted to the backup path around that link.

For example, as shown in figure 2–10, for primary path (1→2→3→4), when failure occurs at (2→3), in the path protection scheme, the traffic will switch to backup path (1→6→5→4), figure 2-10 (a), and in the link protection scheme, the traffic will be routed to (1→2→6→3→4), figure 2-10 (b), here link (2→6→3) is the backup link of (2→3).

In the following sections we will discuss path protection in more detail since our approach is based on path protection.
2.8.1 Dedicated Path Protection

In dedicated path protection, 50% of the entire capacity in the network is reserved for protection purposes. It is obvious that dedicated protection delivers the highest level of protection but leads to inefficient network utilization. A typical example is the UPSR (Unidirectional Path Switched Ring) architecture used in SONET/SDH ring networks [LCC01][RS98].

Dedicated path protection can be of two types:

1+1 Protection

When using this type of protection, the traffic to be protected is sent along two parallel paths simultaneously (usually over disjoint routes) from the source to the destination. During normal operation, the destination node receives two identical traffic streams and just simply selects one of them by monitoring the signal quality or bit error rate. When failure occurs on the selected path, the destination simply switches onto the other path. When the original working path is repaired, no action is needed since the repaired path now becomes protection path. This kind of protection does not need protection signaling
since the destination node can handle a failure by itself, the source node does not need to do any thing, but always send the traffic onto the alternate path whenever it send traffic to the destination. This makes 1+1 protection very easy to implement, and its restoration time is very short. But its disadvantage is obvious, it wastes valuable bandwidth. Fig 2-11 shows 1+1 protection.

![Diagram of 1+1 protection](image)

Figure 2–11: 1+1 protection.

**1:1 Protection**

This type of protection still uses two parallel paths from the source to the destination, but the difference is that during normal operation, traffic is transmitted over only one path (primary path) at a time, it does not send traffic onto alternate path (backup path). When failure occurs along the primary path, both source and destination nodes switch onto the backup path. In unidirectional transmission systems, because the traffic is sent in only one direction over a fiber, the source node cannot realize a fiber failure by itself, thus the destination node has to inform the source to switch over to the protection backup path, and therefore, it requires signaling protocol called automatic protection switching (APS). When using bi-directional transmission systems (where traffic is transmitted in both directions over the fiber), both ends can detect the failure, so no signaling is required. The advantage of this type of protection is, during normal operation, the unused protection path can be used for transmitting low-priority traffic. This achieves better utilization of network resources than 1+1 protection [LCC01]. When failure occurs, the high-priority
traffic is switched over to the protection path, and low-priority traffic is dropped. Note that if there is no extra transmitter or receiver available to handle the low-priority traffic, this advantage disappears. When the original working path (primary path) is repaired, it is desirable to have the traffic switched back onto the primary path so that new failures can be handled and the low-priority traffic may continue to use the protection path. The disadvantage is that for unidirectional communication system, it requires signaling and adds communication overhead, so it has longer restoration time than 1+1 protection. Fig 2–12 shows the 1:1 protection.

![Diagram](image)

Figure 2–12: dedicated path protection (1:1)

In figure 2–12, the primary lightpath $p_1$ ($4 \rightarrow 5 \rightarrow 6$) uses wavelength $\lambda_1$, and its corresponding backup path $b_1$ ($4 \rightarrow 8 \rightarrow 0 \rightarrow 6$) uses wavelength $\lambda_2$; another primary lightpath $p_2$ ($5 \rightarrow 10 \rightarrow 9$) also uses wavelength $\lambda_1$. This is possible since $p_1$ and $p_2$ do not share common fiber, and the corresponding backup path $b_2$ ($5 \rightarrow 4 \rightarrow 8 \rightarrow 7 \rightarrow 9$) uses wavelength $\lambda_3$. Backup path $b_2$ cannot use wavelength $\lambda_2$ because it shares edge ($4 \rightarrow 8$) with $b_1$, and the wavelength $\lambda_2$ of edge ($4 \rightarrow 8$) is being used by $b_2$.

As illustrated in figure 2–12, in dedicated path protection, the resources used by one backup path cannot be shared by other backup paths.
2.8.2 Shared Path Protection

In shared path protection, at the time of setting up for a primary lightpath, an edge disjoint backup lightpath is also reserved. However, the backup wavelength reserved on the links of the backup path may be shared with other backup paths. It is important to ensure that backup paths that share resources will never need to use those resources at the same time. For single fault assumption, this can be achieved by corresponding primary paths being edge disjoint.

Figure 2–13 is an example that illustrates WDM layer shared path protection. In fig 2 - 13, the primary lightpaths are $p_1 (4\rightarrow 5\rightarrow 6)$ and $p_2 (5\rightarrow 10\rightarrow 9)$, both using wavelength $\lambda_1$. The corresponding backup paths are $b_1 (4\rightarrow 8\rightarrow 0\rightarrow 6)$ and $b_2 (5\rightarrow 4\rightarrow 8\rightarrow 7\rightarrow 9)$, they both use wavelength $\lambda_2$. Note that the two primary lightpaths are fiber disjoint, so their corresponding backup lightpaths can share wavelength $\lambda_2$ on fiber $(4\rightarrow 8)$. If the fiber $(4\rightarrow 5)$ fails, all the traffic on primary lightpath $(4\rightarrow 5\rightarrow 6)$ on wavelength $\lambda_1$ is routed over its backup lightpath $(4\rightarrow 8\rightarrow 0\rightarrow 6)$ on wavelength $\lambda_2$.

Shared path protection has more efficient utilization of network resources (bandwidth) than dedicated path protection. Our research is focused on the shared path protection scheme.

![Diagram of WDM shared path protection](image)

Figure 2–13: The WDM shared path protection (1:N)
2.9 Introduction to MILP and CPLEX

As mentioned in chapter 1, we consider the logical topology design problem as an optimization problem and one of our approaches is to formulate MILP problem and use CPLEX to solve it. In the following section, we give a brief overview of MILP techniques and CPLEX.

2.9.1 Optimization problem

Optimization problems are defined by three basic components:

- An objective function—which we want to minimize or maximize. For instance, in a manufacturing process, we might want to maximize the profit or minimize the cost.
- A set of variables—which affect the value of the objective function. In the manufacturing problem, the variables might include the amounts of different resources used or the time spent on each activity.
- A set of constraints—that allow the variables to take on certain values but exclude others. For the manufacturing problem, it does not make sense to spend a negative amount of time on any activity, so we constrain all the "time" variables to be non-negative.

The optimization problem is then:

Find values of the variables that minimize or maximize the objective function while satisfying the constraints.

2.9.2 LP (Linear Programming)

According to [CRC99], the definition for linear programming is:
A problem expressible in the following form—
minimize \[ \sum_{j=1}^{n} c_j x_j \]

subject to \[ \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad i = 1, \ldots, m \]

variables: \( x_j \)

problem data: \( c_j, a_{ij}, b_i \)

The basic problem of linear programming is to minimize a linear objective function of continuous real variables, subject to linear constraints.

### 2.9.3 ILP and MILP

ILP (Integer linear programming) — A linear program with additional constraints that all of the variables must take on integer values.

MILP (Mixed integer linear program) — A linear program with additional constraints that some of the variables must take on integer values.

Compared to ILP, MILP has less complexity. Because for LP problem, the complexity is partially determined by the number of integer variables. As in MILP, some of the variables are non-integer, the total number of integer variables are less than in ILP.

CPLEX is a software tool for solving linear optimization problems, which is commonly referred to as Linear Programming (LP) problems. More information about CPLEX is available in [IL01].
Chapter 3
Optimal Logical Topology Design

In this chapter, we will describe our two main approaches for logical topology design. In each case, the goal is to determine a topology that

- accommodates the entire traffic flow
- provides protection against any single fiber failure
- does not exceed resource limitations, such as the number of transmitters, the number of receivers and the number of available wavelengths on each fiber.

In order to achieve this goal we need to

i) determine the optimum logical topology
ii) determine the optimal routing and wavelength assignment for the lightpaths corresponding to each logical edge and
iii) determine how the traffic should be routed over the logical topology.

In the first approach, we formulate the logical topology design problem as a MILP optimization problem. Given an optical fiber network, the associated resource constraints, and a traffic matrix, the goal of the MILP is to produce a logical topology with the minimum possible congestion. This is the most general formulation of the problem and solving this MILP leads to the optimum logical topology.

Due to the complexity of the logical topology design problem, the complete MILP formulation can be solved only for very small networks. Therefore, we have also developed a heuristic to efficiently solve this problem. The heuristic is able to produce a good (but not necessarily the best) solution within a reasonable amount of time. In this chapter, we will give an overview of our heuristic. Details of the implementation and experimental results on the heuristic are reported in [MH03].
3.1 The MILP Formulation

We are given the physical topology of a fiber network $G[N, E]$ with $|N| = n$ nodes and $|E| = m$ edges. Each edge $(i, j) \in E$, can accommodate a set $K$ of channels, with $|K| = K$. We are also given a traffic matrix $\Lambda$, the number of transmitters and receivers at each node, and a set $P$ (with cardinality $P$) of potential logical edges. If we allow the possibility that there may be at most one logical edge between any two nodes in the network, then there are $n(n-1)$ potential logical edges. If we also consider that there may be multiple logical edges between a given node pair, then there will be $r n(n-1)$ potential logical edges, where $r$ is the maximum number of logical edges allowed between any two nodes. For any logical edge $p \in P$, $o(p)$ represents the originating node and $l(p)$ represents the terminating node for the corresponding lightpath.

Given the above information, we can solve our MILP formulation to obtain

i) a set of logical edges selected from the complete set of potential logical edges $P$.

ii) the primary and backup lightpath for each selected logical edge

iii) the traffic flow on each logical edge

3.1.1 Notations

In this section, we define the notation used in our MILP formulation.

- $G[N, E]$ represents a physical fiber network with a set of nodes $N$ and a set of edges $E$. $|N| = n$ is the number of nodes in the network and $|E| = m$ is the number of edges in the network.
- $N$ represents a set of physical nodes, with $|N| = n$.
- $E$ represents a set of edges (bi-directional), with $|E| = m$.
- $K$ represents a set of wavelengths that each edge $(i, j) \in E$ can accommodate.
- $P$ represents a set of potential lightpaths.
- $o(p)$ represents a source node of lightpath $p$.
- $l(p)$ represents a destination node of lightpath $p$.
- $\Lambda = (\lambda^{sd})$ is a $n \times n$ static traffic matrix, where $\lambda^{sd}$ represents the average traffic exchanged between a source $s$ and destination $d$. 

30
• \((r_u^1, r_u^2, \ldots, r_u^k)\) is the given receiver vector, where \(r_u^k\) provides the number of receivers at node \(u\) tuned to channel \(k\).

• \((t_u^1, t_u^2, \ldots, t_u^k)\) is the transmitter vector, where \(t_u^k\) provides the number of transmitters at node \(u\) tuned to channel \(k\).

• \(\varepsilon = \min\{x^{sd}: x^{sd} > 0, s,d \in \mathbb{N}\}\) is the minimum non-zero value in the traffic matrix.

### 3.1.2 Binary Variables

In this section, we define the binary (0-1 integer) variables that are used in our formulation. We have used three main types of binary variables in our formulation. They are:

i) logical edge selection variables \(b_p\)

ii) route creation variables for lightpaths \(x_p, y_p\) and

iii) channel assignment variables for lightpaths \(w_{kp}, z_{kp}\).

The descriptions of these variables are given below.

\[
b_p = \begin{cases} 
1, \text{if logical edge } p \text{ is selected.} \\
0, \text{otherwise.} 
\end{cases} \quad \forall p \in \mathbf{P}
\]

There is a binary variable \(b_p\) associated with each potential logical edge in the set \(\mathbf{P}\). The value of the variable \(b_p\) determines whether a particular logical edge \(p\) is to be included in the final logical topology. For a network with \(n\) nodes, we assume that there may be up to \(r\) logical edges between any node pair. Hence there are \(rn(n-1)\) such variables. In our experiments, we have set \(r = 1\). After solving the MILP, if the value assigned to \(b_p\) is 1, this means that the \(p^{th}\) logical edge will be included in the logical topology.
\[ x_{ij}^p = \begin{cases} 
1, & \text{if primary lightpath for } p^{th} \text{ logical edge uses edge } (i, j). \\
0, & \text{otherwise.} 
\end{cases} \]

\[ y_{ij}^p = \begin{cases} 
1, & \text{if backup lightpath for } p^{th} \text{ logical edge uses edge } (i, j). \\
0, & \text{otherwise.} 
\end{cases} \]

The binary variables \( x_{ij}^p \) (\( y_{ij}^p \)) determine the route, over the physical topology, for the primary (backup) lightpath corresponding to the \( p^{th} \) logical edge. In the remainder of the thesis, we will refer to the primary (backup) lightpath associated with the \( p^{th} \) logical edge as the \( p^{th} \) primary (backup) lightpath. For a particular primary (backup) lightpath, the variables \( x_{ij}^p \) (\( y_{ij}^p \)) can have a value of 1 only if corresponding logical edge is selected to be in the final logical topology (i.e. \( b_p = 1 \)). In this case, \( x_{ij}^p = 1 \) (\( y_{ij}^p = 1 \)), if the \( p^{th} \) primary (backup) lightpath uses the edge \( (i, j) \) in the physical topology. If the \( p^{th} \) logical edge is not selected (i.e. \( b_p = 0 \)), then \( x_{ij}^p = 0 \) and \( y_{ij}^p = 0 \) for all \( (i, j) \in E \). There are \( 2mP \) route creation binary variables in our MILP formulation.

\[ w_{kp} = \begin{cases} 
1, & \text{if the } p^{th} \text{ primary lightpath is assigned channel } k. \\
0, & \text{otherwise.} 
\end{cases} \]

\[ z_{kp} = \begin{cases} 
1, & \text{if the } p^{th} \text{ backup lightpath is assigned channel } k. \\
0, & \text{otherwise.} 
\end{cases} \]

As mentioned in chapter 1, each lightpath in a logical topology has a physical route and a channel associated with it. The binary variables \( w_{kp} \) (\( z_{kp} \)) are used to determine the channel assigned to the \( p^{th} \) primary (backup) lightpath. If the \( p^{th} \) logical edge appears in
the logical topology then \( w_{kp} = 1(z_{kp} = 1) \) implies that the \( p^{th} \) primary (backup) lightpath is assigned channel \( k \). If the \( p^{th} \) logical edge does not appear in the logical topology, then there is no need to establish the associated lightpaths. In this case \( w_{kp} = 0 \) and \( z_{kp} = 0 \), for all possible values of \( k \). For each primary (backup) lightpath, there are \( K \) channel assignment variables associated with it - one variable for each possible value of \( k \). There are \( P \) primary and \( P \) backup lightpaths to be considered. Therefore, our MILP formulation has \( 2KP \) channel assignment variables. Figures 3-1 and 3-2 illustrate the significance of the binary variables and show how they are used to determine the logical topology and establish the lightpaths.

![Physical topology](image)

**Figure 3-1:** A physical network with potential logical edges

Figure 3.1 shows a physical network with a set of four potential logical edges \( L_0, L_1, L_2, L_3 \), and two wavelengths \( \lambda_0 \) and \( \lambda_1 \) per fiber. Suppose logical edges 1 and 3 are selected to be in the final logical topology and logical edges 0 and 2 are not. In other words, \( b_0 = 0, b_1 = 1, b_2 = 0, \) and \( b_3 = 1 \). Since we do not need to establish lightpaths corresponding to logical edges 0 and 2, the associated binary variables are all set to 0. Therefore we have:
i) For logical edge 0:

\[ b_0 = 0, \]
\[ x_{ij}^0 = 0, \quad y_{ij}^0 = 0 \quad \forall \ (i,j) \]
\[ w_{k0} = 0, \text{ and } z_{k0} = 0 \text{, for all } k \]

ii) For logical edge 2:

\[ b_2 = 0, \]
\[ x_{ij}^2 = 0, \quad y_{ij}^2 = 0 \quad \forall \ (i,j) \]
\[ w_{k2} = 0, \text{ and } z_{k2} = 0 \text{, for all } k \]

For logical edges 1 and 3, the physical routes and channel assignments for the primary and backup lightpaths are shown in Figure 3-2(a).

![Figure 3-2: (a) RWA for selected lightpaths and (b) final logical topology](image)

For logical edge 1, the primary lightpath P₁ is assigned channel \( \lambda_{i1} \), and uses edges (2,1) and (1,4). The corresponding backup lightpath B₁ is assigned channel \( \lambda_{o0} \), and uses edges (2,3) and (3,4). For logical edge 3, the primary lightpath P₃ is assigned channel \( \lambda_{o0} \), and uses edges (4,1). The corresponding backup lightpath B₃ is assigned channel \( \lambda_{o0} \), and uses edges (4,3) and (3,1). Therefore we have:
iii) For logical edge 1:

\( b_1 = 1, \)
\( x_{21}^1 = 1, \ x_{14}^1 = 1, \ \text{and} \ x_{i}^1 = 0 \ \text{for all remaining values of} \ (i, j). \)
\( y_{23}^1 = 1, \ y_{34}^1 = 1, \ \text{and} \ y_{i}^1 = 0 \ \text{for all remaining values of} \ (i, j). \)
\( w_{01} = 0, w_{11} = 1. \)
\( z_{01} = 1, z_{11} = 0. \)

iv) For logical edge 3:

\( b_3 = 1, \)
\( x_{41}^3 = 1, \ \text{and} \ x_{i}^3 = 0 \ \text{or all remaining values of} \ (i, j). \)
\( y_{43}^3 = 1, \ y_{31}^3 = 1, \ \text{and} \ y_{i}^3 = 0 \ \text{for all remaining values of} \ (i, j). \)
\( w_{03} = 1, w_{13} = 0. \)
\( z_{03} = 1, z_{13} = 0. \)

The result logical topology is shown in figure 3–2 (b).

### 3.1.3 Continuous variables

In this section we briefly describe some of the continuous variables used in our MILP formulation. The complexity of the formulation is critically dependent on the number of integer variables. Therefore, we have tried to minimize the number of integer variables and tried to formulate the constraints in terms of the continuous variables whenever possible. Even though we have continuous variables for computational advantages, the values allowed by our restriction are 0 or 1.

\[
\delta_{k}^{ip} = \begin{cases} 
1, & \text{if primary path } p \text{ is assigned channel } k \text{ and uses edge } (i, j). \\
0, & \text{otherwise.}
\end{cases}
\]
This variable is used for the routing and wavelength assignment of primary lightpaths. For a given fiber (physical edge \((i, j)\)) and a channel \(k\) on that fiber, the variable \(\delta_{ij}^{kp}\) indicates whether the \(p^{th}\) primary lightpath is using that channel. There are \(m\) edges in the physical network with \(K\) channels on each edge and \(P\) primary lightpaths to consider. Therefore there are \(mKP\) continuous variables for all possible values of \(k, p\) and \(ij\) in \(\delta_{ij}^{kp}\).

\[
\alpha_{pq}^k = \begin{cases} 
1, & \text{if } p^{th} \text{ primary lightpath and } q^{th} \text{ backup lightpath both use channel } k. \\
0, & \text{otherwise.}
\end{cases}
\]

This variable is used in constraints 3.10 – 3.13 to ensure that a primary lightpath is channel disjoint with respect to all backup lightpaths. \(\alpha_{pq}^k = 1\) implies that the \(p^{th}\) primary lightpath and \(q^{th}\) backup lightpath both use channel \(k\). We have to consider \(P\) primary lightpaths and for each primary lightpath we must consider all \(P\) backup paths. Therefore, we have \(P^2\) pairs of lightpaths. For each such pair, there are \(K\) channels that may be used. Hence, there are \(KP^2\) continuous variables for all possible values of \(k, p\) and \(q\) in \(\alpha_{pq}^k\).

\[
\gamma_{pq}^k = \begin{cases} 
1, & \text{if } p^{th} \text{ backup lightpath and } q^{th} \text{ backup lightpath both use channel } k. \\
0, & \text{otherwise.}
\end{cases}
\]

\[
\xi_{ij} = \begin{cases} 
1, & \text{if } p^{th} \text{ backup lightpath and } q^{th} \text{ backup lightpath share edge } (i, j). \\
0, & \text{otherwise.}
\end{cases}
\]

\[
\theta_{pq} = \begin{cases} 
1, & \text{if } p^{th} \text{ backup lightpath and } q^{th} \text{ backup lightpath share at least one edge.} \\
0, & \text{otherwise.}
\end{cases}
\]
The variables $\gamma^k_{pq}$, $\xi^p_{ij}$, and $\theta_{pq}$ are used in constraints 3.14 to 3.23, which enforce the rules for backup multiplexing. For all these variables, we are considering two backup lightpaths, and the order in which the lightpaths are chosen is not important. In other words, $\gamma^k_{pq} = \gamma^k_{qp}$, $\xi^p_{ij} = \xi^q_{ij}$, and $\theta_{pq} = \theta_{qp}$. Therefore, we only need to consider $P(P-1)/2$ combinations of lightpaths in each case. Hence, there are $KP(P-1)/2$ variables of type $\gamma^k_{pq}$, $mP(P-1)/2$ variables of type $\xi^p_{ij}$, and $P(P-1)/2$ variables of type $\theta_{pq}$.

$$
\eta^p_{ij} = \begin{cases} 
1, & \text{if } p^{\text{th}} \text{ primary lightpath uses edge (i, j) and corresponding backup lightpath uses channel } k. \\
0, & \text{otherwise.}
\end{cases}
$$

This variable is used in constraints 3.26 to 3.30 which determine the number of transmitters and receivers being used at each node. There are $mKP$ such variables.

$f_p^{sd}$: is the amount of $s$-$d$ traffic (traffic originating from node $s$ and destined for node $d$) that is routed over the $p^{\text{th}}$ logical edge.

$F_{\text{max}}$ is the maximum amount of traffic flow on any lightpath.

The above variables are used for routing the traffic requests between different node pairs over the logical topology.

### 3.2 Constraints of the MILP formulation

In this section we will describe the different physical constraints that must be satisfied for the successful routing and wavelength assignment of selected lightpaths. We will also discuss the constraints for routing traffic over the logical topology. Finally, we will derive the mathematical equations that represent these constraints and give detailed explanations of the equations.

The MILP formulation consists of two parts:
i) the objective function and
ii) the constraints

The objective function in our formulation is:

\[ \text{Minimize } F_{\text{max}} \]

Here \( F_{\text{max}} \) is the congestion of the network, which is defined as the maximum amount of traffic flow on any lightpath. We recall that we discussed congestion earlier in chapter 1. In our formulation, we have normalized the traffic flow on any lightpath with respect to the maximum amount of data that can be transmitted over a single lightpath (2.5 - 10 Gb/s depending on technology). Therefore, the total amount of traffic flow on any lightpath can never exceed 1. Hence, \( 0 \leq F_{\text{max}} \leq 1 \). The value of the congestion of a network determines the factor by which the traffic matrix can grow and still be accommodated by the given logical topology. Therefore, minimizing congestion is desirable because it allows the maximum possible growth in the traffic matrix. For example if the congestion of a network is 0.5, then every entry in the traffic matrix can be doubled and the network would still be able to accommodate the entire traffic without allocation of any additional resources.

### 3.2.1 Path creation and channel allocation constraints

\[
\sum_{j: (i,j) \in E} x_{ij}^p - \sum_{j: (j,i) \in E} x_{ji}^p = \begin{cases} 
  b_p, & \text{if } i = o(p) \\
  -b_p, & \text{if } i = l(p) \\
  0, & \text{otherwise}
\end{cases} \quad \forall i \in N, \forall p \in P \tag{3.1}
\]

\[
\sum_{j: (i,j) \in E} y_{ij}^p - \sum_{j: (j,i) \in E} y_{ji}^p = \begin{cases} 
  b_p, & \text{if } i = o(p) \\
  -b_p, & \text{if } i = l(p) \\
  0, & \text{otherwise}
\end{cases} \quad \forall i \in N, \forall p \in P \tag{3.2}
\]

Constraints 3.1 (3.2) employ the standard technique used in network flow programing to find a route, over the physical topology, for the \( p^{th} \) primary (backup) lightpath, from
its source \( s = o(p) \) to its destination \( d = l(p) \). We will explain constraint 3.1 in detail for a primary lightpath. Constraint 3.2 works in exactly the same manner for a backup lightpath.

For each lightpath \( p \), constraint 3.1 must be satisfied at each node \( i \) in the network. If the \( p^{th} \) logical edge is not selected (i.e. \( b_p = 0 \)), then \( x^p_{ij} = 0 \) and \( y^p_{ij} = 0 \) for all \( (i, j) \in E \). In this case constraint 3.1 is trivially satisfied at all nodes.

If \( b_p = 1 \), we have to consider three cases.

**Case 1 \((i = s)\):**

\[
\sum_{(i,j) \in E} x^p_{ij} - \sum_{(j,i) \in E} x^p_{ji} = 1, \quad \text{if } i = s
\]

The above equation states that there is one outgoing edge \((s, j) \in E\) such that \( x^p_{sj} = 1 \) and this is the first edge in the physical route (from \( s \) to \( d \)) for the \( p^{th} \) primary lightpath. None of the incoming edges to node \( i=s \) are on the physical route associated with the \( p^{th} \) primary lightpath.

**Case 2 \((i = d)\):**

\[
\sum_{(i,j) \in E} x^p_{ij} - \sum_{(j,i) \in E} x^p_{ji} = -1, \quad \text{if } i = d
\]

The above equation states that there is one incoming edge \((j, d) \in E\) such that \( x^p_{jd} = 1 \) and this is the last edge in the physical route (from \( s \) to \( d \)) for the \( p^{th} \) primary lightpath. None of the outgoing edges from node \( i=d \) are on the physical route associated with the \( p^{th} \) primary lightpath.

**Case 3 \((i \neq s, i \neq d)\):**
\[ \sum_{j : (i,j) \in E} x_{ij}^p - \sum_{j : (j,i) \in E} x_{ji}^p = 0, \quad \forall \ i \in \mathbb{N}, \ i \neq s, \ i \neq d \]

This equation holds for all other nodes in the network. In this case, if \( i \) is an intermediate node in the path from \( s \) to \( d \), there is exactly one incoming edge to node \( i \) and one outgoing edge from node \( i \) which are on the physical route associated with the \( p^{th} \) primary lightpath. Otherwise, none of the incoming or outgoing edges of node \( i \) are on the physical route of the \( p^{th} \) primary lightpath.

Constraint (3.2) describes the same situation but for backup lightpaths.

\[ \sum_{k \in K} w_{kp} - b_p = 0, \quad \forall \ p \in P \tag{3.3} \]

\[ \sum_{k \in K} z_{kp} - b_p = 0, \quad \forall \ p \in P \tag{3.4} \]

Constraints 3.3 and 3.4 enforce the wavelength continuity constraint for primary and backup lightpaths respectively. If the \( p^{th} \) logical edge is selected \( (b_p = 1) \), then \( \sum_{k \in K} w_{kp} = 1 \) and \( \sum_{k \in K} z_{kp} = 1 \). Since the channel assignment variables are binary variables, this means that \( w_{kp} (z_{kp}) = 1 \), for exactly one value of \( k \) and \( w_{kp} (z_{kp}) = 0 \) for all other values of \( k \), for the \( p^{th} \) primary (backup) lightpath. This ensures that a lightpath (primary or backup) is assigned exactly one channel.

### 3.2.2 Fiber disjoint constraints

\[ x_{ij}^p + y_{ij}^p - b_p \leq 0, \quad \forall (i,j) \in E, \quad \forall \ p \in P \tag{3.5} \]

Constraint 3.5 states that if edge \((i,j)\) is used by the \( p^{th} \) primary lightpath then it cannot be used by the \( p^{th} \) backup lightpath and visa-versa. In other words, the \( p^{th} \) primary lightpath and \( p^{th} \) backup lightpath must be fiber disjoint. This constraint is necessary because, if a primary lightpath and its backup lightpath have a common edge, and a failure occurs on
that edge, both the lightpaths will be affected. In this case, it will not be possible to recover from the failure.

### 3.2.3 Constraints for routing primary lightpaths

\[
x_{ij}^p + w_{kp} - \delta_{ij}^{kp} \leq 1, \quad \forall (i, j) \in E, \quad \forall k \in K, \quad \forall p \in P
\]  
\[
\delta_{ij}^{kp} - x_{ij}^p \leq 0, \quad \forall (i, j) \in E, \quad \forall k \in K, \quad \forall p \in P
\]  
\[
\delta_{ij}^{kp} - w_{kp} \leq 0, \quad \forall (i, j) \in E, \quad \forall k \in K, \quad \forall p \in P
\]  
\[
\sum_{p \in P} \delta_{ij}^{kp} \leq 1, \quad \forall (i, j) \in E, \quad \forall k \in K
\]  

Constraints 3.6 – 3.9 ensure that two different primary lightpaths are not assigned the same channel \( k \) on a common edge \((i, j)\). The first three constraints are used to define the variable \( \delta_{ij}^{kp} \). Constraint 3.6 sets \( \delta_{ij}^{kp} \geq 1 \), whenever both \( x_{ij}^p = 1 \) and \( w_{kp} = 1 \). Constraints 3.7 and 3.8 force \( \delta_{ij}^{kp} \leq 1 \). Therefore, \( \delta_{ij}^{kp} = 1 \), if and only if, the following two conditions are satisfied:

i) \( p^{th} \) primary lightpath uses edge \((i, j)\) and

ii) \( p^{th} \) primary lightpath uses channel \( k \).

If either condition is not satisfied then, constraint 3.7 or 3.8 (or both) force \( \delta_{ij}^{kp} = 0 \).

Constraint 3.9 guarantees that, for all the primary lightpaths using edge \((i, j)\), at most one of them can be assigned wavelength \( k \). This ensures that each primary path is channel disjoint with every other primary path.

\[
w_{kp} + z_{kq} - \alpha_{pq}^k \leq 1, \quad \forall k \in K, \quad \forall p, q \in P
\]  
\[
-w_{kp} + \alpha_{pq}^k \leq 0, \quad \forall k \in K, \quad \forall p, q \in P
\]  
\[
-z_{kq} + \alpha_{pq}^k \leq 0, \quad \forall k \in K, \quad \forall p, q \in P
\]  

41
\[ x_{ij}^p + y_{ij}^q + \sum_{k \in K} \alpha_{pq}^k \leq 2, \ \forall (i, j) \in E, \ \forall p, q \in P \]  

Constraints 3.10 – 3.13 ensure that a primary lightpath and a backup lightpath are never assigned the same channel \( k \) on a common edge \((i, j)\). The first three constraints are used to define the variable \( \alpha_{pq}^k \). Constraint 3.10 sets \( \alpha_{pq}^k \geq 1 \), whenever both \( w_{kp} = 1 \) and \( z_{kp} = 1 \). Constraints 3.7 and 3.8 force \( \alpha_{pq}^k \leq 1 \). Therefore, \( \alpha_{pq}^k = 1 \), if and only if, the following two conditions are satisfied:

i) \( p^{th} \) primary lightpath uses channel \( k \) and

ii) \( q^{th} \) backup lightpath uses channel \( k \).

If either condition is not satisfied then, constraint 3.11 or 3.12 (or both) force \( \alpha_{pq}^k = 0 \).

According to the above definition, \( \alpha_{pq}^k = 1 \), implies that \( p^{th} \) primary lightpath and \( q^{th} \) backup lightpath both use channel \( k \). Therefore, \( \sum_{k \in K} \alpha_{pq}^k = 1 \), implies that \( p^{th} \) primary lightpath and \( q^{th} \) backup lightpath both use the same channel. In this case, Constraint 3.13 ensures that \( p^{th} \) primary lightpath and \( q^{th} \) backup lightpath do not share common edge \((i, j) \in E\).

### 3.2.4 Constraints for backup multiplexing

\[ y_{ij}^p + y_{ij}^q - \xi_{ij}^{pq} \leq 1, \ \forall (i, j) \in E, \ \forall p < q \in P \]  

\[ - y_{ij}^p + \xi_{ij}^{pq} \leq 0, \ \forall (i, j) \in E, \ \forall p < q \in P \]  

\[ - y_{ij}^q + \xi_{ij}^{pq} \leq 0, \ \forall (i, j) \in E, \ \forall p < q \in P \]  

Constraints 3.14 – 3.16 are used to define the variable \( \xi_{ij}^{pq} \). \( \xi_{ij}^{pq} \) is a continuous variable, which is set to 1 when both of the following conditions are satisfied:

i) \( p^{th} \) backup lightpath uses edge \((i, j)\)
ii) $q^{th}$ backup lightpath uses edge $(i,j)$.

If either condition is not satisfied then, constraint 3.15 or 3.16 (or both) force $\xi_{ij}^{pq} = 0$.

\begin{align}
\xi_{ij}^{pq} - \theta_{pq} & \leq 0, \quad \forall (i,j) \in E, \quad \forall p < q \in P \quad (3.17) \\
\theta_{pq} & \leq 1 \quad \forall p < q \in P \quad (3.18) \\
\theta_{pq} - \sum_{(i,j) \in E} \xi_{ij}^{pq} & \leq 0 \quad \forall p < q \in P \quad (3.19)
\end{align}

Constraints 3.17 - 3.19 all together define variable $\theta_{pq}$, which is set to 1 whenever $p^{th}$ and $q^{th}$ backup lightpaths share at least one edge. For example, according to definition, if $\xi_{ij}^{pq} = 1$, from 3.17 we get $1 \leq \theta_{pq}$, which along with constraint 3.18 forces $\theta_{pq} = 1$.

$\sum_{(i,j) \in E} \xi_{ij}^{pq}$ gives the total number of edges shared by $p^{th}$ and $q^{th}$ backup lightpaths. If they do not share any edge then $\sum_{(i,j) \in E} \xi_{ij}^{pq} = 0$ and constraint 3.19 will force $\theta_{pq} = 0$, otherwise constraints 3.17 and 3.18 will set $\theta_{pq} = 1$.

\begin{align}
z_{kp} + z_{kq} - \gamma_{pq}^k & \leq 1, \quad \forall k \in K, \quad \forall p < q \in P \quad (3.20) \\
- z_{kp} + \gamma_{pq}^k & \leq 0, \quad \forall k \in K, \quad \forall p < q \in P \quad (3.21) \\
- z_{kq} + \gamma_{pq}^k & \leq 0, \quad \forall k \in K, \quad \forall p < q \in P \quad (3.22)
\end{align}

Constraints 3.20 - 3.22 together define the continuous variable $\gamma_{pq}^k$. $\gamma_{pq}^k$ is defined in exactly the same way as $\alpha_{pq}^k$, except that in this case both lightpaths being considered are backup lightpaths. So, $\gamma_{pq}^k = 1$, implies that the $p^{th}$ and $q^{th}$ backup lightpaths both use channel $k$ and $\sum_{k \in K} \gamma_{pq}^k = 1$ implies that the $p^{th}$ and $q^{th}$ backup lightpaths are assigned the same channel.
\[ x^p_{ij} + x^q_{ij} + \theta_{pq} + \sum_{k \in K} \gamma^k_{pq} \leq 3, \quad \forall (i, j) \in E, \quad \forall p < q \in P \] (3.23)

If two backup lightpaths (p and q) are multiplexed, then

i) they share one or more edges (i.e. \( \theta_{pq} = 1 \)) and

ii) they are assigned the same channel (i.e. \( \sum_{k \in K} \gamma^k_{pq} = 1 \))

Constraint 3.23 then states that when two backup lightpaths (p and q) are multiplexed, the corresponding primary lightpaths cannot share a common edge. In other words, for a given edge \( (i, j) \in E \) \( x^p_{ij} \) and \( x^q_{ij} \) cannot both be 1. The need for this constraint can be illustrated by the following example (Figure 3-3).

![Diagram](image)

**Figure 3–3**: An example illustrating violation of constraint 3.23

In this example, two backup lightpaths b₁ and b₂ are multiplexed. They are both assigned wavelength \( \lambda_i \) and they share edge \( (3 \rightarrow 4) \). The corresponding primary lightpaths p₁ and p₂ also share an edge \( (1 \rightarrow 4) \). This clearly violates constraint 3.23. Now, if there is a fault on edge \( (1 \rightarrow 4) \), two different lightpaths b₁ and b₂ will be using \( \lambda_i \) on edge \( (3 \rightarrow 4) \) at the same time. This violates the basic WDM principle and leads to resource conflict.
Therefore, two backup lightpaths should be multiplexed only if they will never be used simultaneously. This means their corresponding primary paths must be fiber disjoint.

### 3.2.5 Constraints corresponding to transmitters and receivers

The number of lightpaths starting from or terminating at a node is directly related to the number of transmitters and receivers at that node. The relevant constraints are derived in this section. We consider two cases.

**Case 1:** There are no fiber failures.

Under fault free conditions, the following constraints must be satisfied.

\[
\sum_{p: o(p)=u} w_{kp} \leq i_u^k, \quad \forall k \in K, \quad \forall u \in N \tag{3.24}
\]

\[
\sum_{p: t(p)=u} w_{kp} \leq r_u^k, \quad \forall k \in K, \quad \forall u \in N \tag{3.25}
\]

Constraint 3.24 (3.25) simply states that the number of primary lightpaths originating (terminating) at node \(u\) and using channel \(k\), should not exceed the number of transmitters (receivers) at node \(u\) tuned to channel \(k\).

**Case 2:** There is a single fiber failure on edge \((i,j)\).

In this case the following constraints must be satisfied.

\[
x_{ij}^p + z_{kp} - \eta_{ij}^{kp} \leq 1, \quad \forall (i,j) \in \mathbb{E}, \quad \forall k \in K, \quad \forall p \in P \tag{3.26}
\]

\[
\eta_{ij}^{kp} - x_{ij}^p \leq 0, \quad \forall (i,j) \in \mathbb{E}, \quad \forall k \in K, \quad \forall p \in P \tag{3.27}
\]
\[ \eta_{ij}^{kp} - z_{kp} \leq 0, \quad \forall (i, j) \in E, \quad \forall k \in K, \quad \forall p \in P \quad (3.28) \]

Constraints 3.26-3.28 together define the variable \( \eta_{ij}^{kp} \). \( \eta_{ij}^{kp} \) is set 1 when the following two conditions are satisfied:

i) \( p^{th} \) primary lightpath uses edge \((i, j)\) and

ii) \( p^{th} \) backup lightpath uses channel \( k \)

If either one of the above conditions are not satisfied, then constraint 3.27 or 3.28 (or both) set \( \eta_{ij}^{kp} = 0 \).

\[ \sum_{p: o(p)=u} w_{kp} - \delta_{ij}^{kp} + \eta_{ij}^{kp} \leq t_u^k, \quad \forall (i, j) \in E, \quad \forall k \in K, \quad \forall u \in N \quad (3.29) \]

In constraint 3.29, \( \sum_{p: o(p)=u} w_{kp} \) gives the number of primary lightpaths at node \( u \), using transmitters tuned to channel \( k \), before any fiber failures. If edge \((i, j)\) fails, \( \sum_{p: o(p)=u} \delta_{ij}^{kp} \) provides the number of those primary lightpaths originating from node \( u \) and using channel \( k \), which are now destroyed. Similarly, \( \sum_{p: o(p)=u} \eta_{ij}^{kp} \) provides the number of new backup lightpaths, replacing the destroyed primary lightpaths originating from node \( u \), which use channel \( k \). Therefore, the total number of lightpaths (including the working primary paths and the new backup lightpaths replacing the destroyed primary paths) originating from node \( u \) and using channel \( k \) is \( \sum_{p: o(p)=u} w_{kp} - \delta_{ij}^{kp} + \eta_{ij}^{kp} \). Constraint 3.29 then states that this number should not exceed \( t_u^k \), the number of transmitters at node \( u \) tuned to channel \( k \).
\[ \sum_{p: (i(p)=u)} w_k^p - \delta_{ij}^p + \eta_{ij}^p \leq r_u^k, \quad \forall (i,j) \in E, \quad \forall k \in K, \quad \forall u \in N \]  \tag{3.30}

In a similar manner, constraint 3.30 ensures that the total number of lightpaths using channel \( k \) and terminating at node \( u \) does not exceed \( r_u^k \), the number of receivers at node \( u \) tuned to channel \( k \).

### 3.2.6 Traffic flow constraints

\[ f_p^{sd} - b_p \lambda^{sd} \leq 0, \quad \forall p \in P, \quad \forall s,d \in N, \ s \neq d \]  \tag{3.31}

Constraint (3.31) ensures that:

i) If \( p^{th} \) logical edge is not selected (i.e. \( b_p = 0 \)), then the amount of traffic on the \( p^{th} \) logical edge is 0 and

ii) If \( p^{th} \) logical edge is selected (i.e. \( b_p = 1 \)), the amount of \( s \)-\( d \) traffic (traffic originating from \( s \) and destined to \( d \)) on \( p^{th} \) logical edge cannot exceed \( \lambda^{sd} \), the total amount of traffic between \( s \) and \( d \).

\[ \varepsilon b_p - \sum_{s \neq d} f_p^{sd} \leq 0, \quad \forall p \in P, \quad \forall s,d \in N, \ s \neq d \]  \tag{3.32}

Constraint (3.32) ensures that the total amount of traffic on any selected logical edge cannot be less than \( \varepsilon \). This constraint is used to eliminate logical edges that carry very little traffic.

\[ \sum_{p: (i(p)=i)} f_p^{sd} - \sum_{p: (i(p)=i)} f_p^{sd} = \begin{cases} 
\lambda^{sd}, & \text{if } i = s \\
-\lambda^{sd}, & \text{if } i = d \\
0, & \text{otherwise}.
\end{cases} \quad \forall s,d \in N, \ s \neq d \]  \tag{3.33}
Constraint (3.33) is used to route the traffic, from a given source node $s$ to a given destination node $d$, over the logical edges. For any node $i \in N$, if $i = s$ ($i$ is the source node): the sum of the $s$-$d$ traffic flow on all lightpaths that originated from $i$ will be the value of $\lambda_{sd}^i$ (the total amount of traffic between $s$ and $d$). If $i = d$ ($i$ is the destination node), then the sum of the $s$-$d$ traffic on all lightpaths that destined to $i$ will be the value of $\lambda_{sd}^i$. If $i$ is intermediate node, then the total amount of $s$-$d$ traffic on all lightpaths that come into the node $i$ will be the same as the total amount of $s$-$d$ traffic on all lightpaths that go out of the node $i$.

$$\sum_{s \neq d} f_{p}^{sd} - F_{\text{max}} \leq 0, \forall p \in P$$ (3.34)

As mentioned earlier, $F_{\text{max}}$ is defined as the maximum traffic flow on any logical edge. Constraint (3.34) ensures the total amount of traffic on any lightpath cannot exceed $F_{\text{max}}$.

$$\sum_{s \neq d} f_{p}^{sd} \leq 1, \forall p \in P$$ (3.35)

In our formulation, all traffic values are normalized with respect to the maximum capacity of a lightpath. Therefore, constraint (3.35) ensures that the total amount of traffic on any lightpath does not exceed 1, its maximum capacity.

### 3.3 A Heuristic approach

In our heuristic approach, we divided the problem into two subproblems:

- logical topology design with shared path protection and
- traffic routing over the logical topology.

Our investigation concentrated on logical topology design with shared path protection and has been reported in detail in [MH03]. The routing problem will be done by another graduate student.
We developed a heuristic algorithm, which can create a logical topology in a very short time. To simplify the problem, we used a very simple routing scheme where we assumed that the entire traffic between each source/destination node pair (e.g. each entry $t_{sd}$ in the traffic matrix $A$) is routed on one route. To handle the traffic $t_{sd}$, we established a route $s \rightarrow A_1 \rightarrow A_2 \rightarrow \ldots A_k \rightarrow d$ from the source $s$ to the destination $d$ such that $s$ is connected to $A_1$ by a logical edge, $A_1$ is connected to $A_2$ by a logical edge, $\ldots$, $A_k$ is connected to $d$ by a logical edge. In this process, some of the logical edges used in the path from $s$ to $d$ are logical edges created earlier to handle the traffic due to some other source destination pair. If some pre-existing logical edges are used, each of these pre-existing logical edges must have sufficient spare capacity to carry the additional traffic $t_{sd}$. We define the "cost" of a route from a source $s$ to a destination $d$ to be the number of additional logical edges that we need to establish the route. For example, suppose our heuristic determines that the traffic $t_{sd}$ should use multi-hop route $s \rightarrow a \rightarrow b \rightarrow d$. If $s \rightarrow a$ and $b \rightarrow d$ are the new logical edges and $a \rightarrow b$ is an existing logical edge, then the total cost to transmit traffic $t_{sd}$ is $(1 + 0 + 1) = 2$.

The heuristic attempts to find the "minimum cost" path from $s$ to $d$, where we use existing logical edges as much as possible to minimize the number of new logical edges created to handle the traffic from $s$ to $d$.

The basic idea of our heuristic algorithm is that we sort traffic in descending order, try, in a greedy manner, to solve the high traffic first. There are several reasons for adopting this approach. We recall that the number of new lightpaths created is the cost of handling the traffic $t_{sd}$. At the beginning, there is more chance of creating a logical edge from a source to a destination. A situation where most of the paths from sources to the respective destinations also uses less resources of the network. Later on, when considering source destination pairs with low traffic, it is more likely that the low traffic can share the spare capacity of existing logical edges; therefore we can make more efficient use of the network capacity.
Given information about the network i.e., the physical topology (the number of nodes, the
edges), the number of wavelengths per fiber, the number of transmitters and receivers
tuned at each wavelengths and the traffic matrix \( A \), our heuristic algorithm is as follows:

Step1: Create, if possible, 3 edge-disjoint paths for each of the \( n(n-1) \) node pairs
using dijkstra's algorithm [Ski97] for determining the shortest path.

Step2: Sort all the traffic in the given traffic matrix in descending order.

Step3: While all nonzero entries in \( A \) have not been considered, repeat Step 4 –
Step 7.

Step4: Take the highest traffic in \( A \). Let this be \( t_{sd} \) from source \( s \) to destination \( d \).

Step5: Create a set \( Z \) of nodes initially consisting only the source node \( s \). Create
two sets \( Z_{old} \) and \( Z_{new} \) of nodes both initially empty.

Step6: Add to set \( Z \) all nodes \( y \) such that
   - we can reach \( y \) from any node \( x \in Z \), using existing logical edges
     that have a spare capacity to carry the additional traffic \( t_{sd} \)
   - \( y \in Z_{old} \)

Step7: If \( Z \) includes the destination node \( d \), go to step 12.

Step8: Include in set \( Z_{new} \) all nodes \( y \) such that
   - the node \( y \) does not appear in \( Z_{old} \) or \( Z \)
   - the node \( y \) does not already appear in \( Z_{new} \)
   - it is possible to set up a lightpath from some node \( x, x \in Z \), using
     one of the three paths from \( x \) to \( y \) determined in step 1. Let the
     path used be \( P1 \).
   - it is possible to set up a lightpath from some node \( x \) using one of
     the three paths from \( x \) to \( y \) determined in step 1. Let the path
     used be \( P2 \) where \( P2 \) must be such that \( P1 \neq P2 \).

Step9: If \( Z_{new} \) is empty, report failure and stop.

Step10: If \( Z_{new} \) includes the destination node \( d \), go to step 12.

Step11: Set \( Z_{old} = Z_{old} \cup Z \), \( Z = Z_{new} \), \( Z_{new} = \phi \). Go to step 6.

Step12: Set up required new pairs of lightpaths from the set of path-pairs \( (P1, P2) \)
as found in Step 8 so that a route is established from \( s \) to \( d \). Each of these
pairs of new lightpaths corresponds to a new edge in the logical topology. Each edge in this route from s to d will now carry an additional traffic $t_{sd}$. Set $t_{sd}$ to zero.

Step13: If $A$ has at least one non-zero entry, go to step 4. Otherwise, report success and stop.

It may be readily verified that this is a heuristic, which attempts to find a minimum cost route between all communicating source destination pairs in a greedy manner. The actual heuristic we used also attempts further optimization by using shared path protection. Details are available in [MH03].

Some explanation of these steps is given below. In Step 1, these shortest paths provide the routes for the primary and backup lightpaths for each source destination pairs. Because in shared path protection, for each primary lightpath, its corresponding backup lightpath must be edge-disjoint. So it is convenient to only consider edge-disjoint paths. Dijkstra’s algorithm [Ski97] is a well know algorithm for finding shortest path. Three shortest paths ($n = 3$) for every node pair provide enough route choices for the primary path and the backup path in most networks [RM98].

In step 2, the reason why we sort the traffic in descending order is already explained in previous paragraph.

In step 5, we check if there is enough spare capacity in some existing lightpaths to carry the traffic $t_{sd}$. If there exist nodes $A, B, \ldots X$ such that there are existing lightpaths $s \rightarrow A$, $A \rightarrow B$, $\ldots$, $X \rightarrow d$, and each of the lightpaths have at least $t_{sd}$ spare capacity then these existing lightpaths can carry the additional traffic $t_{sd}$ without adding any additional lightpath to the network. We choose the sequence lightpaths $s \rightarrow A \rightarrow B \rightarrow \ldots \rightarrow X \rightarrow d$, which can transmit the traffic $t_{sd}$ by a minimum number of hops.
In Step 8, if traffic $t_{sd}$ cannot be transmitted by existing lightpaths, we try to find one new lightpath for it (e.g. a new single-hop). The reason why we prefer new single-hop is already explained in previous paragraph.

In step 8 to 12, if we have to create a new lightpath, we actually try to find the two dedicated lightpaths first without considering backup lightpath multiplexing at this stage, this can be sure that we find both primary and backup path at the same time and make the problem simple, but obviously this is a greedy approach.
Chapter 4
Analysis and Results

In this chapter we will analyze the complexity of our MILP formulation and compare it to the only other existing formulation for the complete logical topology design problem [SRM02]. We will also solve the MILP for a number of small networks, and compare the results to those obtained using our heuristics.

4.1 Complexity Analysis

For our complexity analysis, we consider an arbitrary physical network with:

- Number of nodes = \( n \)
- Number of edges = \( m \)
- Number of channels per fiber = \( K \) and
- Number of potential logical edges = \( P = n(n-1) \)

We will measure the complexity of the MILP formulation in terms of three parameters:

i) the number of integer variables
ii) the number of continuous variables and
iii) the number of constraints

Of these three parameters, the most important is the number of integer variables. As mentioned in chapter 3, there are three types of integer variables. The number of each type of variable is given below.
<table>
<thead>
<tr>
<th>Type of integer variable</th>
<th>Number of variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical edge selection variables</td>
<td>$b_p$</td>
</tr>
<tr>
<td></td>
<td>$P = n(n-1)$</td>
</tr>
<tr>
<td>Route assignment variables $x_{ij}^p$, $y_{ij}^p$</td>
<td>$2mP = 2mn(n-1)$</td>
</tr>
<tr>
<td>Channel assignment variables $w_{kp}$, $z_{kp}$</td>
<td>$2KP = 2Kn(n-1)$</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>$n(n-1)[2m+2K+1]$</td>
</tr>
</tbody>
</table>

The different continuous variables and the number of such variables are:

<table>
<thead>
<tr>
<th>Type of continuous variable</th>
<th>Number of variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_{ij}^p$</td>
<td>$mKP$</td>
</tr>
<tr>
<td>$\alpha_{pq}^k$</td>
<td>$KP^2$</td>
</tr>
<tr>
<td>$\gamma_{pq}^k$</td>
<td>$KP(P-1)/2$</td>
</tr>
<tr>
<td>$\xi_{ij}^p$</td>
<td>$mP(P-1)/2$</td>
</tr>
<tr>
<td>$\theta_{pq}$</td>
<td>$P(P-1)/2$</td>
</tr>
<tr>
<td>$\eta_{ij}^p$</td>
<td>$mKP$</td>
</tr>
<tr>
<td>$f_{rd}^p$</td>
<td>$Pn(n-1)=P^2$</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>$2mKP+(K+1)P^2+[K+m+1] P(P-1)/2$</td>
</tr>
</tbody>
</table>

Each constraint of the MILP, in chapter 3, is actually a composite constraint, which generates a number of individual constraints. In order to illustrate this, we will examine one composite constraint in detail. The others can be expanded in the same manner. For example, we consider composite constraint (3.5), which ensures that a primary lightpath must be fiber disjoint with its corresponding backup lightpath.

$$x_{ij}^p + y_{ij}^p - b_p \leq 0, \quad \forall (i,j) \in E, \quad \forall p \in P \quad (3.5)$$

Constraint 3.5 states that an edge $(i,j)$ in the physical topology cannot be used by both the primary and the backup lightpaths associated with the $p^{th}$ logical edge. This constraint
must be satisfied for each of the \( P \) potential logical edges. In addition, for a given \( (p^{th}) \) logical edge, constraint 3.5 must be satisfied for each edge \((i, j) \in E\). Since there are \( m \) edges in the physical topology, composite constraint 3.5 actually corresponds to \( mP \) individual constraints. For example, Figure 4-1 shows a physical network with \( n=3 \) nodes, \( m=6 \) edges and \( P=2 \) potential logical edges.

![3-node network with two logical edges](image)

Figure 4-1: 3-node network with two logical edges

Therefore composite constraint 3.3 will generate \( mP = 12 \) individual constraints as shown below.

<table>
<thead>
<tr>
<th>Physical Edge</th>
<th>Potential Logical Edge</th>
<th>( L_0 )</th>
<th>( L_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>((1,2))</td>
<td>( x_{12}^0 + y_{12}^0 - b_0 \leq 0 )</td>
<td>( x_{12}^1 + y_{12}^1 - b_1 \leq 0 )</td>
<td></td>
</tr>
<tr>
<td>((2,1))</td>
<td>( x_{21}^0 + y_{21}^0 - b_0 \leq 0 )</td>
<td>( x_{21}^1 + y_{21}^1 - b_1 \leq 0 )</td>
<td></td>
</tr>
<tr>
<td>((1,3))</td>
<td>( x_{13}^0 + y_{13}^0 - b_0 \leq 0 )</td>
<td>( x_{13}^1 + y_{13}^1 - b_1 \leq 0 )</td>
<td></td>
</tr>
<tr>
<td>((3,1))</td>
<td>( x_{31}^0 + y_{31}^0 - b_0 \leq 0 )</td>
<td>( x_{31}^1 + y_{31}^1 - b_1 \leq 0 )</td>
<td></td>
</tr>
<tr>
<td>((3,2))</td>
<td>( x_{32}^0 + y_{32}^0 - b_0 \leq 0 )</td>
<td>( x_{32}^1 + y_{32}^1 - b_1 \leq 0 )</td>
<td></td>
</tr>
<tr>
<td>((2,3))</td>
<td>( x_{23}^0 + y_{23}^0 - b_0 \leq 0 )</td>
<td>( x_{23}^1 + y_{23}^1 - b_1 \leq 0 )</td>
<td></td>
</tr>
</tbody>
</table>
In a similar way, it can be readily shown that each of the constraints given in section 3.2 correspond to a group of individual constraints. The following table shows the number of individual constraints generated from each of the composite constraints given in chapter 3.

<table>
<thead>
<tr>
<th>Type of composite constraint</th>
<th>Number of individual constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route creation constraints 3.1, 3.2</td>
<td>$2nP$</td>
</tr>
<tr>
<td>Channel assignment constraints 3.3, 3.4</td>
<td>$2P$</td>
</tr>
<tr>
<td>Constraints for defining $\delta_{ij}^k$ 3.6-3.8</td>
<td>$3mKP$</td>
</tr>
<tr>
<td>Constraint ensuring all primary lighpaths are channel disjoint 3.9</td>
<td>$mK$</td>
</tr>
<tr>
<td>Constraints for defining $\alpha_{pq}^k$ 3.10 - 3.12</td>
<td>$3KP^2$</td>
</tr>
<tr>
<td>Constraint ensuring primary and backup lighpaths are channel disjoint 3.13</td>
<td>$mP^2$</td>
</tr>
<tr>
<td>Constraints for defining $\zeta_{ij}^p$ 3.14-3.16</td>
<td>$3mP(P-1)/2$</td>
</tr>
<tr>
<td>Constraints for defining $\omega_{pq}$ 3.17-3.19</td>
<td>$(m+2)P(P-1)/2$</td>
</tr>
<tr>
<td>Constraints for defining $\gamma_{pq}^k$ 3.20-3.22</td>
<td>$3KP(P-1)/2$</td>
</tr>
<tr>
<td>Constraints for backup multiplexing: 3.23</td>
<td>$mP(P-1)/2$</td>
</tr>
<tr>
<td>Transmitter/Receiver constraints (without faults): 3.24, 3.25</td>
<td>$2nK$</td>
</tr>
<tr>
<td>Constraints for defining $\eta_{ij}^{kp}$ 3.26-3.28</td>
<td>$3mKP$</td>
</tr>
<tr>
<td>Transmitter/Receiver constraints (with faults): 3.29, 3.30</td>
<td>$2mnK$</td>
</tr>
<tr>
<td>Traffic flow constraints: 3.31-3.35</td>
<td>$2P^2 + 3P$</td>
</tr>
<tr>
<td>Total number of constraints</td>
<td>$mK + 2nK + 2mnK + P[4.5K + 3.5m + 3] + P[2n + 6mK + 4 - 2.5m - 1.5K]$</td>
</tr>
</tbody>
</table>

The number of integer variables, continuous variables and constraints in the MILP formulation given in [SRM02] are given below. Here, $m$, $n$, $K$ and $P$ are defined in the
same manner as in our formulation and $r = 3$ is the number of number of physical routes considered for each lightpath.

<table>
<thead>
<tr>
<th>Number of integer variables</th>
<th>$2m + mK + 2nK + P[r^2K + rK + 1]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of continuous variables</td>
<td>$P^2$</td>
</tr>
<tr>
<td>Number of constraints</td>
<td>$P[rK + n + 4] + K[m^2 + 2n + 2mn + 7m] + 3m$</td>
</tr>
</tbody>
</table>

Table 4–1 illustrates how quickly these numbers increase with the size of the network and the number of available wavelengths per fiber. We see that, even for moderate sized networks, with a relatively small number of channels per fiber, the problem quickly becomes infeasible.

<table>
<thead>
<tr>
<th>No. of nodes</th>
<th>No. of Wavelength</th>
<th>No. of int variables</th>
<th>No. of cont. variable</th>
<th>No. of constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Ours</td>
<td>Muk.</td>
<td>Ours</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>444</td>
<td>1328</td>
<td>4470</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>636</td>
<td>2624</td>
<td>8070</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>1020</td>
<td>5216</td>
<td>15270</td>
</tr>
<tr>
<td>14</td>
<td>8</td>
<td>18382</td>
<td>18298</td>
<td>12.6×10^5</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>21294</td>
<td>36330</td>
<td>17.8×10^5</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>27118</td>
<td>73294</td>
<td>28.2×10^5</td>
</tr>
<tr>
<td>20</td>
<td>8</td>
<td>67260</td>
<td>37980</td>
<td>81.9×10^5</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>73340</td>
<td>75420</td>
<td>10.4×10^6</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>85500</td>
<td>150300</td>
<td>15.6×10^6</td>
</tr>
</tbody>
</table>

Table 4–1: Comparison of number of variables and constraints

As mentioned earlier, the most important factor affecting the complexity of the MILP is the number of integer (binary) variables in the formulation. The complexity increases exponentially with the number of binary variables. So, if a formulation has $N$ binary variables, CPLEX may potentially have to solve $2^N$ linear programs, where
N = n (n-1)[2m+2k+1]. Typically, this number can be somewhat reduced by using branch and bound techniques, but it is still exponential in N. Therefore, it is important to study how the number of integer variables increase with the size of the network.

We see from Table 4–1 that the number of continuous variables and the number of constraints is higher in our formulation compared to that in [SRM02]. This is because we have deliberately defined some variables as continuous variables, even though they could have been formulated as binary variables. For example, \( \delta_{ij}^p \), \( \alpha_{ij}^k \), \( \xi_{ij} \), \( \theta_{pq} \), \( \gamma_{pq} \), and \( \eta_{ij}^p \) are all continuous variables. However, the possible values for these variables are always restricted to 0 or 1, by the constraints used in defining these variables. Defining the above variables as continuous variables requires us to put additional constraints. But the cost associated with this is very small compared to the benefit — a significant reduction in the number of integer variables. This effect becomes clearly evident, when we consider realistic values for the number of wavelengths per fiber.

### 4.2 Implementation

In order to solve our MILP formulation for different networks we used a well-known and widely used linear programming tool, CPLEX [IL01]. However, CPLEX is not able to handle a generalized formulation, with composite constraints, as given in chapter 3. Therefore, it is necessary to generate the set of individual constraints before CPLEX can be used. The set of individual constraints must be presented in a specific format (called LP format) [IL01], which can be understood by CPLEX. The final set of (individual) constraints depends on the specifications of the particular design problem, such as the size and connectivity of the physical topology, the capacity of the fibers and the number of transceivers at each node. From Table 4-1, it is clear that there may be hundreds of decision variables and thousands constraints generated, even for small networks.

Clearly, it is not feasible to generate all these constraints manually. Therefore, we have implemented a program (in C) to automatically generate them in the proper format. The main steps in the implementation are given below.
Step 1: From an input file read the following network specifications:

- The set of nodes
- The set of edges
- The number of channels per fiber
- The number of transmitters and receivers at each node, tuned to a given wavelength

Step 2: Based on the above information, generate the set of potential logical edges P.

Step 3: For each composite constraint, generate all relevant individual constraints in the proper format.

Step 4: For each decision variable, generate the upper and lower bounds for that variable

Step 5: Write to an output file

- The objective function
- All individual constraints
- Bounds on all decision variables

The details of the program are straightforward and included in the appendix. The above program generates a file (in LP format). This file is then given as input to the CPLEX solver, which assigns values to all decision variables, to obtain an optimal solution.

4.3 Experimental Results

As mentioned earlier, the complete MILP formulation can only be solved for very small networks. Therefore, we were only able to perform a very limited amount of testing. We tested two small physical topologies with different number of channels and with different traffic matrices. In each case we used two transmitters and two receivers per channel per node. The results of our experiments are shown below in Table 4–2.
<table>
<thead>
<tr>
<th># of nodes</th>
<th>Traffic Matrix</th>
<th># of channels</th>
<th># of lightpaths</th>
<th>Congestion</th>
<th>Solution time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>MILP</td>
<td>heuristic</td>
<td>MILP</td>
</tr>
<tr>
<td>3</td>
<td>T1</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>T2</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>0.8</td>
</tr>
<tr>
<td>4</td>
<td>T3</td>
<td>4</td>
<td>10</td>
<td>6</td>
<td>0.11</td>
</tr>
<tr>
<td></td>
<td>T4</td>
<td>4</td>
<td>N/A</td>
<td>10</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 4–2: Some comparative results using our MILP and our heuristic

In Table 4–2, T1 and T3 are traffic matrices with low traffic for the 3-node and 4-node networks respectively and T2 and T4 are traffic matrices with relatively high traffic for the two networks. The actual traffic matrices are shown below.

\[
T_1 = \begin{bmatrix}
0.00 & 0.15 & 0.30 \\
0.80 & 0.00 & 0.50 \\
0.70 & 0.20 & 0.00
\end{bmatrix} \quad T_2 = \begin{bmatrix}
0.00 & 0.60 & 0.90 \\
0.80 & 0.00 & 0.70 \\
0.70 & 0.90 & 0.00
\end{bmatrix}
\]

\[
T_3 = \begin{bmatrix}
0.00 & 0.015 & 0.03 & 0.07 \\
0.08 & 0.00 & 0.05 & 0.09 \\
0.07 & 0.02 & 0.00 & 0.06 \\
0.09 & 0.07 & 0.04 & 0.00
\end{bmatrix} \quad T_4 = \begin{bmatrix}
0.00 & 0.15 & 0.30 & 0.70 \\
0.80 & 0.00 & 0.50 & 0.90 \\
0.70 & 0.20 & 0.00 & 0.60 \\
0.90 & 0.70 & 0.40 & 0.00
\end{bmatrix}
\]

### 4.3.1 Solution Details

In this section, we will examine one of the solutions, generated by CPLEX, in detail. We consider, as an example, the 3-node network of Figure 4–2, with traffic matrix T1 and two channels per fiber.
Figure 4–2: A 3-node physical topology.

<table>
<thead>
<tr>
<th>Logical Edge</th>
<th>Total traffic on logical edge</th>
<th>Primary lightpath</th>
<th>Backup lightpath</th>
<th>Wavelength assigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0</td>
<td>0.15</td>
<td>P0: 0 – 1</td>
<td>B0: 0 – 2 – 1</td>
<td>$\lambda_0$</td>
</tr>
<tr>
<td>L1</td>
<td>0.30</td>
<td>P1: 0 – 2</td>
<td>B1: 0 – 1 – 2</td>
<td>$\lambda_1$</td>
</tr>
<tr>
<td>L2</td>
<td>0.80</td>
<td>P2: 1 – 0</td>
<td>B2: 1 – 2 – 0</td>
<td>$\lambda_1$</td>
</tr>
<tr>
<td>L3</td>
<td>0.50</td>
<td>P3: 1 – 2</td>
<td>B3: 1 – 0 – 2</td>
<td>$\lambda_0$</td>
</tr>
<tr>
<td>L4</td>
<td>0.70</td>
<td>P4: 2 – 0</td>
<td>B4: 2 – 1 – 0</td>
<td>$\lambda_0$</td>
</tr>
<tr>
<td>L5</td>
<td>0.20</td>
<td>P5: 2 – 1</td>
<td>B5: 2 – 0 – 1</td>
<td>$\lambda_1$</td>
</tr>
</tbody>
</table>

Table 4–3: Solution details for a 3-node network

Table 4–3 shows the physical routes and channel assignments for all selected logical edges in the final solution as well as the total traffic on each logical edge. Table 4–4 shows how the traffic, for each source destination pair, is routed over the logical topology.
<table>
<thead>
<tr>
<th>Source – Destination</th>
<th>Amount of traffic</th>
<th>Traffic routed over different logical edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 → 1</td>
<td>0.15</td>
<td>L1 (0.15) → L5 (0.15)</td>
</tr>
<tr>
<td>0 → 2</td>
<td>0.30</td>
<td>L0 (0.15) → L3 (0.15) ; L1 (0.15)</td>
</tr>
<tr>
<td>1 → 0</td>
<td>0.80</td>
<td>L2 (0.75) ; L3 (0.05) → L4 (0.05)</td>
</tr>
<tr>
<td>1 → 2</td>
<td>0.50</td>
<td>L3 (0.50)</td>
</tr>
<tr>
<td>2 → 0</td>
<td>0.70</td>
<td>L4 (0.70)</td>
</tr>
<tr>
<td>2 → 1</td>
<td>0.20</td>
<td>L5 (0.20)</td>
</tr>
</tbody>
</table>

Table 4–4: Routing over the logical topology for the 3-node network

From Table 4–4 we can see that some traffic goes through multiple hops, e.g. the traffic from 0 → 1 is routed over logical edges L1 and L5. In addition, the total traffic between two nodes may be split up and distributed over different routes. For example, as shown in Figure 4–3, the total amount of traffic from node 0 → 2, $\lambda_{02} = 0.30$. This traffic is distributed on two routes:

i) a single hop route traversing logical edge L1 and

ii) a two-hop route (0 → 1 → 2) traversing logical edges L0 and then logical edge L3.

![Figure 4–3: Routing for source destination pair (0, 2)](image-url)
Some of the primary and backup lightpaths are shown in figure 4–4.

Figure 4–4: Some lightpaths for the 3-node network

The results for the 4-node network is similar. We were unable to consider real-life networks such as the NSF network due to the large number of integer variables.
Chapter 5
Critical Summary

Logical topology design with shared path protection is a very complicated problem, with no known tractable solution for the general case. The MILP formulation, even for partially relaxed versions of the problem, is computationally very difficult. Researchers, in an attempt for a practical method have looked even more relaxed versions [SRM02]. At this stage the researchers are proposing heuristics whose performance characteristics are quite unknown.

It is well known that the number of integer variables increase the complexity of the problem in an exponential way. Our first approach is a formulation of the general problem using MILP. The novel feature is that we used as many continuous variables as possible to play the role of integer variables. We did this by defining the restrictions in such a way that the continuous variables were forced to have a value of 0 or 1. Our analysis of the number of integer variables and constraints reveal that this solution is better, for realistic values of the number of wavelengths (K) per fiber, than the proposed solution of the problem that fixes the paths to three paths.

However our MILP approach is still very time consuming. For the simple 4-node network shown in chapter 4, using ILOG CPLEX 7.5, the solution time was 88311.63 seconds, running nearly one week. From this we can conclude that the identification of the optimal logical topology using MILP approach is not very practical. This general solution is however still useful as a benchmark for heuristics.

The only feasible approach appears to be the use of heuristics. We decoupled the whole problem into two sub-problems:

- logical topology design problem and
- optimal routing on the logical topology
We have developed a heuristic approach that can solve the logical topology design with shared path protection much faster. Details of the heuristic approach and the experimental results are available in [MH03].

We suggest that better heuristics be developed for finding the logical topology. Even though the problem of routing, is less complicated than the logical topology design problem, it is still a challenge given that the number of channels per fiber is currently 200 in a laboratory setting and expected to rise significantly in the near future. More investigation is needed to find an efficient and tractable way to solve this problem.

[Ag02] A. Agrawal, "Algorithms for the Logical Topology Design in WDM All-Optical Networks", aagrawal@usc.edu, March 2002


http://www.coritel.it/publications/Opt_download/ConteONDM.pdf


http://atlas-conferences.com/c/a/e/x/35.htm


http://student.seas.gwu.edu/~hjlee/working.pdf


[MT01] E. Modiano, A. Narula-Tam, "Survivable routing of logical topologies in WDM

[Mu92a] B. Mukherjee, "WDM-Based Local Lightwave Networks Part I: Single-Hop

[Mu92b] B. Mukherjee, "WDM-Based Local Lightwave Networks Part II: Multihop

[Mu94] B. Mukherjee et al., "Some principles for designing a wide-area optical
1994.

0-07-044435-8.

[OB01] A. E. Ozdaglar, D. P. Bertsekas, "Routing and Wavelength Assignment in

[Qi00] C. Qiao, "Labeled Optical Burst Switching for IP-over-WDM Integration", IEEE

[Ra02] B. Rajagopalan, et al., "Signaling for Protection and Restoration in Optical Mesh
Networks", Internet Draft <draft-bala-protecion-restoration-signalling-00.txt>, work in
progress, 2002.


Appendix A: Glossary

APS – Automatic Protection Switching

CPLEX – C Programming Language + simplEX

D-MLTDA – Decreasing Multi-hop Logical Topology Design Algorithm

GLTDA – Greedy logical topology design algorithm

ILP – Integer Linear Programming

I-MLTDA – Increasing Multi-hop Logical Topology Design Algorithm

IP – Internet Protocol

ITU-T – International Telecommunications Union on Telecommunications

LP – Linear Programming

MILP – Mixed-Integer Linear Programming

NSF – National Science Foundation

OADM – Optical Add/Drop Multiplexer

OXC – Optical Cross-Connect

PLTDA – Pre-established Logical Topology Design Algorithm

RLTDA – Random Logical Topology Design Algorithm

RWA – Routing & Wavelength Assignment

SDH – Synchronous Digital Hierarchy

SMLTDA – Single-hop Maximization Logical Topology Design Algorithm

SONET – Synchronous Optical Network

UPSR – Unidirectional Path Switched Ring

WDM – Wavelength Division Multiplexing
# include <stdio.h>

int exists(int, int, int [], [], int);

#define MAXNUMNODE 40
#define MAXNUMEDGE 1000
#define MAXNUMPATH 1000
#define MAXNUMCHANNEL 10

main()
{
  int i = 0, j = 0, p = 0, q = 0, k = 0, u = 0, l = 1, f = 0, t, r, numedges, n, numchannel,
  numpaths, source, dest;
  int counter = 0;
  int from, to;
  char s[100000], s1[1000], ss[1000], sx[1000], sy[1000], sw[1000], sz[1000], sd[1000],
  sa[1000], se[1000], sr[1000], swr[1000], swt[1000], sdgr[1000], sdgt[1000], sf[1000],
  sfo[1000], st[1000];
  float a, A[MAXNUMNODE][MAXNUMNODE], e = 1.0;
  int edge[MAXNUMEDGE][2];

  int P[MAXNUMPATH][2],
      R[MAXNUMNODE][MAXNUMCHANNEL],
      T[MAXNUMNODE][MAXNUMCHANNEL];

  FILE *fp, *inp;

  /* read the #1 line from "in" */

  inp = fopen("in", "r");
  fp = fopen("out", "w");

  fscanf(inp, "%d %d %d" ,
         &n,
         &numedges,
         &numchannel);
  printf("%d %d %d\n",
         n, numedges, numchannel);

  numedges = numedges * 2;
  numpaths = l * n * (n-1);
/* create edge array */

for (i = 0; i < numedges; i++)
{
    fscanf(inp, "%d %d", &from, &to);
    edge[i][0] = from;
    edge[i][1] = to;
    printf("%d %d\n", edge[i][0], edge[i][1]);
    i++; 
    edge[i][0] = to;
    edge[i][1] = from;
    printf("%d %d\n", edge[i][0], edge[i][1]);
}

/* create traffic matrix array A */

for (i = 0; i < n; i++)
{
    for (j = 0; j < n; j++)
    {
        fscanf(inp, "%f", &a);
        A[i][j] = a;
        printf("%f", A[i][j]);
    }
    printf("\n");
}

/* create lightpath array P */

for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
    {
        if (i != j)
        {
            P[p][0] = i;
            P[p][1] = j;
            fprintf(fp, "%d, " , P[p][0]);
            fprintf(fp, "%d\n", P[p][1]);
            p++;
        }
    }
/* create receiver vector array R */

for (i = 0; i < n; i++)
{
    for (j = 0; j < numchannel; j++)
    {
        fscanf(inp, "%d", &r);
        R[i][j] = r;
        printf("%d", R[i][j]);
    }
    printf("\n");
}

/* create transmitter vector array T */

for (i = 0; i < n; i++)
{
    for (j = 0; j < numchannel; j++)
    {
        fscanf(inp, "%d", &t);
        T[i][j] = t;
        printf("%d", T[i][j]);
    }
    printf("\n");
}

/* Form the objective function */

fprintf(fp,"Minimize\n obj: Fmax\n");
fprintf(fp, "Subject to\n");

/* path creation and channel allocation, corresponding primary and backup paths must be fibre-disjoint */

for (p = 0; p < numpaths; p++)
{
    for (i = 0; i < n; i++)
    {
        strcpy(s, "");
        strcpy(ss, "");
        strcpy(sx, "");
        strcpy(sy, ");
        for (j = 0; j < n; j++)
        {
            
        }
if (exists(i, j, edge, numedges) == 1)
{
    sprintf(sx, "x%d_%d-%d-x%d_%d+%d", p, i, j, p, j, i);
    strcat(s, sx);

    sprintf(sy, "y%d_%d-%d-y%d_%d+%d", p, i, j, p, j, i);
    strcat(ss, sy);

    counter++;
    fprintf(fp, " c%d: x%d_%d-%d+y%d_%d-%d-b%d<=0\n",
            counter, p, i, j, p, j, i, p);
}

if(P[p][0] == i)
{
    sprintf(sx, "-b%d=0", p);
    sprintf(sy, "-b%d=0", p);
}
else
{
    if(P[p][1] == i)
    {
        sprintf(sx, "+b%d=0", p);
        sprintf(sy, "+b%d=0", p);
    }
    else
    {
        sprintf(sx, "=0");
        sprintf(sy, "=0");
    }
}

s[strlen(s)-1] = '\0';
ss[strlen(ss)-1] = '\0';
strcat(s, sx);
strcat(ss, sy);
counter++;
fprintf(fp, " c%d: %s\n", counter, s);
counter++;
fprintf(fp, " c%d: %s\n", counter, ss);
}

strcpy(s, "");
strcpy(ss, "");
strcpy(sw, "");
counter++;
sprintf(s, " c%d: ", counter);
counter++;
sprintf(ss, " c%d: ", counter);

for (k = 0; k < numchannel; k++)
{
    sprintf(sw, "w%d._%d+=", k, p);
    strcat(s, sw);
    sprintf(sz, "z%d._%d+=", k, p);
    strcat(ss, sz);
}
s[strlen(s)-1] = '\0';
ss[strlen(ss)-1] = '\0';

sprintf(sw, "-b%d=0", p);
sprintf(sz, "-b%d=0", p);

strcat(s, sw);
strcat(ss, sz);
fprintf(fp, "%s
", s);
fprintf(fp, "%s
", ss);
}

/* each primary path is edge-channel disjoint with every other primary path */

strcpy(sd, "");

for (k = 0; k < numchannel; k++)
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
        {
            if (exists(i, j, edge, numedges) == 1)
            {
                strcpy(s, "");
                for (p = 0; p < numpaths; p++)
                {
                    counter++;
                    fprintf(fp, " c%d: x%d._%d._%d+w%d._%d-d%d._%d_%d._%d<=1\n",
                            counter, p, i, j, k, p, k, p, i, j);

                    counter++;
                    fprintf(fp, " c%d: d%d._%d._%d-x%d._%d._%d<=0\n", counter, k, p,
                                i, j, p, i, j);
                }
            }
        }
counter++;
fprintf(fp, " c%d: d%d_%d_%d-%d-w%d_%d<=0\n", counter, k, p, i, j, k, p);

sprintf(sd, "d%d_%d_%d_%d",
          k, p, i, j);
strcat(s, sd);
}

counter++;
s[strlen(s)-1] = '\0';
fprintf(fp, " c%d: %s<=1\n", counter, s);

}

/* each primary path is edge-channel disjoint with every other backup path */

strcpy(sa, "");

for (p = 0; p < numpaths; p++)
  for (q = 0; q < numpaths; q++)
  {
    strcpy(s, "");
    for (k = 0; k < numchannel; k++)
    {
      counter++;
      fprintf(fp, " c%d: w%d_%d+z%d_%d-a%d_%d-%d<=1\n", counter, k, p, k, q, k, p, q);

      counter++;
      fprintf(fp, " c%d: -w%d_%d+a%d_%d-%d<=0\n", counter, k, p, k, p, q);

      counter++;
      fprintf(fp, " c%d: -z%d_%d+a%d_%d-%d<=0\n", counter, k, q, k, p, q);

      sprintf(sa, "a%d_%d_%d+%", k, p, q);
      strcat(s, sa);
    }

    s[strlen(s)-1] = '\0';

    for (i = 0; i < n; i++)
    {
      for (j = 0; j < n; j++)
      {

if (exists(i, j, edge, numedges) == 1)
{
    counter++;
    fprintf(fp, "c%d: x%d_%d_%d+y%d_%d_%d+%s<=2\n", counter, p, i, j, q, i, j, s);
}

/*/ two backup paths sharing a channel and an edge ==> corresponding primary paths fibre-disjoint */

strcpy(sr, """);
strcpy(se, "");

for (p = 0; p < numpaths; p++)
    for (q = 0; q < numpaths; q++)
    {
        if (p != q)
        {
            strcpy(s, "");
            for (i = 0; i < n; i++)
                for (j = 0; j < n; j++)
                {
                    if (exists(i, j, edge, numedges) == 1)
                    {
                        counter++;
                        fprintf(fp, "c%d: y%d_%d_%d+y%d_%d_%d-c%d_%d_%d_%d<=1\n",
                                counter, p, i, j, q, i, j, p, q, i, j);

                        counter++;
                        fprintf(fp, "c%d: -y%d_%d_%d+c%d_%d_%d_%d<=0\n", counter, p, i, j, p, q, i, j);

                        counter++;
                        fprintf(fp, "c%d: -y%d_%d_%d+c%d_%d_%d_%d<=0\n", counter, q, i, j, p, q, i, j);

                        counter++;
                        fprintf(fp, "c%d: c%d_%d_%d_%d-o%d_%d<=0\n", counter, p, q, i, j, p, q);

                        sprintf(se, "c%d_%d_%d_%d-", p, q, i, j);
                        sprintf(s, se);
strcpy(ss, "\n");
for (k = 0; k < numchannel; k++)
{
    sprintf(sr, "r%d_%d_%d\n", k, p, q);
    strcat(ss, sr);
}

counter++;
ss[strlen(ss)-1] = '\0';
fprintf(fp, " c%d: x%d_%d_%d+z%d_%d_%d-o%d_%d_%d+s<=3\n",
        counter, p, i, j, q, i, j, p, q, ss);
}

counter++;
fprintf(fp, " c%d: o%d_%d<=1\n", counter, p, q);

counter++;
s[strlen(s)-1] = '\0';
fprintf(fp, " c%d: o%d_%d-%d<=0\n", counter, p, q, s);

for (k = 0; k < numchannel; k++)
{
    counter++;
    fprintf(fp, " c%d: z%d_%d+z%d_%d-%r%d_%d_%d<=1\n", counter, k, p, k, q,
            k, p, q);
    counter++;
    fprintf(fp, " c%d: -z%d_%d-%r%d_%d_%d<=0\n", counter, k, p, k, p, q);
    counter++;
    fprintf(fp, " c%d: -z%d_%d-%r%d_%d_%d<=0\n", counter, k, q, k, p, p);
}

/* constraints corresponding to transmitters and receivers at each node */
/* case 1 */

strcpy(swrr, "\n");
strcpy(swrt, "\n");

for ( u = 0; u < n; u++)
for (k = 0; k < numchannel; k++)
{
    strcpy(s, ");
    strcpy(ss, "");
    for (p = 0; p < numpaths; p++)
    {
        if(P[p][1] == u)
        {
            sprintf(swr, "w%d_%d+%", k, p);
            strcat(s, swr);
        }
        if(P[p][0] == u)
        {
            sprintf(swt, "w%d_%d+%", k, p);
            strcat(ss, swt);
        }
    }
    counter++;
    s[strlen(s)-1] = ");
    fprintf(fp, "% c%d: %s<=%d\n", counter, s, R[u][k]);

    counter++;
    ss[strlen(ss)-1] = ");
    fprintf(fp, "% c%d: %s<=%d\n", counter, ss, T[u][k]);
}

/* case 2 */

for (p = 0; p < numpaths; p++)
    for (k = 0; k < numchannel; k++)
        for (i = 0; i < n; i++)
            for (j = 0; j < n; j++)
            {
            if (exists(i, j, edge, numedges) == 1)
            {
                counter++;
                fprintf(fp, "% c%d: x%d_%d_%d+z%d_%d-g%d_%d_%d_%d<=1\n",
                        counter, p, i, j, k, p, k, p, i, j);

                counter++;
                fprintf(fp, "% c%d: g%d_%d_%d_%d-x%d_%d_%d<=0\n", counter, k,
                        p, i, j, p, i, j);
counter++;
    fprintf(fp, "c%d: g%d_%d_%d_%d-%d-%d_<=0\n", counter, k, p, i, j, k, p);
}
}

strcpy(sdgr, "");
strcpy(sdgt, ");

for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
    {
        if (exists(i, j, edge, numedges) == 1)
        {
            for (u = 0; u < n; u++)
                for (k = 0; k < numchannel; k++)
                {
                    strcpy(s, "");
                    strcpy(ss, "");
                    for (p = 0; p < numpaths; p++)
                    {
                        if(P[p][0] == u)
                        {
                            sprintf(sdgt, "w%d_%d-%d_%d_%d_%d%+\", k, p, k, p, i, j, k, p, i, j);
                            strcat(s, sdgt);
                        }
                        if(P[p][1] == u)
                        {
                            sprintf(sdgr, "w%d_%d-%d_%d_%d_%d%+\", k, p, k, p, i, j, k, p, i, j);
                            strcat(ss, sdgr);
                        }
                    }
                    counter++;
                    ss[strlen(ss)-1] = '0';
                    fprintf(fp, "c%d: %s<=%d\n", counter, ss, R[u][k]);
                }
        counter++;
        s[strlen(s)-1] = '0';
        fprintf(fp, "c%d: %s<=%d\n", counter, s, T[u][k]);
    }
/* traffic flows and constrains on the logic topology */

/* e */

for (source = 0; source < n; source++)
    for (dest = 0; dest < n; dest++)
    {
        if ((A[source][dest] < e) && (A[source][dest] != 0.0))
            e = A[source][dest];
    }

/* 32, 33, 35, 36 */

strcpy(sf, "");

for (p = 0; p < numpaths; p++)
    {
        strcpy(s, "");
        strcpy(ss, "");
        for (source = 0; source < n; source++)
            for (dest = 0; dest < n; dest++)
            {
                if (source != dest)
                {
                    counter++;
                    fprintf(fp, " c%d: f%d_%d_%d-%fb%d<=0\n", counter, source, dest, p,
                            A[source][dest], p); /* 32 */
                    sprintf(sf, "f%d_%d_%d-", source, dest, p);
                    strcat(s, sf);
                    sprintf(sf, "f%d_%d_%d+", source, dest, p);
                    strcat(ss, sf);
                }
            }
        counter++;
        s[strlen(s)-1] = '\0';
        fprintf(fp, " c%d: %fb%d-%s<=0\n", counter, e, p, s); /* 33 */
        counter++;
        ss[strlen(ss)-1] = '\0';
        fprintf(fp, " c%d: %s-Fmax<=0\n", counter, ss); /* 35 */
        counter++;
        fprintf(fp, " c%d: %s<=1\n", counter, ss); /* 36 */
    }
strcpy(sfo, "");
strcpy(sfl, "");

for (source = 0; source < n; source++)
    for (dest = 0; dest < n; dest++)
        if (source != dest)
            {
            for (i = 0; i < n; i++)
                {
                strcpy(s, "");
                strcpy(ss, "");

                for (p = 0; p < numpaths; p++)
                    {
                    if(P[p][0] == i)
                        {
                        sprintf(sfo, "f%d_%d_%d+", source, dest, p);
                        strcat(s, sfo);
                        }
                    else
                        {
                        if(P[p][1] == i)
                            {
                            sprintf(sfl, "-f%d_%d_%d", source, dest, p);
                            strcat(ss, sfl);
                            }
                        }
                    } /* end if */
                } /* for p */
            }

if (i == source)
    {
    counter++;
    s[strlen(s)-1] = '0';
    fprintf(fp, " c%d: %s%s=%f
", counter, s, ss, A[source][dest]);
    }
else
    {
    if (i == dest)
        {
        counter++;
        s[strlen(s)-1] = '0';
        fprintf(fp, " c%d: %s%s=-%f\n", counter, s, ss, A[source][dest]);
        }
    else
        
    
91
{ 
    counter++;
    s[strlen(s)-1] = '0';
    fprintf(fp, " c%d: %s%0\n", counter, s, ss);
} /* end if */
} /* end if */
} /* end for i */
} /* end if */

/* bounds */

fprintf(fp, "Bounds\n");

for (p = 0; p < numpaths; p++)
{
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
        {
            if (exists(i, j, edge, numedges) == 1)
            {
                fprintf(fp, " 0<=x%d_%d_%d<=1\n", p, i, j);
                fprintf(fp, " 0<=y%d_%d_%d<=1\n", p, i, j);
            }
        }
    for (k = 0; k < numchannel; k++)
    {
        fprintf(fp, " 0<=w%d_%d<=1\n", k, p);
        fprintf(fp, " 0<=z%d_%d<=1\n", k, p);
    }
    fprintf(fp, " 0<=b%d<=1\n", p);
}

for (p = 0; p < numpaths; p++)
    for (source = 0; source < n; source++)
        for (dest = 0; dest < n; dest++)
        {
            if (source != dest)
            {
                fprintf(fp, " 0<=f%d_%d_%d\n", source, dest, p);
            }
        }

for (p = 0; p < numpaths; p++)
    for (k = 0; k < numchannel; k++)
        for (i = 0; i < n; i++)
            for (j = 0; j < n; j++)
            { 

if (exists(i, j, edge, numedges) == 1) {
    fprintf(fp, "0<=%d_%d_%d_%d\n", k, p, i, j);
    fprintf(fp, "0<=%d_%d_%d_%d\n", k, p, i, j);
}

for (p = 0; p < numpaths; p++)
    for (q = 0; q < numpaths; q++)
    {
        if (p != q)
        {
            for (k = 0; k < numchannel; k++)
            {
                fprintf(fp, "0<=%d_%d_%d_%d\n", k, p, q);
                fprintf(fp, "0<=%d_%d_%d_%d\n", k, p, q);
            }
            for (i = 0; i < n; i++)
                for (j = 0; j < n; j++)
                {
                    if (exists(i, j, edge, numedges) == 1)
                    {
                        fprintf(fp, "0<=%d_%d_%d_%d\n", p, q, i, j);
                    }
                }
            fprintf(fp, "0<=%d_%d\n", p, q);
        }
    }

fprintf(fp, "Binaries\n", s);

strcpy(s, ";");

counter = 0;
for (p = 0; p < numpaths; p++)
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
        {
            if (exists(i, j, edge, numedges) == 1)
            {
                counter++;
                if (counter > 20)
                {
                    counter = 0;
                    sprintf(s1, "%d_%d\n", p, i, j);
                    strcat(s, s1);
                    strcat(s, s1);
                    counter++;
                }
            }
sprintf(s1, "y%d_%d_%d\n", p, i, j);
strcat(s, s1);
}
else
{
sprintf(s1, "x%d_%d_%d", p, i, j);
strcat(s, s1);
counter++;
sprintf(s1, "y%d_%d_%d", p, i, j);
strcat(s, s1);
}
}
}

for (p = 0; p < numpaths; p++)
    for (k = 0; k < numchannel; k++)
    {
        counter++;
        if (counter > 20)
        {
            counter = 0;
            sprintf(s1, "w%d_%d\n", k, p);
            strcat(s, s1);
        }
        else
        {
            sprintf(s1, "w%d_%d", k, p);
            strcat(s, s1);
        }
    }

for (p = 0; p < numpaths; p++)
    for (k = 0; k < numchannel; k++)
    {
        counter++;
        if (counter > 20)
        {
            counter = 0;
            sprintf(s1, "z%d_%d\n", k, p);
            strcat(s, s1);
        }
        else
        {
            sprintf(s1, "z%d_%d", k, p);
            strcat(s, s1);
        }
    }
for (p = 0; p < numpaths; p++)
{
    counter ++;
    if (counter > 20)
    {
        counter = 0;
        sprintf(s1, " b%d\n", p);
        strcat(s, s1);
    }
    else
    {
        sprintf(s1, " b%d", p);
        strcat(s, s1);
    }
}

fprintf(fp, "%s\n", s);
fprintf(fp, "End\n", s);
fclose(fp);

int exists(int a, int b, int e[1000][2], int num)
{
    int k, flag = 0;

    for (k = 0; k < num; k++)
        if ((e[k][0] == a) && (e[k][1] == b))
        {
            flag = 1;
            break;
        }
    else
        flag = 0;

    return flag;
}
**VITA AUCTORIS**

<table>
<thead>
<tr>
<th>NAME:</th>
<th>Hong Guan</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLACE OF BIRTH</td>
<td>Beijing, China</td>
</tr>
<tr>
<td>YEAR OF BIRTH</td>
<td>1960</td>
</tr>
<tr>
<td>EDUCATION</td>
<td>Beijing University of Aeronautics and Astronautics</td>
</tr>
<tr>
<td></td>
<td>Beijing, China</td>
</tr>
<tr>
<td></td>
<td>1979-1983 B.Sc</td>
</tr>
<tr>
<td></td>
<td>University of Windsor</td>
</tr>
<tr>
<td></td>
<td>Windsor, Ontario, Canada</td>
</tr>
<tr>
<td></td>
<td>2000-2003 M.Sc</td>
</tr>
</tbody>
</table>