2004

Approximation algorithms for optimal routing in wavelength routed WDM network.

Yumei Lu
University of Windsor

Approximation Algorithms for Optimal Routing in Wavelength Routed WDM Network

By

Yumei Lu

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

2004

Your file Votre référence
ISBN: 0-612-96385-3
Our file Notre référence
ISBN: 0-612-96385-3

Abstract

Finding an optimum routing scheme, so that a given logical topology for a wavelength routed WDM network handles all traffic requirements in an efficient manner, is an important problem in optical network design. One major design objective is to minimize the congestion of the network, defined as the traffic on the logical link which has the maximum traffic. This is known to be a hard problem taking an enormous amount of time even for moderate sized networks and is intractable for larger networks. The approach we have outlined is that of obtaining an approximate solution to the problem rather than an exact solution. We have developed three closely related approximate algorithms to solve this problem. These three algorithms significantly improved the running time for finding the minimum congestion. We based our approach on the approximation fraction multi-commodity algorithm by Lisa Fleischer [F99] to calculate the primal and dual solutions for the linear program to solve the multi-commodity flow problem.

We have compared our algorithm with the standard linear program solution obtained using CPLEX. The experiments show that our algorithms require, on the average, only less than 10% of the time CPLEX uses for networks with 25 nodes or less. Our algorithms perform well for larger networks which CPLEX cannot handle.

Key Words: Congestion, Multi-commodity, Approximate solution, CPLEX

To my parents and my son

Acknowledgement

I would like to take this opportunity to express my sincere gratitude to my supervisor, Dr. Subir Bandyopadhyay and Dr. Arunita Jaekel who teaches me not only the knowledge, but also the research method and attitude. This work could not be achieved without their extensive guidance and constant encouragement. Also I would like to thank all the committee members, Dr. Xiang Chen, Dr. Liwu Li, and Dr. Dan Wu, for their valuable time and comments.

My special thanks go to my parents, my son, my sister and my brother. Their love and care have always been with me during these years. Their trust and encouragement pulled me through hard times.

I would like to thank everybody for offering the help for this thesis work.

Table of Contents

List of Tables

List of Figures

1. Introduction

With the rapid development of internet technology and usage of computer networks in recent years, the demand for high speed networks has been increasing dramatically. Optical networks are widely utilized in long-haul networks using Wavelength-Division Multiplexing (WDM). WDM is the most useful technology to meet the rapidly increasing user requirements for increased communication. In an optical network, the transmissions from many different end-users can be combined and transmitted on the same fiber [Muk97]. By transmitting data using a number of separate optical carriers on each fiber, with each optical carrier having a distinct wavelength, WDM technology can exploit the huge bandwidth of optical networks.

1.1 WDM network fundamentals

A typical WDM network, given in Figure 1.1, shows end-nodes, denoted by squares, typically representing computers that can be sources or destinations of data communication. Each circle denotes a router node - an optical device that has a number of incoming and outgoing fibers, each carrying optical signals. These routers can take an optical signal at some frequency say ω_1, using any incoming fiber and send it on any outgoing fiber using some optical frequency ω_2. In some scenarios the routers are

equipped with wavelength conversion devices so that the frequencies ω_1 and ω_2 may be distinct.

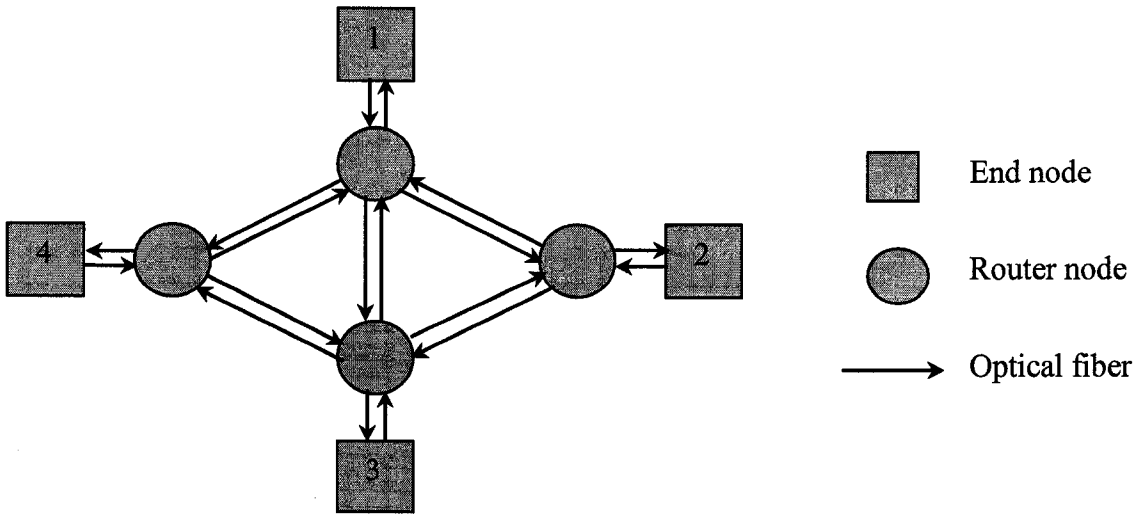


Figure 1.1 An Optical Network

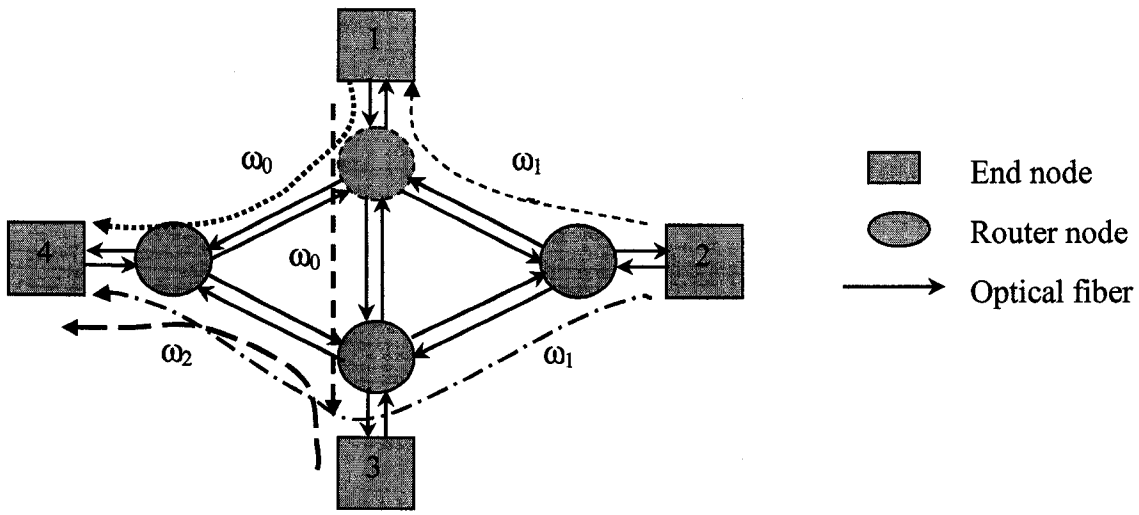


Figure 1.2 An optical network with lightpaths

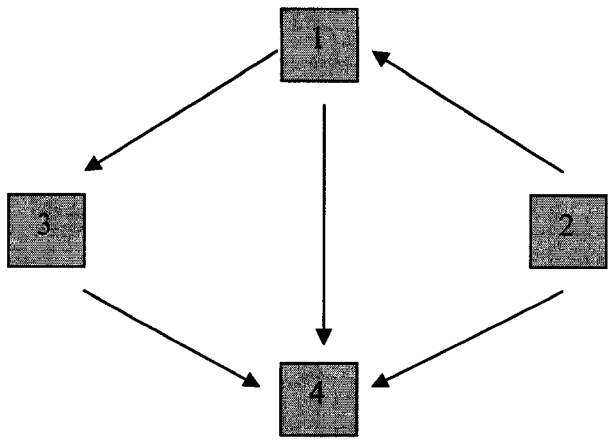


Figure 1.3 Logical topology of an optical network

The low impedance region of an optical fiber is 1200 -1600nm. This region may be divided into a number of intervals or bands so that each optical carrier on any given fiber operates within a given distinct band. Depending on the technology used and the characteristics of the fibers in the network, there can be up to 200 different carrier wavelengths on a single fiber [Muk97]. Each possible carrier wavelength on a given fiber is called a *channel*. A *lightpath* [Muk00] from a source s to a destination d is an all-optical circuit switched communication path between nodes s and d in the network. If a lightpath spans more than one fiber link, the intermediate nodes in the fiber path are optical routers that direct the lightpath in the optical domain from some incoming fiber to some outgoing fiber. In an all-optical WDM network, all communication uses lightpaths. If a channel is being used for some communication that channel cannot be used for any other communication. In order to communicate from a source s to a destination d, it must be possible to find a route $s \rightarrow x_1 \rightarrow x_2 \rightarrow \ldots x_k \rightarrow d$ such that

a) there exists a fiber connecting node s to node x_1, a fiber connecting node x_1 to node x_2, ... a fiber connecting node x_k to node d,

b) there exists a free channel on the fiber connecting node s to node x_1, a free channel on the fiber connecting node x_1 to node x_2, ... , a free channel on the fiber connecting node x_k to node d.

The logical topology is defined by the end-nodes (which are the sources or sinks of data) and the lightpaths which define the optical communications paths between the end-nodes. It is convenient to represent the logical topology using a directed graph where the nodes in the graph represent the end-nodes and the directed edges represent the lightpaths. For example, if the lightpaths on the physical topology shown in Figure 1.1 are as shown in Figure 1.2, the logical topology is as shown in Figure 1.3.

The designer has great control over the way the logical topology may be defined. Given the physical topology of an optical network, and the traffic requirements between each end-node in the network, the logical topology design problem is to find a set of lightpaths to carry the specified traffic using a minimum amount of network resources.

To represent the traffic requirements in a network, with say n nodes, it is convenient to use a n X n matrix. Such matrices are called traffic matrices. The entry $T_{i,j}$ in row i and column j of a traffic matrix T denoted the amount of traffic from end node i to end node j. For instance a traffic matrix for the network shown in Figure 1.1 may be as follows Figure1.4.

$$\text{Traffic Matrix} = \begin{pmatrix} 0 & 52 & 21 & 32 \\ 12 & 0 & 37 & \mathbf{14} \\ 41 & 21 & 0 & 33 \\ 13 & 42 & 12 & 0 \end{pmatrix}$$

Figure 1.4 Traffic Matrix

This means that the traffic from 2 to 4 is 14% of the capacity of a lightpath. At the moment, the capacity of a lightpath is 2.5 Gbits/second so that the traffic from 2 to 4 is 0.35 Gbits/second. The capacity of a lightpath may go up in the future as the technology progresses. Given a logical topology and a traffic matrix, it is necessary to find an optimal way to route the traffic between all end-node pairs having non-zero traffic. This is also a complicated problem. For example in the case of the physical network shown in Figure 1.1 and the traffic matrix *T*, the traffic of 0.35 Gbits/second from node 2 to node 4 may be handled in a variety of ways. For instance, in the case of the logical topology shown in Figure 1.3 there are 3 paths from 2 to 4 as follows:

1) Path 1-- Using the lightpath $2 \rightarrow 4$
2) Path 2 -- Using the lightpath $2 \rightarrow 1$ and then the lightpath $1 \rightarrow 4$
3) Path 3 -- Using the lightpaths $2 \rightarrow 1$, $1 \rightarrow 3$ and then $3 \rightarrow 4$.

The traffic of 0.35 Gbits/second from 2 to 4 may be sent entirely using only path 1, path 2 or path3. Alternatively a part of the traffic may be sent on path 1, a part on path 2 and the

rest using path 3. When developing an efficient scheme for routing, we have to consider all the traffic requirements as specified in the traffic matrix and devise a routing scheme so that the traffic on the logical edge carrying the maximum traffic is as low as possible. In this thesis we have studied this problem.

1.2 WDM routing problem viewed as multi-commodity flow problem

The routing problem in WDM networks is quite similar to the problems studied extensively in the operations research community and is known as the *multi-commodity network flow problem.* Multi-commodity network flow problem is very useful in optimizing transportation systems, warehousing, distribution networks etc [AMO93]. The idea in multi-commodity network flow is that we have a graph where the nodes represent sources and sinks of the commodities and edges represents a way to transport a commodity from one node to another. For example, to model a railroad system, the nodes will be warehouses in the rail stations storing different goods to be transported and the edges represent the train line connecting two stations. An item that needs to be transported from its originating station (usually called the source) to its destination is called a commodity. An important problem in planning a scheme for efficient transportation for such a network is to find an optimum way to send all the commodities from their respective sources to their corresponding destinations. It is also important to define what we are optimizing. We could, for example, in the case of the railway transportation problem, minimize the total cost of transportation, or make sure that there is as much spare capacity for future increases in our traffic requirements.

We can use the same idea for defining the routing problem in WDM networks. Each source-destination pair having a non-zero entry in the traffic matrix represents a commodity. Our problem is to minimize the congestion. This means that we wish to have a minimum amount of traffic on the logical edge that is carrying the maximum traffic. Informally, we wish to route the traffic in a way that the logical edges are carrying as little traffic as possible. This ensures that, if the traffic requirements increase in the future, the scheme does not have to be modified until the traffic on the edge carrying maximum traffic exceeds the capacity of a lightpath.

The problem of finding an efficient algorithm to find the minimum congestion in wavelength routed WDM networks may be specified using linear constraints so that a linear programming tool may be used to solve the problem. One straight-forward solution of the problem is to use a package like CPLEX to solve the problem specified as a linear program. A serious drawback of this approach is that, even for networks with 20 nodes, CPLEX takes a long time to get a solution. With larger networks, CPLEX simply cannot handle the problem. It is important to find reasonably efficient algorithms that can handle networks of practical size. This is the problem we have studied in this investigation.

1.3 Motivation of this investigation

In WDM networks, if we use a LP solver such as the CPLEX, we get an exact solution meaning that we get the absolute optimum value. However, an exact solution in such

problems is not as important as solving the problem to establish the viability of the routing scheme to be used. For instance, our experiments reveal that, on our system, it is not possible to handle a 30 node network using CPLEX. When developing the routing scheme for a network with 30 or more nodes, for all practical purposes, our design is finished if

- we can develop a near-optimal routing scheme where we have the exact value of the congestion using our scheme
- we know that our scheme gives a congestion within, say, 1% of the optimum value,
- the value of the congestion is less than the capacity of a lightpath.

In such a situation, we know that our design is viable even though it does not give the actual minimum congestion. This is the approach we will explore in this thesis.

In the operations research community, approximate algorithms have been proposed to solve network flow problems in large networks. Garg and Konemann[GK98] give a simple, deterministic algorithm to solve the maximum flow problem. In a way similar to that in [Y95], this algorithm augments flow in the network using shortest paths. They obtained an improvement in run time for the flow problems. Their main contribution is to provide an analysis for the correctness of their algorithms. Lisa K.Fleischer [F99] has proposed a faster approximation algorithm for maximum multi-commodity concurrent flow problem based on Garg and Konemann[GK98]. Fleischer gave the first approximation algorithm with a run time that is independent of the number of

commodities. It is faster than Garg and Konemann's algorithm when the graph is sparse or there are a large number of commodities.

Our approach is based on a simplified version of Fleischer's algorithm. We have developed a fast approximation algorithm for to minimize the congestion in the logical topology of WDM wavelength routed network.

1.4 Thesis outline

In this thesis, we have presented three variations of our scheme for obtaining an approximate solution for the minimum congestion problem of the wavelength-routed WDM networks and have carried out experiments comparing our algorithms with the standard solution obtained using the CPLEX LP solver.

In chapter 2 we have given a background review on WDM networks, multi-commodity flow problems, and shortest path algorithms. In chapter 3 we have described our algorithms in detail. In chapter 4 we have described how we carried out our experiments and the experimental results. Chapter 5 concludes the thesis with some critical summary and suggestions for future work.

2. Background Review

In this chapter, we review some relevant background material including WDM Networks, basic concepts of multi-commodity network flows, and existing linear program-based solutions for routing in WDM networks.

2.1 Overview of WDM Networks

Three generations of networks based on the physical technology have been developed [G91] [SMM01]. First generation networks are based on copper-wire or microwave-radio technology. Second generation networks use optical fibers in traditional architectures, in which the fiber is used simply as a replacement for copper. Because of the advantages of fiber, the second-generation networks can achieve performance improvements over first-generation networks, such as, higher data rates, lower error rates, and reduced electro-magnetic radiation (emissions from the cabling). However, the performance of second generation networks is limited by the maximum speed of electronics (a few gigabits per second) employed in the switches and end-nodes. In third generation networks, fiber is used for its unique properties in order to meet the needs of high bandwidth applications. To overcome the shortcomings of the first and second generation networks, totally new approaches are employed to exploit the vast bandwidth (50 tera bits/second which may be divided to give up to 200 channels) available in fibers. The third generation networks are designed as all-optical in nature. That is, information is transferred in the optical domain

(without facing any electro-optical conversions) through the network until it is delivered to its final destination.

2.1.1 Fiber overview

An optical fiber has bandwidth capacities on the order of 50 Tbps (50 terabits per second) and extremely low bit error rates[SMM01]. With these advantages, optical fiber has become the obvious choice as a medium of transmission for future networks.

The cross-sectional diagram in figure 2.1 shows the three main components of a fiber [B93]. The center of the fiber is a glass filled medium called the core. The core is responsible for guiding light down the line. The next layer above the core is the cladding. The cladding is also made up of glass and has a refractive index coefficient which is slightly lower than that of the core. To protect these two glass components, a protective layer called a buffer surrounds the cladding interface.

Transportation of information down a fiber requires an optical transmitter interfaced to one end of the line with an optical receiver at the other end. An example of this configuration is shown in figure 2.2. In this diagram, an electrical signal modulates an optical signal which has a predetermined carrier wavelength. The modulated optical signals are injected into the core of the fiber. The light signals then propagate down the length of the fiber through reflections at the core-cladding interface. At the end of the line, the optical signals are received and converted back to electrical signals.

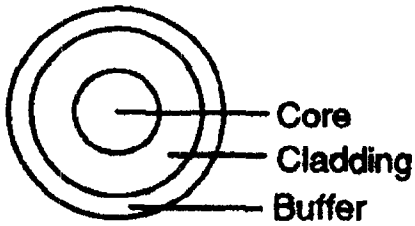


Figure 2.1. Fiber Cross Section

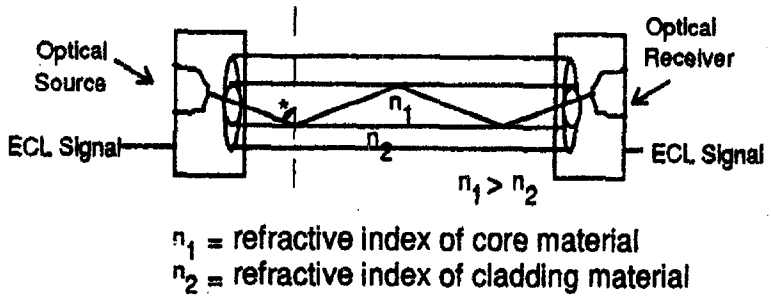


Figure 2.2. Fiber Optical Transmission

As long as the angle of incident light into the fiber is greater than the critical angle (marked as * in Figure 2.2), total internal reflection will occur. Total internal refection implies that no light escapes the core material as it travels down the length of the fiber.

2.1.2 Lightpath

There are multiple communication channels at different wavelengths (corresponding to carrier frequencies) in a single optical fiber. Lightpaths are end-to-end circuits switched

communication connections that traverse one or more links and use one WDM channel per link [RS97]. Lightpaths serve as the physical communication links for a variety of high-speed networks such as asynchronous transfer mode (ATM) networks.

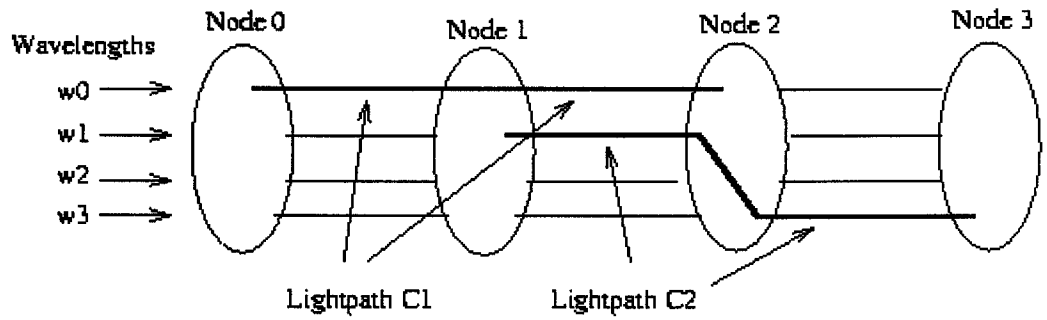


Figure 2.3. A network with wavelengths $\{\omega_0, \omega_1, \omega_2, \omega_3\}$. Channels are shown as lines between nodes

An example of a WDM wavelength routed network is shown in Figure 2.3. It is composed of four nodes with three links connecting nodes 0 and 1, nodes 1 and 2, and nodes 2 and 3. Each link has four WDM channels at wavelengths $\{\omega_0, \omega_1, \omega_2, \omega_3\}$. There are two established lightpaths: C_1 from node 0 to node 2 and C_2 from node 1 to node 3. If a lightpath uses different wavelengths on different fibers along its path, then wavelength conversion devices are needed. For example, lightpath C_2 needs a converter at node 2 because it uses channel ω_1 from node 1 to node 2 and ω_2 from node 2 to node 3. However, the lightpath C_1 uses the same wavelength (ω_0) along its entire path. Hence, it does not need a wavelength converter. The advantage of wavelength conversion is that

WDM channels will be used more efficiently, but the disadvantage is increased cost and complexity.

2.1.3 WDM networks

Concurrency among multiple user transmissions is needed to make use of the vast bandwidth available without experiencing any electronics bottleneck. WDM increases the information capacity of a fiber by transmitting information on several different wavelengths simultaneously. By using multiplexers (MUX) and de-multiplexers (DMUX), the system is able to achieve this simultaneous transmission without experiencing significant interference between the information channels[GR00].

[CGK92] first proposed an architectural approach based on light paths on WDM networks. Before this paper, existing network architectures could not efficiently utilize the light fiber bandwidth in the wide area networks.

Figure 2.4 shows a WDM all-optical network employing wavelength routing, consisting of optical routing nodes interconnected by optical links[RS95]. Each link is assumed to be bi-directional and actually consists of a pair of unidirectional links. An optical routing node is capable of routing each wavelength on an incoming link to any outgoing link. Nodes 1, 2, 3, 4, 5 in Figure 2.4 are router nodes. However, the same wavelength on two incoming links cannot be routed simultaneously onto a single outgoing link.

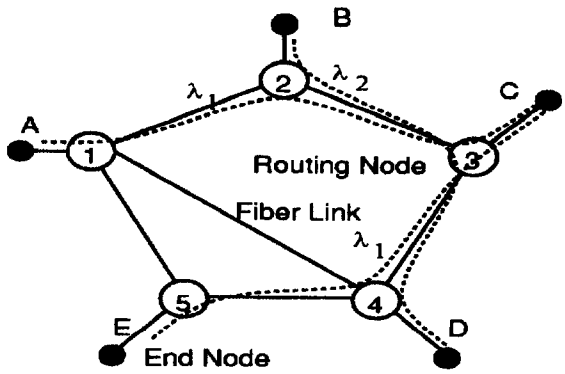


Figure 2.4: A WDM network with some lightpaths

A WDM network consists of routing nodes, interconnected by point-to-point fiber-optical links. In the network shown in Figure 2.4, each connection (or lightpath) must be assigned a specific path in the network and a wavelength. The wavelength assigned must be such that no other lightpath on any path that shares a common fiber, is assigned the same wavelength. For example, in Figure 2.4, lightpath L_1, between node A and node C (A $\rightarrow$ 1 $\rightarrow$ 2 $\rightarrow$ 3 $\rightarrow$ C), is assigned wavelength λ_1. Therefore, lightpath L_2, between node B and node C (B $\rightarrow$ 2 $\rightarrow$ 3 $\rightarrow$ C), must be assigned a different wavelength λ_2, since it shares links 2→3 and 3→C with lightpath L_1.

2.1.4 Classification of WDM network

Optical networks can be classified either as single-hop [M92-1] or as multi-hop [M92-2] according to the number of hops that the data traverses until it reaches its destination node.

Single-hop networks:

In single-hop networks, there is no intermediate node where an electro-optical conversion is required. In other words, the data from source *s* to destination *d* uses only a single lightpath. The lightpath itself may traverse several fibers and several router nodes. However, this is done completely in the optical domain. As a result, a significant amount of dynamic coordination between nodes is required. For a packet transmission to occur, one of the transmitters of the sending node and one of the receivers of the destination node must be tuned to the same wavelength for the duration of the packet's transmission.

Multi-hop networks:

In multi-hop networks, information from a source to a destination may have to hop through zero or more intermediate nodes [M92-2]. At each intermediate node, the data is converted to the electronic domain, analysed and routed to the different appropriate output link, converted back into the optical domain and transmitted over the correct fiber. The wavelength to which a node's transmitter or receiver is to be is rarely changed. The intermediate nodes are responsible for routing data among wavelengths such that the data sent out on one of the sender's transmission wavelengths finally gets to the destination on one of the destination's receiving wavelengths, after "multi-hopping" through a number of intermediate nodes.

2.1.5 Physical and logical topology

Figure 2.5a is an example of the physical connection of a WDM network. A, B, C, and D are four end nodes, and nodes 1, 2, 3, 4 are router nodes. The physical topology of the network consists of the physical set of routing/end-nodes and the fiber-optic links

connecting them. These links are used to set up lightpaths between end nodes. In this example, the physical topology of the network consists of optical routers interconnected by pairs of point-to-point fiber links in an arbitrary mesh topology as shown in Fig. 2.5a [RSK96]. Each pair of links is represented by an undirected edge between routing nodes in this figure. End-nodes are attached to the routers. Each end node has a limited number of optical transmitters and receivers. Each link is capable of carrying a certain number of wavelengths. A routing node takes in a signal at a given wavelength at one of its inputs and routes it to a particular output.

The set of all lightpaths that have been set up between end nodes defines the logical topology. Depending on the set of established lightpaths, there may be many different logical topologies that can be supported over a given physical topology. Fig 2.5c shows a number of lightpaths set up over the physical topology of fig. 2.5a. Fig. 2.5b shows the logical topology corresponding to this set of lightpaths. The logical topology is a graph with the nodes corresponding to the end-nodes in the network with a directed edge from node A to node B if a lighpath has been set up from node A to node B.

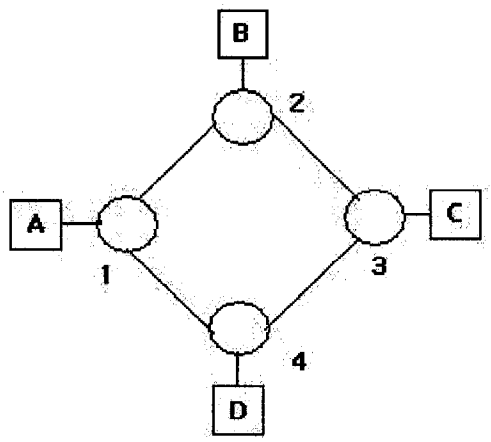


Figure 2.5a. Physical network

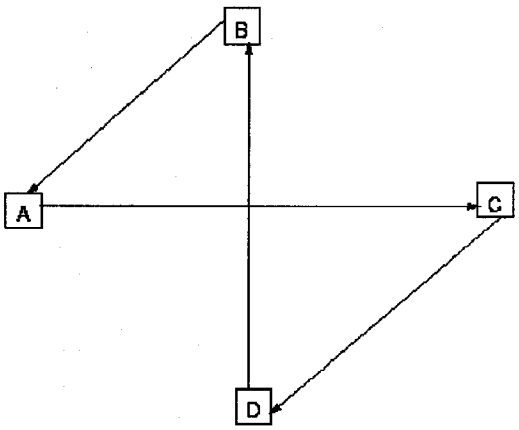


Figure 2.5b. Logical Topology

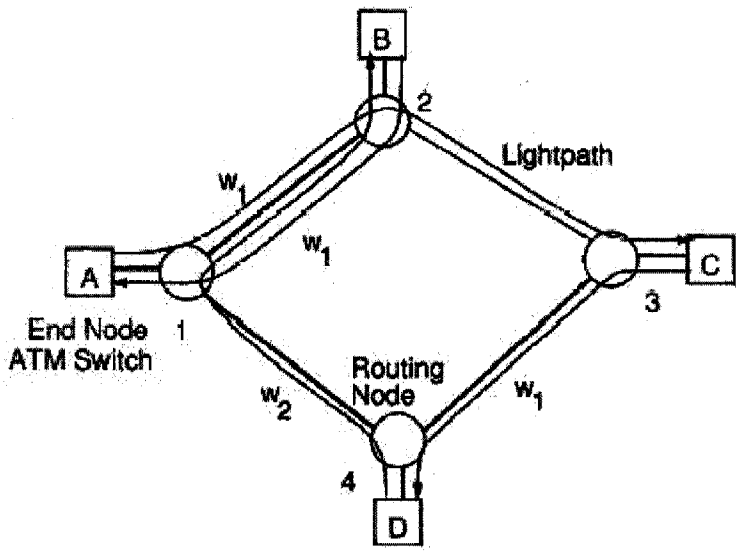


Figure 2.5c. Physical Topology

Ideally in a network with N nodes, we would like to set up lightpaths between all the $N(N-1)$ node pairs. However this is usually not possible because of the number of wavelengths available imposes a limit on how many lightpaths can be set up.

2.2 Linear Programming

Linear Programming, a specific class of mathematical techniques to solve problems in which a linear function is maximized (or minimized) subject to given linear constraints[V96].

The general form of a linear program is

$$\begin{aligned} \text{Maximize} \quad & c_1 x_1 + \ldots + c_n x_n \\ \text{Subject to} \quad & a_{11} x_1 + \ldots + c_n x_n \le b_1, \\ & a_{m1} x_1 + \ldots + a_{mn} x_n \le b_m, \\ & x_1 \ge 0, \ldots, x_n \ge 0. \end{aligned}$$

Here $c_1, \ldots, c_n$, $b_1, \ldots, b_n$ and $a_{11}, \ldots, a_{mn}$ are given constants, and $x_1, \ldots, x_n$ are variables whose values are to be determined, maximizing the given objective subject to the given constraints. There are n variables and m constraints, in addition to the nonnegativity restrictions on the variables. The constraints are called linear because they involve only linear functions of the variables. Quadratic terms such as x_1^2 or $x_1 x_2$ are not permitted. If minimization is desired instead of maximization, this can be accomplished by reversing the signs of $c_1, \ldots, c_n$.

An important mathematical property of linear programs is the theory of duality[W97]. The *dual* of the linear program given above is

$$\text{Minimize} \qquad b_1 y_1 + \ldots + b_m y_m$$

$$\text{Subject to} \qquad a_{11} y_1 + \ldots + a_{m1} y_m \geq c_1,$$

$$a_{1n} y_1 + \ldots + a_{mn} y_n \geq c_n,$$

$$y_1 \geq 0, \ldots, y_m \geq 0.$$

This is a linear program in the variables $y_1, \ldots, y_m$. It is not hard to show that if $(x_1, \ldots, x_m)$ is in the feasible region for the first linear program and $(y_1, \ldots, y_m)$ is in the feasible region for the dual linear program, than the first objective function $c_1 x_1 + \ldots + c_n x_n$ is less than or equal to the dual objective function $b_1 y_1 + \ldots + b_m y_m$. The remarkable fact is that the optimal values of these two objectives are always the same. This is of great practical importance for both the interpretation of the solutions of linear programs and the methods for calculating their solutions.

2.3 Single Commodity Network Flows

Network flow problems occur in many areas including applied mathematics, computer science, engineering, management, and operations research [AMO93], [ABF84]. For example, telephone networks permit us to communicate with each other within our local communities and across regional and international borders. National highway systems, rail networks, and airline service networks provides us with the means to cross great

geographical distances to accomplish our work. In all of these problems, and in many more, we wish to move some entity (electricity, a consumer product, a person or a vehicle, a message) from one point to another in an underlying network, and to do so as efficiently as possible, both to provide good service to the users of the network and to use the underlying (and typically expensive) transmission facilities effectively. Such an entity is called a *commodity*. In this section we consider the case where there is only one commodity to be moved from its source to its destination.

There are two basic types of network flow problems.

i) **Maximum flow problem:** If a network has capacity constraints on each arc – indicating the maximum amount of commodity that may be carried by each edge, how can we send the maximum quantity of the commodity from its source to its destination in the network without violating the capacity constraints of the arcs?

ii) **Minimum cost flow problem**: If a network has capacity constraints on each arc, and, for each arc, there is a specified cost per unit quantity of the commodity for using that arc, how can we send a specified amount of the commodity at the minimum possible cost?

In the sense of traditional applied and pure mathematics, these problems are not difficult to solve. Since we need only consider a finite number of alternatives for each problem, simply enumerating the set of possible solutions and choosing the best solution easily solve the problems. Unfortunately, this approach is far from pragmatic, since the number of possible alternatives can be very large. So instead, we would like to devise algorithms

that are in a sense "good," that is, whose computation time is small, or at least reasonable, for problems met in practice. One way to ensure this objective is to devise algorithms whose running time is guaranteed not to grow very fast as the underlying network becomes larger.

2.3.1 Maximum Flow algorithms

In the maximum flow problem in a capacitated network, we wish to send as much flow (in other words, as much quantity of the commodity) as possible between the source node s and the destination node t of the commodity, without exceeding the capacity of any arc. In this section, we review four well-known algorithms for solving maximum flow problems.

2.3.1.1 Generic Augmenting Path algorithm

The Generic Augmenting Path Algorithm [AMO93] is one of the simplest and most intuitive algorithms for solving the maximum flow problem. It uses the concept of *residual networks*. In a residual network, the labels on the edges represent the available capacity on that edge. Initially (when there is zero flow over the network), the labels indicate the actual edge capacities. As flows are sent over certain edges, the labels of these edges must be modified to reflect the remaining (unused) capacity of the edge.

We refer to a directed path from the source to the destination in the residual network (the network with zero flow) as an augmenting path. The generic augmenting path algorithm proceeds by identifying augmenting paths and augmenting flows on these paths until the network contains no such path. An outline of the generic augmenting path algorithm is given below.

Begin

x:=0;

while *G(x)* contains a directed path from node *s* to node *t* do

begin

identify an augmenting path *p* from node *s* to node *t*;

$\delta := \min\{r_{ij}: (i, j) \in p\}$;

augment δ units of flow along *p* and update *G(x);*

end;

end;

To illustrate the above algorithm, we use the network given in figure 2.6. The objective is to send the maximum possible flow from node 1 to node 4. Fig. 2.6a shows the initial residual network and edge capacities, before any flow has been sent. Suppose that the algorithm selects the path 1-3-4 for augmentation. The residual capacity of this path is $\delta = \min\{r_{13}, r_{34}\} = \min\{4, 5\} = 4$. This augmentation reduces the residual capacity of arc (1, 3) to zero (thus we delete it from the residual network) and increases the residual capacity of arc (3, 1) to 4 (so we add this arc to the residual network). The augmentation also decreases the residual capacity of arc (3, 4) from 5 to 1 and increases the residual capacity of arc (4, 3) from 0, to 4. Figure 2.6(b) shows the residual network at this stage. In the second iteration, suppose that the algorithm selects the path 1-2-3-4. The residual capacity of this path is $\delta = \min\{2, 3, 1\} = 1$. Augmenting 1 unit of flow along this path

yields the residual network shown in figure 2.6(c). In the third iteration, the algorithm augments 1 unit of flow along the path 1-2-4. Figure 2.6(d) shows the corresponding residual network. Now the residual network contains no augmenting path, so the algorithm terminates.

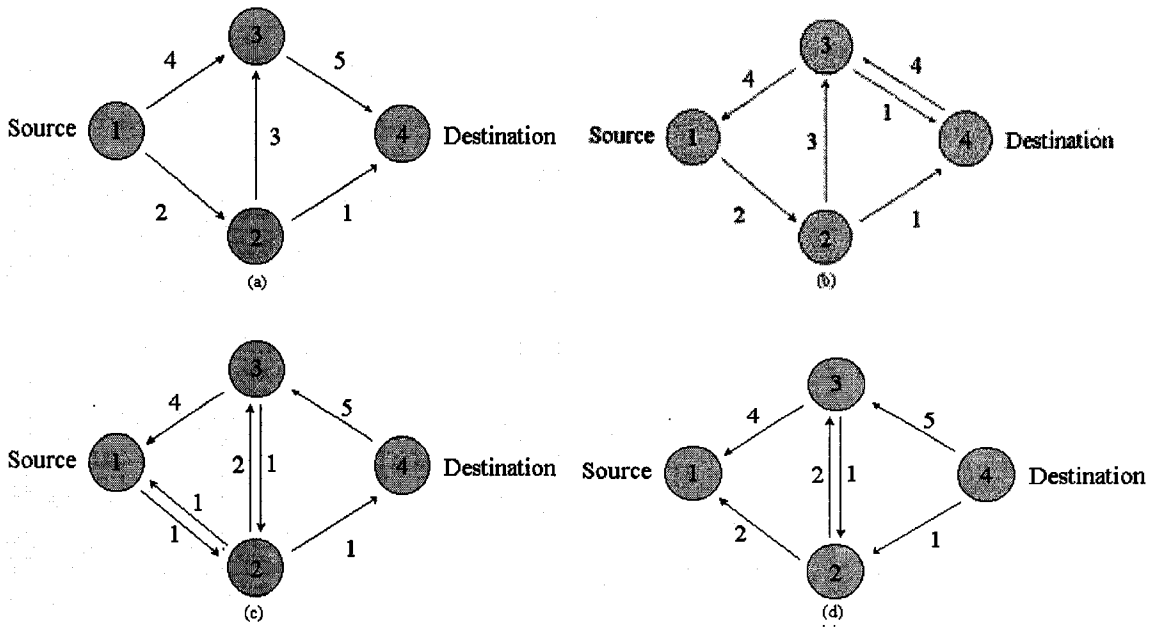


Figure 2.6 (a) initial residual network with zero flow; (b) network after augmenting four units along the path 1-3-4; (c) network after augmenting one unit along the path 1-2-3-4; (d) network after augmenting one unit along the path 1-2-4.

2.3.1.2 Labeling algorithm and the max-flow min-cut theorem

Labeling algorithm [AMO93] is an implementation of the augmenting path algorithm that specifies

(i) how to identify an augmenting path or to show that the network contains no such path, and

(ii) whether the algorithm terminates in finite number of iterations, and when it terminates, whether it has obtained a maximum flow.

The labelling algorithm uses a search technique to identify a directed path in $G(x)$ (G is a network with flow x through the path) from the source to the destination. The algorithm starts from the source node to find all nodes that are reachable from the source along a directed path in the residual network (the network with zero flow). At any step, the algorithm has partitioned the nodes in the network into two groups: labelled and unlabeled. Labelled nodes are those nodes that the algorithm has reached in the search process and so the residual network; the unlabeled nodes are those nodes that the search process of the algorithm has not yet reached. The algorithm iteratively selects a labelled node and scans its arc adjacency list to reach and label additional nodes. Eventually, the destination becomes labelled and the algorithm sends the maximum possible flow on the path from node s to node t. It then erases the labels and repeats this process. The algorithm terminates when it has scanned all the labelled nodes and the sink remains unlabeled, implying that the source node is not connected to the sink node in the residual network.

A flow x^* is a maximum flow if and only if the residual network $G(x^*)$ contains no augmenting path. Labelling algorithm uses generic augment in path algorithm to find max flow. We establish it by specifying a labelling algorithm that maintains a feasible flow x in the network and sends additional flow along directed paths from the source node to the destination node in the residual network $G(x)$. Eventually, $G(x)$ contains no directed path from the source to the sink. At this point, the value of the flow x is the max-flow.

2.3.1.3 Shortest augmenting path algorithm

The shortest augmenting path algorithm [AMO93] always augments the flow along a shortest path from the source to the destination in the residual network. A natural approach for implementing this approach would be to look for the shortest path by performing a breadth first search in the residual network. The shortest augmenting path algorithm proceeds by augmenting flows along admissible paths. It constructs an admissible path incrementally by adding one arc at a time. The algorithm maintains a partially admissible path (i.e., a path from s to some node i consisting solely of admissible arcs) and iteratively performs advance or retreat operations from the last node of the partial admissible path, which we refer to as the current node. If the current node i has an admissible arc (i, j), the algorithm performs an advance operation and add arc(i, j) to the partial admissible path; otherwise, it performs a retreat operation and backtracks one arc. These operations are repeated until the partial admissible path reaches the

destination node at which time the algorithm performs an augmentation. The process is repeated until the flow is maximized.

Figure 2.7 shows an example of the shortest augmenting path algorithm, trying to send the maximum flow from node 1 to node 8. The algorithm first computes the initial distance labels by performing the backward breadth-first search of the residual network starting at the destination node. The numbers beside the nodes in Figure 2.7 specify these values of the distance labels. It adopts the convention of selecting the arc(i, j) with the smallest value of j whenever node i has several admissible arcs.

Iteration 1: The algorithm starts at the source node with a null partial admissible path. The source node has several admissible arcs, so it performs an advance operation. This operation adds the arc (1, 2) to the partial admissible path. It stores this path using predecessor indices, so it set pred(2) = 1. Now node 2 is the current node and the algorithm performs an advance operation at node 2. In doing so, it adds arc (2,5) to the partial admissible path, which now becomes 1-2-5. It also set pred(5) = 2. In the next iteration, the algorithm adds arc (5, 8) to the partial admissible path obtaining 1-2-5-8, which is an admissible path to the destination node. It performs an augmentation of value $\min\{r_{12}, r_{25}, r_{5,8}\} = \min\{2,1,2\} = 1$, and thus saturates the arc(2, 5). Figure 2.7(b) specifies the residual network at this stage.

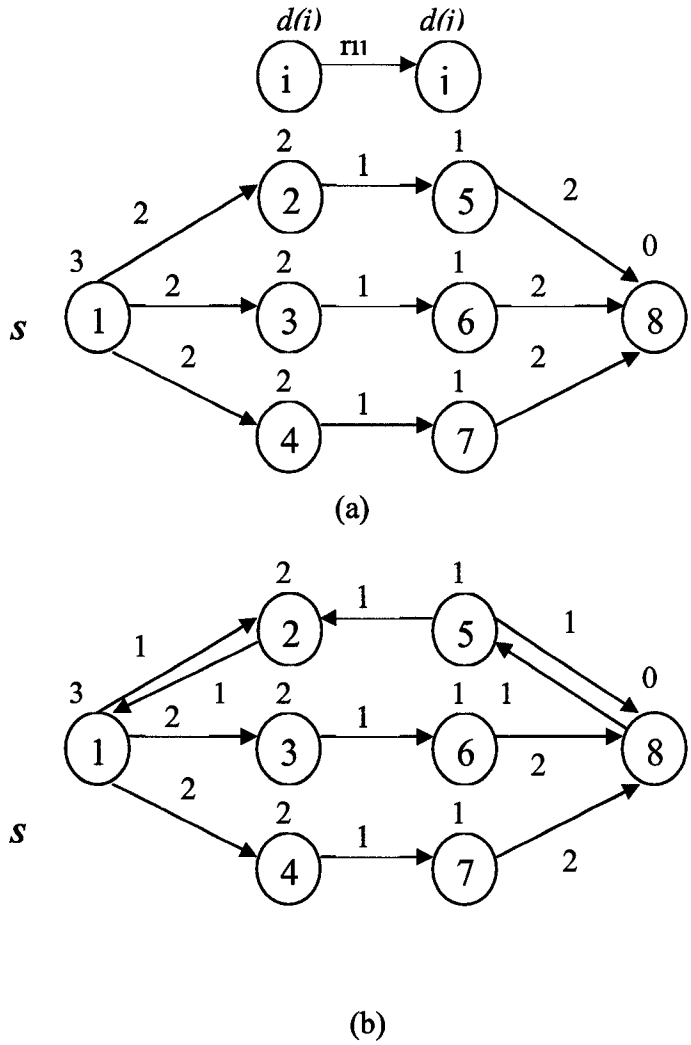


Figure 2.7 The shortest augmenting path algorithm

Iteration 2: The algorithm again starts at the source node with a null partial admissible path. It then adds the arc (1, 2) and node 2 becomes the new current node. Now it finds that node 2 has no admissible arc. Thus it deletes arc (1,2) from the partial admissible path which again becomes a null path. In the subsequent operations, the algorithm identifies the admissible paths 1-3-6-8, 1-4-7-8 and augments unit flows on these paths.

2.4 Multi-commodity Network Flows

The algorithms discussed in section 2.3 assume that there is a single commodity to be sent from some source *s* to a destination *d*. The multi-commodity network flow problem is defined over a network where more than one commodity needs to be shipped from their respective sources to their destinations without violating the capacity constraints associated with each arcs[AL94]. MCNF models arise in many real-world applications. Most of them involve network routing and design problems. For example, traffic routing between different source-destination pairs in a WDM network can be viewed as a multi-commodity flow problem. In a communication network, nodes represent the sources and the destinations for messages, and arcs represent transmission lines. Messages between different pairs of nodes define distinct commodities; the supply and demand for each commodity is the amount of data to be sent between the origin and destination nodes of that commodity. Each transmission line has a fixed capacity. The problem of determining the scheme to get the required flow of all messages at a minimum cost is an important multi-commodity flow problem.

Multi-commodity network flow problems extends the single commodity network flow problem[W03]. If we disregard the bundle constraints - the arc capacity constraints that tie together flows of different commodities passing through the same arc, a multi-commodity network flow problem can be viewed as several independent single commodity network flow problems [W03]. Informally the multi-commodity network flow [SM90] can be stated as follows: Let there be a network of entities (cities,

neighborhoods, computers, etc.) in which there exists traffic or flow (passengers, messages, etc.) between all pairs of entities. The flow is sustained through channels (links, communication lines etc.) which are capacitated [W03][K02]. The algorithm seeks to assign flows in a manner such that our requirements are satisfied for an optimal value of some objective function.

In general, there are two major multi-commodity network flow problems in the literature [AMO93]:

- the maximum concurrent flow problem,
- the minimum cost multi-commodity network flow problem.

The maximum multi-commodity network flow problem is to maximize the sum of flows for all commodities between their respective origins and destinations. The minimum cost MCNF problem is to find the flow assignment satisfying the demands of all commodities with minimum cost without violating the capacity constraints on all arcs.

The existence of the bundle constraints makes MCNF problems much more difficult than single commodity network flow (SCNF) problems. For example, many SCNF algorithms exploit the single commodity flow property, in which flows in opposite directions on an arc, could be canceled out. In MCNF problems, flows for different commodities do not cancel each other.

A multi-commodity flow problem is defined on a directed network $G = (V, E)$ with capacities u: $E \rightarrow R$ and K commodities. The j^{th} commodity has an associated source s_j

and a destination t_j, $1 \leq j \leq K$. The problem is to find flows for the j^{th} commodity, from s_j to t_j, that satisfy node conservation constrains and meet some objective function criteria so that the sum of flows on any edge does not exceed the capacity of the edge, for all j, $1 \leq j \leq K$.

Let P_j denote the set of paths from s_j to t_j, and let $P = \bigcup_j P_j$. Variable $x(p)$ equals the amount of flow sent along path p. For the maximum multi-commodity flow problem, the objective is to maximize the sum of the flows. The corresponding linear programming formulation is then

$$\text{Max} \sum_{p \in P} x(p)$$

$$\forall e : \sum_{p : e \in P} x(p) \leq u(e)$$

$$\forall p : x(p) \geq 0$$

The dual to this linear program [GK98] corresponds to the problem of assigning lengths to the edges of the graph so that the length of the shortest path from s_j to t_j is at least 1 for all commodities j. The length of an edge represents the marginal cost of using an additional unit of capacity of the edge.

$$\text{Min} \sum_{e} u(e) l(e)$$

$$\forall p : \sum_{e \in P} l(e) \geq 1$$

$$\forall e : l(e) \geq 0$$

For the maximum concurrent flow problem, there are demands d_j associated with each commodity j, and the objective is to satisfy the maximum possible proportion of all demands. We will discuss such formulations in more detail in chapter 3.

2.4.1 Approximation algorithms

The number of variables and constraints in the MCNF formulations given above grow very rapidly with network size, particularly when the number of commodities is large. Therefore, it is very difficult to obtain optimal solutions, even for networks of moderate size. In this context, approximation algorithms can be an important tool for obtaining "good" solutions, in a reasonable amount of time. These algorithms do not necessarily generate the best solution, but can guarantee that their solution will be within a specified bound of the optimal solution. In this section we will briefly discuss an existing approximation algorithm for solving the generic MCNF problem as given in [F99] and [GK98]. The problem is to find a flow that maximizes the ratio of the demands. In chapter 3 we will present our own algorithm for solving the optimal routing problem in WDM networks.

Let $x(p)$ denote the flow quantity on path p. The maximum flow problem can be formulated as the following linear program:

$$\text{Max } \lambda$$

$$\forall e : \sum_{p:e\in P} x(p) \le u(e)$$

$$\forall j : \sum_{p \in P_j} x(p) \geq \lambda d_j$$

$$\forall p : x(p) \geq 0$$

The corresponding dual LP problem is to assign lengths to the edges of the graph, and weights z_j to the commodities so that the length of the shortest path from s_j to t_j is at least z_j for all commodities j, and the sum of the product of commodity weights and demands if at least 1[GK98]. The length of an edge represents the marginal cost of using an additional unit of capacity of the edge, and the weight of a commodity represents the marginal cost of not satisfying another unit of demand of the commodity.

$$\text{Min} \sum_{e} u(e) l(e)$$

$$\forall j, \forall p \in W_j : \sum_{e \in P} l(p) \geq 1$$

$$\sum_{1 \leq j \leq k} d_j z_j \geq 1$$

$$\forall e : l(e) \geq 0$$

$$\forall j : z_j \geq 0$$

The papers [F99][GK98] introduce the following approximation algorithm. Initially, $l(e) = \delta/u(e)$, $z_j = \min_{P \in W_j} l(p), x \equiv 0$. The algorithm proceeds in phases. In each phase, there are K iterations. In iteration j, the objective is to route d_j units of flow from s_j to t_j. This is done in steps. In one step, the shortest path P from s_j to t_j is computed using the current length function. Let u be the bottleneck capacity of this path. Then the minimum of u and the remaining demand is sent along this path. The dual variables l are updated as before, and z_j is set equal to the length of the new minimum length path from s_j to t_j. The entire

procedure stops when the dual objective function value is at least one, i.e. $D(l):=\sum_e u(e)l(e) \geq 1$. The following is a summary of the algorithm.

Input: network G, capacities u(e), vertex pairs (s_j, t_j) with demands d_i, $1 \leq j \leq k$, accuracy ε

Output: primal x and dual solution l

Initialize $l(e) = \delta/u(e)$ $\forall e, x \equiv 0$.

while $D(l) < 1$

for $j = 1$ to k do

$d_j' = d_j$

while $D(l) < 1$ and $d_j' > 0$

p = shortest path in P_j using l

$u = \min\{d_j', \min_{e \in P} u(e)\}$

$d_j' = d_j' - u$

x(p) = x(p) +u

$\forall e \in P, l(e) = l(e)(1 + \varepsilon u / u(e))$

end while

end while

return (x, l).

2.4 Overview of shortest path algorithms

One important operation in the approximation algorithms is to find the shortest path for a particular commodity. This operation may have to be repeated many times for each

commodity. Therefore, it is extremely important to find the shortest path as efficiently as possible. In this section we review some well-known shortest path algorithms that we have used in our implementations.

We can group shortest path algorithms into 2 classes:

- those that employ combinatorial or network traversal techniques such as label-setting methods, label-correcting methods, Dijkstra algorithm [AMO93] [AMT90].

- those that use algebraic or matrix techniques such as Floyd-Warshall [F62][W62] [AMT90] algorithms.

Dijkstra's algorithm requires the edge weights of a graph $G(V, E)$ to be positive. It's complexity is $O(|V|^2)$. Floyd's algorithm allows negative weights as well, but requires the digraph to be free of any cycle whose total edge weight is negative (a so-called negative cycle). The algorithm finds shortest paths between every pair of vertices in G and detects negative cycles if they occur. It also provides a compact matrix representation for the $|V|^2$ shortest paths found. The time performance of the algorithm is $O(|V|^3)$.

2.4.1 Dijkstra algorithm

Dijkstra's algorithm [AMO93] [AMT90], solves the problem of finding the shortest path from a node in a graph (the source s) to a destination. It turns out that one can find the

shortest paths from a given source to all points in a graph in the same time, hence this problem is called the Single-source shortest paths problem. We consider a graph, $G=(V,E)$ where V is a set of nodes and E is a set of edges. For an edge $u \rightarrow v$ from node u to node v, $l(u \rightarrow v)$ represents the length of edge. Dijkstra's algorithm keeps two sets of nodes:

- S = set of permanently labeled nodes whose shortest paths from the source node s have already been determined and
- $T = V - S$ (the remaining temporarily labeled nodes).

It also defines the following data structures:

- $d[w]$ = current best estimate of shortest distance from s to node w
- $p[w]$ = current value of predecessors for node w

The basic algorithm is given below:

1. Initialise

   a. $d[w] = \infty$, $\forall w \neq s$ and $d[s] = 0$

   b. $p[w] = null$

   c. $S = \{\}$

   d. $T = V$

2. While there are still vertices in T,

   a. Sort the vertices in T according to current best estimate of their distance from s

   b. Add u, the closest vertex in T, to S, and remove it from T

   c. Relax all the vertices v (still in T) that are adjacent to u as follows:

If $d[v] > d[u]+l(u \rightarrow v)$

{

$d[v] = d[u]+l(u \rightarrow v)$

$p[v] = u$

}

The distance label to any permanently labeled node represents the shortest distance from the source to that node. For any temporary node, the distance label is an upper bound on the shortest path distance to that node. The basic idea of the algorithm is to fan out from node s and permanently label nodes in the order of their distances from node s. Dijkstra's algorithm maintains a directed out-tree S_{tree} rooted at the source that spans the nodes with finite distance labels. The algorithm maintains this tree using predecessor indices [i.e., if $(u,v) \in S_{tree}$, then pred(v) = u]. At termination, when distance labels represent shortest path distances, T is a shortest path tree.

2.4.2 Floyd-Warshall algorithm

Floyd-Warshall algorithm [F62][W62] [AMT90] maintains a list, LIST, of all arcs that might violate their optimality conditions. If LIST is empty, an optimal solution is found. Otherwise, the algorithm examines this list to select an arc, say (i, j), violating its optimality condition. The arc (i, j) is removed from LIST, and if this arc violates its optimality condition the algorithm uses it to update the distance label of node j. Notice that any decrease in the distance label of node j decreases the reduced lengths of all arcs

emanating from node j and some of these arcs might violate the optimality condition. Also notice that decreasing $d(j)$ maintains the optimality condition for all incoming arcs at node j. Therefore, if $d(j)$ decreases, the arcs in $A(j)$ are added to the set LIST. Next, observe that whenever arcs are added to LIST, the algorithm adds all arcs emanating from a single node (whose distance label decreases). This suggests that instead of maintaining a list of all arcs that might violate their optimality conditions, the algorithm maintains a list of nodes with the property that if an arc(i, j) violates the optimality condition, LIST must contain node i, Maintaining a node list rather than the arc list requires less work and leads to faster algorithms in practice.

3. Approximation algorithms for WDM network

In this chapter, we will present our approach, based on the concept of approximation algorithms[F99], for solving the multi-commodity flow problem to minimize the congestion in WDM optical networks. Each lightpath in a WDM network constitutes a commodity from the multi-commodity network flow perspective. The number of lightpaths is $O(n^2)$, in a network with n nodes, since most node pairs will have some traffic between them. This leads to an important aspect of the congestion problem in WDM networks - the routing in WDM network has to deal with a large number of commodities. Multi-commodity flow problems considered in the literature consider a limited number of commodities [AMO93]. The fact that number of commodities in WDM networks is $O(n^2)$ means that the standard LP solving approach is not practical for practical sized networks. For instance, in a 30 node network where the node degree is 3 on an average, the number of edges is 90 and the number of commodities is almost 900. We will see in the next section that the basis size for a node-arc formulation for this problem is very large. In other words, standard LP guarantees an optimal solution, but cannot handle even moderate size networks. One approach is to use some heuristics. Heuristics can give us fast solutions, but we are not aware of any heuristic that provide a guarantee on the quality of the solution.

In this chapter we will introduce our approach, based on approximation algorithms, which can generate near optimal results, with a known error bound, in an acceptably short time.

To give us a basis for comparison, we first present a straightforward implementation of the routing problem, based on the node-arc LP formulation, that could be solved using a solver such as the CPLEX. Then we will discuss why this approach is not feasible for moderate sized networks, let alone large networks. We will then introduce an approximation algorithm for minimizing congestion in WDM networks. We will also discuss the accuracy of the algorithm as well as the stopping criteria. We have implemented three different versions of the algorithm, based on how the shortest paths for each commodity are calculated. We will present the performance comparisons for these implementations in the next chapter.

3.1 Node Arc Formulation

In this section, we present the node-arc formulation for routing in WDM networks to get the minimum congestion. We view the logical topology of a WDM network as a graph $G = (N, E)$, where N is the set of n end-nodes in the WDM network and E is the set of m edges, each edge $x \rightarrow y$ representing a lightpath from end-node x to end-node y. The objective of this LP is to minimize the congestion μ of the network, which is defined as the maximum traffic flow on any edge $e \in E$. We define $f_{e,j}$ as the traffic flow on edge e,

corresponding to the j^{th} source-destination pair (s_j, t_j). Then $\sum_j f_{e,j}$ is the total traffic flow on edge e. We assume that d_j units of traffic must be sent over the network, from s_j to t_j $1 \le j \le K$, where $K = n(n-1)$. The node-arc formulation can now be stated as follows.

Minimize μ

Such that:

$f_{e,j} - d_j \le 0 \quad \forall e \in E$ and $j = 1,2,3.....K.$ (1)

Constraint (1) is used to define the traffic flows between each node pair. The amount of traffic flowing on edge e, corresponding to the j^{th} source-destination pair must be less than or equal to the total amount of traffic between node pair (s_j, t_j).

$$\sum_{e=p\to q} f_{ej} - \sum_{e=q\to p} f_{ej} = \begin{cases} d_j, \text{ if } p = s_j \\ -d_j, \text{ if } p = t_j \\ 0, \text{ otherwise} \end{cases} \qquad \forall p \in N, \quad j = 1,2,3...K. \qquad (2)$$

Constraint (2) is a set of flow conservation constraints, which are used to route the traffic over the lightpaths between the node pair (s_j, t_j). For any node, if $p = s_j$ is source node, the total traffic amount on all lightpaths between node pair (s_j, t_j) originating from the source node p will be d_j. If $p = t_j$, p is destination node, the total traffic amount on all lightpaths between node pair (s_j, t_j) terminating at destination node will be d_j. If p is an intermediate node, then the total amount traffic on all lightpaths coming in the node p will be equal to the amount traffic going out of it.

$$\sum_j f_{e,j} \leq \mu \quad \forall e \in E \qquad , j = 1,2,3...K \tag{3}$$

Constraint (3) is used to ensure that the maximum amount of traffic flow on any lightpath is less than the congestion μ.

In the above formulation, as indicated before, the traffic between a specific source-destination pair is considered as a single commodity so that we are dealing with a large number ($O(n^2)$) commodities. Also, typically the degree of a node is at least 2. Therefore, the number of edges in the network is $O(n)$. This leads to a rapid increase in the number of variables, as well as the number of constraints as the network size increases. By analyzing constraint (2), we see that the number of constraints in the formulation and hence the basis size is $O(n^3)$. Similarly, the number of variables (f_{ej}) is also $O(n^3)$. This means that the standard LP formulation can only be solved for relatively small networks.

For example, if the number nodes is 30, the number of constraints is 27000.

3.2 Main Idea behind Approximation algorithms

For any linear programming (LP) formulation of a problem (the *primal* formulation) there exists a corresponding *dual* formulation problem. In chapter 2, we have reviewed the following well-known relationships between a primal and its corresponding dual problem.

i) If the primal is a maximization problem, then the dual is a minimization problem and vice versa.

ii) For the *optimal* solution, objective values for the primal and the objective value for the dual are the same.

iii) The number of constraints in the primal is equal to the number of variables in the dual and vice versa.

iv) The RHS coefficients of the primal become the objective function coefficients in the dual and vice versa.

Based on the above relationships, we can make the following observations. Let *P1(D1)* be a feasible solution of the primal (dual) formulation. Also, let obj_p (obj_d) be the objective function value corresponding to *P1 (D1)*. Then, if $obj_p = obj_d$, we know (based on relationship ii) that this is the optimal value. Otherwise, if the primal is a maximization (minimization) problem, $obj_p < optimal\_value < obj_d$ (respectively $obj_d < optimal\_value < obj_p$). In this case, by comparing the values of obj_p and obj_d, we can determine how close we are to the optimal value. For example, if $obj_p = 100$ and $obj_d = 110$, we can say that our solution is within 10% of the optimal solution, even though we do not know exactly what the optimal solution is. This is the main concept behind the approximation algorithm presented in this chapter.

3.3 Formulation of the minimum congestion problem

In this section, we consider the arc-chain formulation of the minimum congestion problem introduced in section 3.1. As before, we consider the logical topology of a WDM network viewed as a $G = (V, E)$, where V is the set of the n end-nodes of the WDM network and E is the set of m logical edges. We denote a logical edge from node x to y by $x \rightarrow y$. We consider each non-zero traffic between a source-destination node pair (s_j, t_j) to be one commodity. If there are K node pairs with non-zero traffic between them, there will be K commodities flowing over the network. In our formulation, P_j denotes the set of paths (from s_j to t_j) for commodity j, $P_{a\rightarrow b}$ the set of paths which pass through the logical edge $a \rightarrow b$ and $P = \bigcup_{j=1}^{K} P_j$ is the set of all valid paths (for all commodities). As before, d_j is the traffic demand for commodity j. This means that d_j units of traffic must be sent over the network, from s_j to t_j $1 \le j \le K$. We will use $x(p)$ to denote the traffic flow on path p. The objective of the optimization is to route the traffic in such a way that the demands for all commodities are met and the congestion (μ) of the network is minimized. The linear programming formulation for minimizing the congestion is given below.

Objective: $\quad \text{Min } \mu \qquad (4)$

$$\sum_{p \in P_{a\rightarrow b}} x(p) \le \mu \quad \forall\, a \rightarrow b \in E \qquad (5)$$

$$\sum_{p \in P_j} x(p) \ge d_j, \forall j \quad j=1, 2, 3, \ldots, K. \qquad (6)$$

$$x(p) \ge 0, \qquad \forall p \in P \qquad (7)$$

(4) gives the objective function, which states that the congestion should be minimized. Constraint (5) is actually a composite constraint. It corresponds to m individual constraints, one for each edge in the network. For each edge $a \rightarrow b \in E$, $\sum_{p \in P_{a \rightarrow b}} x(p)$ gives the total traffic flow on edge a → b, summed over all paths containing the edge a → b, for all commodities in the network. Constraint (5) then states that the total traffic flow on edge a → b must be less than or equal to the congestion μ of the network. This constraint must be satisfied since congestion is, by definition, the maximum traffic flow on any edge.

Constraint (6) is another composite constraint. It corresponds to K individual constraints, one for each commodity in the network. For each commodity j, $1 < j < K$, constraint (6) states that the total traffic flow for commodity j, over all paths in P_j, must be greater than or equal to d_j. In other words, constraint (6) ensures that the traffic demands for each commodity are satisfied. Constraint (7) simply states that all traffic flows must be positive.

3.3.1 Primal-dual formulation for congestion minimization

We now consider the basic formulation we described in equations 4-7, apply some transformations for convenience to give us a primal formulation that we will use in our algorithm. Then we will construct the corresponding dual formulation. Our transformations are such that

i) it is easy to obtain initial feasible solutions for both the primal and corresponding dual formulations and

ii) It is possible to improve the primal and dual solutions iteratively, without using the time consuming standard solution methods (e.g. revised simplex method [V96]) for linear programs.

Let $\mu = d_{max} / \lambda$, where $d_{max} = max \{d_j : j = 1, 2, 3, \ldots, K\}$ (8)

Substituting $\mu = d_{max}/\lambda$ in (4) we get a new objective function – minimize d_{max} / λ. This is equivalent to maximizing λ, since d_{max} is a constant.

Substituting (8) into (5), we get:

$$\lambda \sum_{p \in P_{a \to b}} x(p) \leq d_{\max}, \forall a \to b \in E \quad (9)$$

Let $y(p) = \lambda x(p)$. Equations 4 - 7 may be restated as follows:

Maximize λ (10)

$$\sum_{p \in P_{a \to b}} y(p) \leq d_{\max}, \forall a \to b \in E \quad (11)$$

$$\sum_{p \in P_j} y(p) \geq \lambda d_j, \forall j, \qquad j = 1, 2, 3, \ldots, K \qquad (12)$$

We have m constraints in (11) – one for each edge. We associate dual variable $l(e)$ with edge e, one dual variable for each of the m constraints of (11). We have K constraints in (12) – one for each commodity. We associate dual variable z_j for for the j^{th} commodity 1 # j # K in (12). Then, the dual formulation for (10) – (12) is obtained as follows:

$$\text{Minimize} \qquad d_{max} \sum_{e \in E} l(e) \qquad (13)$$

$$\sum_{e \in p} l(e) - z_j \geq 0 \qquad p \in P_j \quad j=1,2,3.....K \qquad (14)$$

$$\sum_{j=1}^{K} d_j z_j \geq 1 \qquad (15)$$

At any given time, we have positive values for the dual variables so that the above constraints 16-17 are satisfied. This means $l(e) > 0$, for all edge $e \in E$. We now discuss how we get, in the next iteration, a new value for each of the dual variables satisfying the constraints 16-17.

We will use $l(p_j^{*})$ to denote the length of the shortest path for commodity j. We now define D^* and z_j^* and $\hat{l}(e)$ as follows:

$$D^* = \sum_{j:1 \le j \le K} d_j l(p_j^{*)} \text{ and } z_j^* = l(p_j^*)/D^* \qquad (16)$$

$$\hat{l}(e) = l(e)/D^*, \quad e \in E \qquad (17)$$

Theorem 1:

If flows in the network are assigned based on arc lengths $l(e)$, there is a feasible solution of the dual formulation with arc lengths $\hat{l}(e) = l(e)/D^*$, $\forall e \in E$.

Proof

Since $\sum_{j \in k} d_j z_j^* = \sum d_j l(p_j^*)/D^* = 1$, if we substitute z_j by z_j^*, then constraint (15) is always satisfied.

The length of the p^{th} path for commodity j is $\sum_{e \in p} l(e)$. Since $l(p_j^*)$ is the length of a *shortest* path for commodity j, the length of any other path for commodity j must be greater than or equal to $l(p_j^*)$. Therefore $\sum_{e \in p} l(e) \ge l(p_j^{*)}$. Hence, relationship (18) follows.

$$\sum_{e \in p} l(e)/D^* \geq l(p_j^*)/D^*$$

(18)

By using the value of z_j* from (15), we get (19)

$$\sum_{e \in p} l(e)/D^* \geq z_j^* \quad p \in P_j, \quad 1 \leq j \leq K \tag{19}$$

Then, using (16) and (17), we can write

$$\sum_{e \in p} \hat{l}(e) - z_j \geq 0 \qquad p \in P_j, 1 \leq j \leq K \tag{21}$$

(21) shows that $\hat{l}(e)$ satisfies constraint (16) of the dual formulation.

Therefore, the dual objective value for a feasible solution (satisfying constraints (16) and (17)) is:

$$d_{max} * \sum_{e \in p} \hat{l}(e) = d_{max} * \sum_{e \in p} l(e)/D^* = d_{max} \sum_{e \in E} l(e) \; / \sum_{j \in k} d_j l(p_j^*) \tag{22}$$

The objective value of the primal can be obtained as follows:

Suppose we are looking at the solution at the end of r iterations. After r iterations, all demands have been sent over the network r times. Then the amount of flow sent for commodity j is

$$rd_j, \qquad j=1,2,3.....K.$$

Assume $\{ f_r(e) : e \in E \}$ is the flow on edge e at the end of r iterations for all edges. Scaling all arc flows $\{ f_r(e) : e \in E \}$ by r would provide a solution to the primal. We know that the congestion (μ) is the maximum flow on any edge in the network. Therefore,

$$\mu = \frac{1}{r} Max\{f_r(e) : e \in E\} = \frac{1}{r} f_r^*$$

The objective value to be maximized, in the primal formulation, is λ. Therefore, using eqn (8), $\lambda = d_{\max} / \mu = r * d_{\max} / Max\{f_r(e) : e \in E\}$ Hence primal objective value is

$$\lambda = \frac{d_{\max}}{\mu} = r d_{\max} / f_r^* \qquad (23)$$

Therefore, by using (22) and (23), we can calculate the primal and dual objective values. When these values are close enough (based on some predetermined limit), we can say we have obtained a "good enough" solution.

3.4 The Approximation algorithm

In this section, we describe the approximation algorithm itself. We show how the primal and dual objective values are calculated and how the stopping criterion is applied.

1. Choose values for δ and ε ($\delta > 0$ and $\varepsilon > 0$)

2. $l(e) := \delta / d_{\max}, x(e) = 0, \forall e \in E$

3. r = 0

4. do

   a. $r = r+1$

   b. for $j=1$ to K

      i) $l(p_j) :=$ shortest path distance for commodity j, $j=1,2,3.....K$

      ii) $x(p_j) := x(p_j) + d_j$

      iii) $l(e) := l(e)[1 + \varepsilon d_j / d_{\max}]$ $e \in p_j$

end for

c. primalSolution $= rd_{\max} / \max\{x(e) : e \in E\}$

d. dualSolution $= \sum_{e \in E} l(e) \; / \; \sum_{j \in k} d_j l(p_j)$

e. u= $\max\{x(e) : e \in E\}$ / r

while ((dualSolution / primalSolution) $< (1+\varepsilon)$)

The first step in the algorithm is to set appropriate values for δ and ε. These determine the accuracy of the final solution and speed of convergence of the algorithm. For example, if we want the final solution to be within 1% of the optimal value, we should choose ε = 0.01. In our experiments we have used $\delta = 0.001$ and $\varepsilon = 0.01$.

In step 2, we set the initial values for the dual variables *l(e).* We start with very small non-zero values (δ/d_{max}) for each arc length.

In step 3, we initialize the iteration counter *r*.

Step 4 is the main iterative step of the algorithm and is repeated until the final solution is found. In this step, we first update the iteration counter r (step 4a). Then we consider each commodity in turn (step 4b) and for each commodity *j*, we take the following actions:

i) We calculate the shortest path for each commodity, based on the arc lengths *l(e).* We have three different implementations of the approximation algorithm, based on how the shortest path is calculated. These variations will be discussed in detail in section 3.5.

(ii) Send dj units of flow along the shortest path p for commodity j, and update the flow on each edge $e \in p$, by dj.

(iii) Update the length of each edge $e \in p$ as follows: $l(e) := l(e)[1 + \varepsilon d_j / d_{\max}]$

In step 4c (after we have considered all the commodities in 4b) , we calculate the primal objective value (L) and dual objective value (U), using equations (19) and (20). Finally, we compare L and U and if they are close enough ($U/L < 1+\varepsilon$), we can stop. Otherwise, we go back to step 4 and start the next iteration.

3.5 Shortest Path algorithm

We use Floyd-Warshall algorithm, Dijkstra algorithm, and Efficient Dijkstra algorithm, to find the shortest path for each commodity. Of these three algorithms, the first two are well known and have been discussed in chapter 2. We have developed the Efficient Dijkstra algorithm, based on the Dijkstra algorithm [AMO93] [AMT90], and we describe it in detail in this section.

3.5.1 Efficient Dijkstra Algorithm

Dijkstra's algorithm gives the tree of shortest paths from the source node s to every other nodes. For example, Figure 3.2 shows the shortest path tree from the source node 1 to every other node in the network, corresponding to the graph shown in Figure 3.1.

In our approximation algorithm, suppose we send traffic demand d_{13} along the shortest path 1→2→3 (Figure 3.2), then, our algorithm increases the length of the edges 1→2 and 2→3. Subsequently, if want to send traffic demand d_{14} we will have to apply Dijkstra's algorithm again on the graph (Figure 3.1). This is because the path 1→2→4 (Figure 3.2), may no longer be the shortest path for this commodity, since the lengths of edges 1→2 and 2→3 have been increased by the previous operation. However, we note that increasing the lengths of edges 1→2 and 2→3 does not affect the whole tree, but only the subtree S1 in Figure 3.2. The Efficient Dijkstra algorithm is an efficient way to find the shortest paths for a specific subtree, without running the complete Dijkstra's algorithm on the entire graph. The Efficient Dijkstra algorithm takes an initial shortest path tree S_{init}(obtained by running Dijkstra's algorithm once on the entire graph) and updates this tree to reflect changes in arc lengths. It also takes a subtree *S1*, such that the shortest paths from the source node s to the nodes in *S1* need to be recalculated.

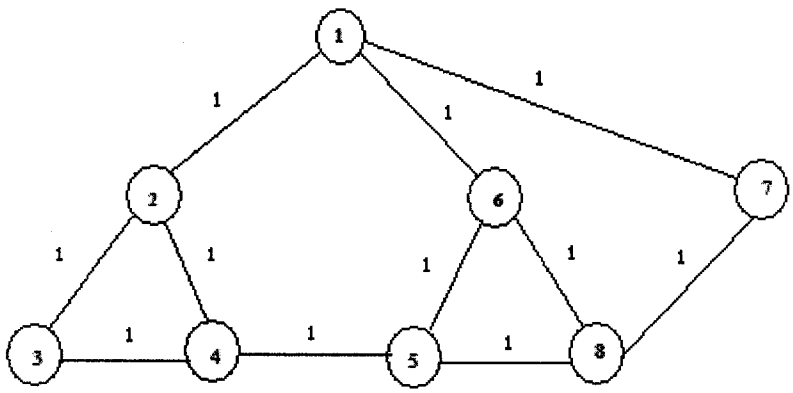


Figure 3.1 A WDM network

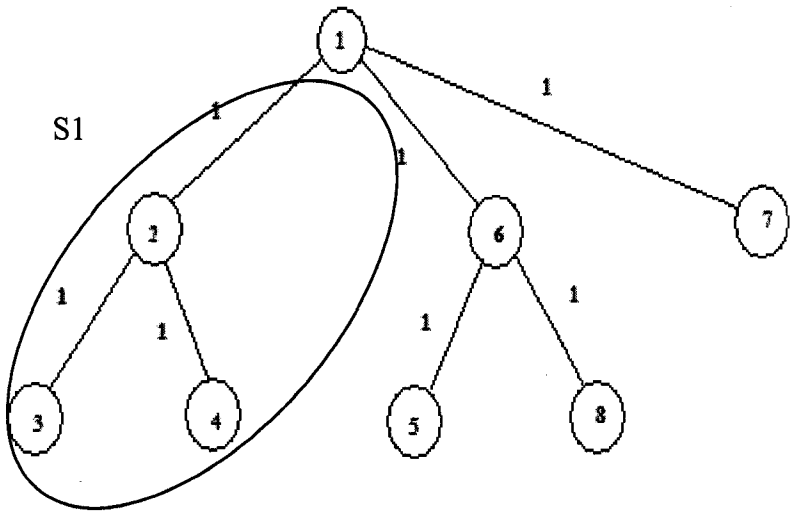


Figure 3.2 The shortest paths from node 1 to other nodes

The Efficient Dijkstra algorithm is given below :

1. Initialise

   a. T = Set of nodes in affected subtree $S1$

   b. $S = V - T$

   c. $d[w] = \infty$, $\forall w \in T$ and $d[w]$ = distance from s to w in S_{init} $\forall w \in S$

   d. $p[w] = null$ $\forall w \in T$ and $p[w]$ = predecessor of w in S_{init} $\forall w \in S$

2. For each node u in S

   For each node v in T

   { If (edge $u \rightarrow v \in E$) Then

   If $d[v] > d[u] + l(u \rightarrow v)$

   {

   $d[v] = d[u] + l(u \rightarrow v)$

   $p[v] = u$

}

}

3. While there are still vertices in T,

a. Sort the vertices in T according to current best estimate of their distance from s

b. Add u, the closest vertex in T, to S, and remove it from T

c. Relax all the vertices v (still in T) that are adjacent to u as follows:

If $d[v] > d[u]+l(u \rightarrow v)$

{

$d[v] = d[u]+l(u \rightarrow v)$

$p[v] = u$

}

Suppose that S_{init} is the initial tree shown in Fingure 3.2. After some flow is sent along path 1→2→3, the lengths of the edges along this path are increased. Suppose the new lengths are given by $l(1 \rightarrow 2) = 3$ and $l(2 \rightarrow 3) = 3$. We need to find a new shortest path tree S_{final}, based on these new arc lengths. We note that only the nodes in $S1$ are affected by the new arc lengths. Therefore, in step 1, the efficient Dijkstra algorithm puts only nodes from subtree $S1$ into the temporarily labeled set T. The remaining nodes are permanently labeled based on the initial shortest path tree S_{init}. So, at the end of step 1 $S = \{1, 5, 6, 7, 8\}$ and $T = \{2, 3, 4\}$.

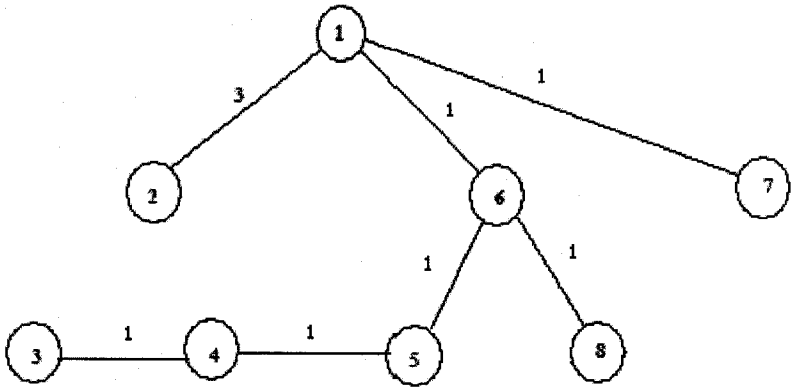


Figure 3.3 The final tree

In step 2, we check if there are any nodes in *T*, which are directly connected to a node in *S*. If so, we update the distance and predecessor of such nodes. In our example, nodes 2 and 4 are connected to nodes 1 and 5 respectively, from *S*. Therefore, the predecessor of node 2 is set to 1 and that of node 4 is set to 5. The corresponding distances from the source are both set to 3. After this, step 3 proceeds exactly like the original Dijkstra algorithm, until a new shortest path tree is found. The final tree is shown below in Figure 3.3.

4. Implementation Details and Experimental Results

4.1 Implementation Details

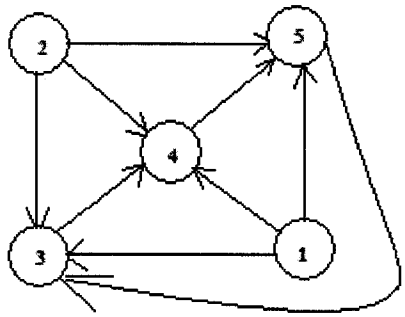
In this chapter, we describe our implementation of the approximation algorithm presented in chapter 3. We also present experimental results on the performance of the approximation algorithm and compare these results with solutions obtained by standard LP techniques. We have carried out our experiments on a large number of networks, ranging in size from 5 nodes to 50 nodes. The results from these experiments will be analyzed at the end of the chapter.

4.1.1 Generating network specification

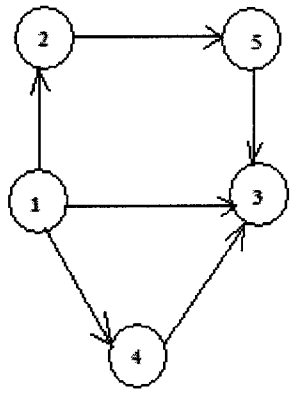
We have used an existing program [H03], to generate logical topologies of different sizes (up to 40nodes). This program takes in as input the underlying physical topology, including the number of available wavelengths per fiber and the number of transceivers per node and a traffic matrix that specifies the traffic demand for each commodity (source-destination pair) in the network. Based on this information, a logical topology is generated which is guaranteed to be able to accommodate the entire traffic demand. Given this logical topology, our objective is to route the traffic over the topology in the

most efficient manner. The traffic matrices used to create the logical topology were generated randomly to reflect low, medium and high load conditions in the network.

The physical topology and traffic matrix for a 5-node network is shown below in Figure 4-1a and 4-1b respectively. The corresponding logical topology (capable of accommodating the traffic demand) is shown in Figure 4-1c.



(a) 5-node physical topology



(c) 5-node logical topology

Traffic Matrix =

$$\begin{pmatrix} 0 & 18 & 18 & 13 & 15 \\ 11 & 0 & 6 & 8 & 9 \\ \mathbf{12} & 6 & 0 & 9 & 7 \\ 4 & 0 & 5 & 0 & 3 \\ 9 & 3 & 17 & 6 & 0 \end{pmatrix}$$

(b) 5-node traffic matrix

Figure 4.1. 5-node network

The entries in the traffic matrix are expressed as a percentage of the total capacity of a lightpath. For example, assume the capacity of a lightpath is 10Gb/s. Then the entry $t_{1\text{-}3}$ = 12, in the above traffic matrix, indicates that the traffic demand from source node 1 to destination node 3 is 12% of the capacity of a single lightpath or 1.2Gb/s.

4.2 Experimental Results

In this section, we present the experimental results. We are interested in i) the quality of the solutions, i.e. how close are they to the optimal solution and ii) the performance of the algorithms, i.e. how quickly can the solution be found. As a benchmark for the comparisons, we generate the optimal solution using the well-known optimization tool CPLEX [ILOG7]. The LP formulation optimized by CPLEX directly minimizes congestion and the constraints are generated using the node-arc representation. In our experiments, we have used δ=0.001, and ε=0.01.

Tables 4.1 – 4.7 give the results of our experiments. Each table corresponds to a particular network size. For example, Table 4.1 summarizes the results for the experiments with 5-node networks. The case number identifies a specific logical topology and traffic matrix combination. The results obtained by CPLEX represent standard LP solution techniques and produce an optimal solution (if a solution is found). D1, D2 and WA represent the results from our approximation algorithm when standard Djikstra's shortest path algorithm, efficient Djikstra (as presented in chapter 3) and Floyd-warshal algorithm are used to find the shortest paths respectively.

Case #	Minimum Congestion Using				Time Using			
	CPLEX	D1	D2	WA.	CPLEX	D1	D2	WA.
1	29.50	29.53	29.55	29.53	0.077	0.260	0.280	0.290
2	57.00	57.00	57.00	57.00	0.073	0.100	0.100	0.100
3	29.50	29.76	29.93	29.76	0.077	0.220	0.240	0.230

4	46.00	46.00	46.00	46.00	0.069	0.210	0.200	0.240
5	60.00	60.00	63.00	60.00	0.069	0.080	0.150	0.080
6	29.50	29.53	29.55	29.53	0.076	0.270	0.270	0.270
7	57.00	57.00	57.00	57.00	0.073	0.090	0.100	0.100
8	29.50	29.76	29.93	29.76	0.077	0.230	0.230	0.240
9	46.00	46.00	46.00	46.00	0.078	0.220	0.210	0.210
10	60.00	60.00	63.00	60.00	0.072	0.080	0.150	0.080
11	29.50	29.53	29.55	29.53	0.075	0.270	0.280	0.280
12	57.00	57.00	57.00	57.00	0.073	0.110	0.100	0.110
13	29.50	29.76	29.93	29.76	0.077	0.220	0.230	0.230
14	46.00	46.00	46.00	46.00	0.072	0.220	0.220	0.230
15	60.00	60.00	63.00	60.00	0.075	0.080	0.150	0.090
16	29.50	29.53	29.55	29.53	0.076	0.260	0.280	0.280
17	57.00	57.00	57.00	57.00	0.074	0.110	0.090	0.110
18	29.50	29.76	29.93	29.76	0.086	0.230	0.240	0.240
19	46.00	46.00	46.00	46.00	0.071	0.210	0.210	0.220
20	60.00	60.00	63.00	60.00	0.073	0.080	0.150	0.090
21	29.50	29.53	29.55	29.53	0.084	0.260	0.270	0.280
22	_	57.00	57.00	57.00	_	0.100	0.100	0.100
23	29.50	29.76	29.93	29.76	0.077	0.230	0.220	0.230
24	46.00	46.00	46.00	46.00	0.074	0.210	0.200	0.220
25	60.00	60.00	63.00	60.00	0.069	0.080	0.150	0.080

Table 4.1. Minimum congestion and running time from 4 different methods of 5 nodes network

Case #	Minimum Congestion Using				Time Using			
	CPLEX	D1	D2	WA.	CPLEX	D1	D2	WA.
1	78.73	83.69	83.53	83.69	3.619	2.619	2.460	3.930
2	73.23	77.79	77.43	77.79	3.120	2.580	2.210	3.640
3	77.42	77.79	77.43	77.79	4.907	2.610	2.270	3.470
4	84.43	88.07	87.89	88.07	2.365	2.440	2.130	3.900
5	79.43	84.14	83.67	84.14	4.424	2.710	2.360	3.900
6	77.42	80.56	79.96	80.56	5.191	2.220	2.180	3.220
7	72.33	85.90	85.84	85.90	2.842	2.720	2.320	3.840
8	81.17	88.88	89.35	88.88	3.189	2.270	1.850	3.210
9	79.43	86.03	85.81	86.03	2.914	1.950	1.680	2.730
10	80.33	85.59	85.02	85.59	3.136	2.430	2.160	3.350
11	72.33	85.90	85.84	85.90	2.842	2.650	2.230	3.730
12	79.43	84.14	83.67	84.14	4.091	2.670	2.310	3.800
13	71.31	76.75	76.57	76.75	3.839	3.620	3.040	5.190
14	77.41	84.79	84.46	84.79	2.690	2.770	2.470	3.950
15	69.15	73.93	74.07	73.93	4.296	3.150	2.640	4.460
16	81.17	88.88	89.35	88.88	3.189	2.210	1.810	3.110
17	79.43	86.03	85.81	86.03	2.914	1.880	1.650	2.730
18	78.62	84.22	84.38	84.22	1.915	2.890	2.390	4.150
19	80.33	85.59	85.02	85.59	3.136	2.340	2.110	3.380
20	69.15	73.93	74.07	73.93	4.080	3.180	2.640	4.490
21	_	88.07	87.89	88.07	_	2.320	2.040	3.340

22	_	84.14	83.67	84.14	_	2.720	2.340	3.810
23	_	77.47	77.64	77.47	_	3.100	2.520	4.440
24	_	80.56	79.96	80.56		2.160	2.150	3.100
25	_	73.93	74.07	73.93	_	3.140	2.630	4.520

Table 4.2. Minimum congestion and running time from 4 different methods of 10 nodes network

Case #	Minimum Congestion Using				Time Using			
	CPLEX	D1	D2	WA.	CPLEX	D1	D2	WA.
1	59.89	63.34	63.84	63.34	15.006	6.660	5.020	12.140
2	72.00	74.68	74.71	74.68	18.037	4.340	3.780	8.070
3	58.09	63.21	63.31	63.21	18.167	6.300	4.230	11.380
4	64.06	67.39	67.34	67.39	11.937	4.830	3.810	8.920
5	59.89	63.34	77.63	68.14	15.535	6.460	3.610	10.410
6	62.25	66.85	66.54	66.85	23.590	5.060	4.450	9.340
7	72.00	74.68	74.71	74.68	23.143	4.080	3.630	7.700
8	59.73	65.37	65.30	65.33	21.476	5.700	4.000	10.600
9	64.00	67.91	67.61	67.65	16.810	4.970	3.630	9.750
10	64.31	68.08	67.71	68.03	10.589	5.570	5.420	10.250
11	59.89	63.34	63.83	63.34	17.078	6.460	4.750	11.510
12	72.00	74.68	74.71	74.68	21.571	4.260	3.630	7.620
13	54.65	58.31	58.33	58.31	20.990	5.720	4.060	10.390
14	64.06	67.39	67.34	67.39	11.469	5.000	3.870	8.840

15	64.31	68.23	67.47	68.14	23.121	5.360	6.000	10.280
16	59.89	63.34	63.83	63.34	17.521	6.440	4.840	11.630
17	72.00	74.68	74.71	74.68	21.571	4.190	3.620	7.510
18	54.65	58.31	58.33	58.31	19.955	5.890	4.090	10.340
19	64.06	67.39	67.34	67.39	11.658	4.950	3.880	8.900
20	64.31	68.23	67.47	68.14	23.376	5.430	5.800	10.360
21	59.89	63.34	63.83	63.34	18.110	6.370	4.750	11.650
22	72.00	74.68	74.71	74.68	22.076	4.190	3.590	7.700
23	54.65	58.31	58.33	58.31	20.445	5.710	4.080	10.380
24	64.06	67.39	67.34	67.39	11.714	4.940	3.960	8.930
25	64.31	68.23	67.47	68.14	23.002	5.440	5.860	10.380

Table 4.3. Minimum congestion and running time from 4 different methods of 14 nodes network

Case #	Minimum Congestion Using				Time Using			
	CPLEX	D1	D2	WA.	CPLEX	D1	D2	WA.
1	55.59	59.41	59.54	59.41	475.79	36.230	23.310	74.060
2	61.70	65.74	65.68	65.74	506.790	28.090	17.360	57.440
3	59.39	63.34	63.26	63.34	494.496	37.560	21.650	75.910
4	62.87	67.49	67.53	67.49	442.383	41.060	24.090	83.620
5	_	62.25	62.07	62.25	_	41.020	24.500	83.310
6	55.09	59.71	59.79	59.62	582.662	36.460	23.450	73.480
7	_	65.74	65.68	65.74	_	26.860	16.550	55.990

8	_	63.34	63.26	63.34	_	35.710	20.900	73.670
9	_	67.49	67.53	67.49	_	39.170	23.250	81.500
10	_	61.09	60.88	61.02	_	40.060	27.240	82.070
11	55.56	59.71	59.79	59.62	351.074	36.420	23.330	73.060
12	61.70	65.74	65.68	65.74	348.046	26.950	16.430	55.740
13	_	63.34	63.26	63.34	_	35.790	20.940	73.610
14	_	67.49	67.53	67.49	_	39.460	23.200	81.440
15	_	61.09	60.88	61.02	_	40.340	27.250	81.960
16	55.56	59.71	59.79	59.62	347.066	36.640	23.510	73.080
17	_	65.74	65.68	65.74	_	26.900	16.450	55.390
18	_	63.34	63.26	63.34	_	35.600	20.810	73.570
19	_	67.49	67.53	67.49	_	39.580	23.250	81.740
20	_	61.09	60.88	61.02	_	40.330	27.260	82.050
21	_	59.71	59.79	59.62	_	36.830	23.450	72.910
22	61.70	65.74	65.68	65.74	344.862	26.880	16.580	55.550
23	_	63.34	63.26	63.34	_	35.770	20.960	73.130
24	_	67.49	67.53	67.49	_	39.570	23.290	81.290
25	_	61.09	60.88	61.02	_	40.460	27.360	82.230

Table 4.4. Minimum congestion and running time from 4 different methods of 20 nodes network

Case #	Minimum Congestion Using				Time Using			
	CPLEX	D1	D2	WA.	CPLEX	D1	D2	WA.
1	56.37	60.16	60.02	60.21	3586.548	186.390	112.000	371.200
2	_	63.20	63.23	63.20	_	166.060	96.1000	326.830
3	_	60.70	60.61	60.70	_	169.460	89.890	346.430
4	_	63.87	63.90	63.87	_	228.530	120.570	460.910
5	_	64.24	63.98	64.24	_	183.630	98.980	371.920

Table 4.5. Minimum congestion and running time from 4 different methods of 25 nodes network

Case #	Minimum Congestion Using				Time Using			
	CPLEX	D1	D2	WA.	CPLEX	D1	D2	WA.
1	_	57.86	57.86	57.89	_	349.620	177.820	742.420
2	_	64.28	64.35	64.28	_	278.070	167.410	579.680
3	_	53.12	53.06	53.12	_	377.650	204.540	773.690
4	_	54.35	54.00	54.35	_	354.300	183.670	732.230
5	_	56.84	56.57	56.84	_	335.260	177.870	701.660
6	_	55.75	55.70	55.75	_	305.040	181.270	631.330
7	_	57.24	62.29	57.21	_	393.650	188.020	848.210
8	_	53.12	53.06	53.12	_	377.290	204.750	774.170
9	_	54.35	54.00	54.35	_	354.220	183.860	733.590
10	_	56.84	56.57	56.84	_	335.220	178.380	701.750

11	_	55.75	55.70	55.75	_	304.100	181.310	630.330
12	_	57.24	62.29	57.21	_	392.990	186.860	847.820
13	_	53.12	53.06	53.12	_	378.850	204.750	774.540
14	_	54.35	54.00	54.35	_	354.410	183.830	734.030
15	_	56.84	56.57	56.84	_	335.320	178.710	700.650
16	_	56.20	56.20	56.29	_	332.930	183.900	658.480
17	_	57.00	63.21	57.05	_	425.420	177.340	882.010
18	_	54.11	54.28	54.11	_	339.600	180.160	701.620
19	_	54.35	54.00	54.35	_	354.070	184.550	734.480
20	_	56.84	56.57	56.84	_	335.140	177.630	701.020
21	_	55.75	55.70	55.75	_	304.940	181.250	630.520
22	_	57.24	62.29	57.21	_	393.370	186.600	848.020
23	_	53.12	53.06	53.12	_	377.060	204.220	774.100
24	_	54.35	54.00	54.35	_	353.840	185.170	733.400
25	_	56.84	56.57	56.84	_	334.890	177.560	701.280

Table 4.6. Minimum congestion and running time from 4 different methods of 30 nodes network

Case #	Minimum Congestion Using				Time Using			
	CPLEX	D1	D2	WA.	CPLEX	D1	D2	WA.
1	_	48.67	57.86	_	_	8160.020	6380.123	_
2	_	48.18	48.47	_	_	7535.755	5760.248	_
3	_	49.20	47.98	_	_	7596.254	4620.157	_
4	_	48.32	48.06	_	_	8071.108	4852.214	_
5	_	48.41	48.97	_	_	7814.699	4960.170	_

Table 4.7. Minimum congestion and running time from 4 different methods of 50 nodes network

Node #	Average of Minimum Congestion Using				Average of Time Using			
	CPLEX	D1	D2	WA.	CPLEX	D1	D2	WA.
5	43.88	44.46	45.10	44.56	0.075	0.177	0.193	0.185
10	77.11	82.67	79.52	82.53	3.435	2.614	2.264	3.736
14	63.24	66.87	67.35	67.00	18.318	5.373	4.334	9.799
20	58.80	63.51	63.47	63.48	432.136	35.990	22.255	73.512
25	56.37	60.16	60.02	60.21	3586.548	186.390	112.000	375.200
30	_	55.87	56.60	55.88	_	351.090	184.857	730.841
50	_	48.56	50.27	_	_	7795.567	5314.582	_

Table 4.8. Average of Minimum congestion and running time for different networks

4.3 Analysis of the Results

From table 4.1 to table 4.7, we can see that CPLEX can only be useful for small network. The running time of the approximation algorithm is much shorter than CPLEX when the network is big or the number of commodities is large.

In table 4.9,

$$\Delta_1^* = \frac{\mu_{Ap} - \mu_{CPLEX}}{\mu_{CPLEX}} * 100\%$$

$$\Delta_2^{**} = \frac{T_{Ap}}{T_{CPLEX}} *100\%$$

μ_{Ap} : Minimum congestion using Approximation algorithms.

μ_{CPLEX} : Minimum congestion using CPLEX.

T_{Ap} : The time Approximation algorithms use

T_{CPLEX} : The time CPLEX uses.

From table 4.9, we can see the difference of the minimum congestion using D1, D2, WA. and using CPLEX is really small. But the time using D1, D2, WA. is much smaller than using CPLEX. For example, on 25-node network, Efficient Dijkstra only uses 2.9 % of the time CPLEX uses.

Node #	Δ_1* (%)			Δ_2** (%)		
	D1	D2	WA.	D1	D2	WA.
5	1.3	2.7	1.5	236.0	257.3	246.6
10	7.2	3.1	7.0	68.7	67.0	108.7
14	5.7	6.4	5.9	29.3	23.7	53.5
20	8.0	7.9	7.9	8.3	5.1	17.0
25	6.7	6.4	6.8	4.1	3.1	10.4

Table 4.9. The difference of Minimum congestion and running time between CPLEX and the approximation algorithms using D1, D2, WA.

5. Conclusion and Future Work

In this thesis, we have presented a practical and efficient method for routing traffic in WDM optical networks, based on the concept of approximation algorithms. The performance of our algorithm is must better than traditional approaches based on linear programming. Furthermore, we can guarantee that the final solution, obtained using the approximation algorithm, is within a pre-determined bound of the optimal solution.

The main contribution of the thesis is to formulate the congestion minimization problem for WDM networks as a MCNF problem and then use an approximation algorithm to solve this problem efficiently. This allows us to efficiently route traffic for practical sized networks and obtain near-optimal solutions.

We have implemented three different versions of our algorithm and compared their performance with each other, as well as with LP based exact solution methods. The results indicate that our method can be used for networks of up to 50 nodes. When focusing on approximating the Routing in WDM optical networks, we mixed this approach with specialized shortest path algorithms on our routing model.

Future Work

In this thesis, we have investigated two well-known algorithms for finding the shortest path for a specific commodity. Since this is a very important operation in our algorithm, it will be useful to investigate more efficient ways of carrying out this operation.

It would also be useful to investigate how the approximation algorithm performs (in terms of both speed and quality of solution) compared to heuristic algorithms which do not provide any performance guarantees.

References

[ABF84] A. Ali, D. Barnett, K. Farhangian, "Multicommodity network problems: Applications and computations," IIE Transactions, vol. 16, no. 2, pp. 127-134, 1984.

[AL94] Baruch Awerbuch, Tom Leighton, "Improved approximation algorithms for the multi-commodity flow problem and local competitive routing in dynamic networks", Proceedings of the twenty-sixth annual ACM symposium on Theory of computing, May 1994

[AMO93] Revindra K. Ahuja, Thomas L. Magnanti, and James B. OrLin, "Network flows," Prentice-Hall, Inc.

[AMT90] Revindra K. Ahuja, Orlin K. Mehlhorn, and R. J. Tarjan, "Faster algorithms for the shortest path problem," Journal of ACM, vol. 37, no. 2, pp. 213-223, 1990.[GK96] M.D. Grigoriadis and L.G.Khachiyan, "Approximate minimum-cost multicommodity flows," Mathematical Programming, 75:477-482, 1996.

[BCLPR03] M. Bouklit; D. Coudert; J.F.Lalande; C.Paul; and H. Rivano, "Approximate Multicommodity Flow for WDM Networks Design," Sirocco'03: colloquium on structural information and communication complexity, 2003, no 17, pp. 43-56

[BHV96] C.Barnhart, C.A. Hane, P. H. Vance, "Multicommodity Flow Problems, in integer programming and combinatorial oprimization," Proceedings of the 5th international IPCO conference, Vancouver, British Columbia, Canada, (1996) 56-71.

[BM99] Jit Biswas, David W.Matula, "Two flow routing algorithms for the maximum concurrent flow problem", Proceedings of 1986 fall joint computer conference on Fall joint computer conference, November 1999.

[BS93] C. Barnhart and Y. Sheffi, "A network-based primal-dual heuristic for the solution of multicommodity network flow problems," Transportation Science, vol. 27, pp. 102-117, May 1993.

[CGK92] I. Chlamtac, A. Ganz, and G. Karmi, "Lightpath communications: An approach to high bandwidth optical WAN's,"IEEE Transactions on Communications, Vol. 40, No. 7, pp. 1171--1182, July 1992.

[F62] R. Floyd, "Algorithm 97, shortest path," Comm. ACM, vol. 5, p. 345, 1962.

[F99] Lisa K. Fleischer "Approximating fractional multicommodity flow independent of the number of commodities," Proceedings of the 40th annual symposium on foundations of computer science, 1999.

[GJ98] N.Garg and J.Konemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," In 39th annual IEEE Symnposium on foundations of computer science, pages 300-309, 1998.

[G91] Paul E. Green, "The Future of Fiber-Optic Computer Networks," Computer , Volume: 24 Issue: 9 , Sept. 1991 Page(s): 78 –87

[GK98] N.Garg and J. Konemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," In 39th annual IEEE symposium on foundations of computer science, pages 300-309, 1998.

[GR00] O. Gerstel and R. Ramaswami, "Optical layer survivability-an implementation perspective", Selected Areas in Communications, IEEE Journal on , Volume: 18 Issue: 10 , Oct. 2000 Page(s): 1885 –1899

[H03] Ming Hou,"Heuristics for WDM path protection", Computer Science, University of Windsor, 2003, Master thesis.

[ILOG02] http://www.ilog.com

[RSK96] R. Ramaswami, K.N. Sjvarajan, "Design of logical topologies for wavelength-routed optical networks", Selected Areas in Communications, IEEE Journal on , Volume: 14 , Issue: 5 , June 1996, Pages:840 – 851

[LD82] Ford Jr., L. and Fulkerson, D., Flows in networks. Princeton, NJ: Princeton University Press, 1982.

[M92-1] B. Mukherjee, "WDM-based local lightwave networks. I. Single-hop systems, " Network, IEEE , Volume: 6 , Issue: 3 , May 1992 Pages:12 – 27

[M92-2] B.Mukherjee, "WDM-based local lightwave networks. II. Multihop systems," Network, IEEE , Volume: 6 , Issue: 4 , July 1992 Pages:20 – 32

[MRB94] B. Mukherjee, S. Ramamurthy and D. Banerjee, "Some principles for designing a wide-area optical network," IEEE/ACM Transactions on Networking, vol. 4, pp. 684-696, Oct. 1994

[SM90] Farhad Shahrokhi, D. W. Matula, "The maximum concurrent flow problem", Journal of the ACM (JACM), Volume 37 Issue 2

[SMM01] C.Siva, Ram Murthy and G. Mohan, "WDM Optical Networks: Concepts, Design, and Algorithms", Prentice Hall PTR, NJ, USA, November 2001.

[Y95] N. Young, "Randomized rounding without solving the linear program. In ACM/SIAM[1], pages 170-178.

[W62] S. Warshall, "A theorem on boolean matrics," Journal of ACM, vol. 9, pp. 11-12, 1962.

[W03] Ling Wang, "Shortest Paths and Multicommodity Network Flows," School of Industrial and Systems Engineering Georgia Institute of Technology, April 2003. PHD thesis.

[YB94] Bulent Yener, Terrance E. Boult, "Study of Upper and Lower Bounds for Minimum Congestion Routing in Light-wave Networks" Proceedings IEEE INFOCOM '94.

Vita Auctoris

NAME:	Yumei Lu
PLACE OF BIRTH:	Kunming, China
YEAR OF BIRTH:	1962
EDUCATION:	The Fouth High School, Yunnan, China
	1975-1978
	Yunnan University, Yunnan, China
	1978-1982 B.Sc.
	Kunming University of Science and Technology,
	Yunnan, China
	1991-1994 M.En.
	Univeristy of Windsor, Windsor, Ontario, Canada
	2000-2001 B.Sc.
	University of Windsor, Windsor, Ontario, Canada
	2001-1004 M.Sc.