June, 2004

LIDS Publication # 2615

Research supported in part by:

NSF Grant ECS-0218328

Simulation and Analysis of Various Routing Algorithms for Optical Networks

Meli, A

Simulation and Analysis of Various Routing Algorithms for Optical Networks

by

Ali S. Meli

Submitted to the Department of Electrical Engineering and Computer Science
on May 24, 2004, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Electrical Engineering

Abstract

The problem of Routing and Wavelength Assignment (RWA) has been receiving a lot of attention recently due to its application to optical networks. Optimal Routing and Wavelength Assignment can significantly increase the efficiency of wavelength-routed all-optical networks. To provide methods for effectively addressing the RWA problem, Professors Bertsekas and Ozdaglar of MIT Laboratory for Information and Decision Systems developed a novel framework that relies on the use of piece-wise linear functions for routing in static and dynamic scenarios [BO01], [OB03]. In this project, we have demonstrated that their method is capable of achieving superior performance. To arrive at this result, we defined various metrics for measuring the efficiency of routing algorithms for both the static and dynamic demands. Using these metrics, we performed a variety of simulations. As we have shown in this report, the outcome clearly indicates that the use of piece-wise linear cost functions is the key to achieving superior performance in all-optical networks.

Thesis Supervisor: Dimitri Bertsekas
Title: Professor

Thesis Supervisor: Asuman Ozdaglar
Title: Assistant Professor

Contents

Chapter 1

Introduction

1.1 Optical Networks

In order to meet the exploding bandwidth requirements of existing and emerging communications applications, all-optical networks have been gaining momentum. These networks have a tremendous bandwidth of around 50 terabits per second. However, the demand for point to point communication per application is not typically as much. Therefore, to better utilize the capabilities of all-optical networks, the bandwidth of an optical fiber is divided into multiple communication channels. Each channel corresponds to a unique wavelength. In other words, these optical networks employ wavelength division multiplexing.

The users of an optical network demand that data be sent from a source point to a destination point. These demands must be routed in the most efficient way over the network. First of all, the router needs to find uncongested paths between the source and destination. Furthermore, in all optical networks the router must assign a wavelength for the data while it is traveling in a link. This all-optical path, consisting of both the routing and the wavelength assignments on the route, is generally known as a *light-path*. The light-path is reserved for a point to point demand until it is terminated. At the termination, all the corresponding wavelengths become available on the light-path.

1.2 Routing and Wavelength Assignment

In all-optical networks, there might be different types of *wavelength continuity constraints*. First, the network might lack wavelength conversion capabilities altogether. In this case, a light path must occupy the same wavelength on all the links it travels across. Second, the network might have full conversion capability at all of its nodes. In this case, the wavelength assignment will not have a material effect on the network and the problem boils down to routing. Alternatively, the network might have wavelength conversion capabilities on only a portion of its nodes.

The problem of providing routes to the light-path requests and to assign a wavelength on each of the links along it is generally known as the *routing and wavelength assignment* (RWA) problem. The RWA problem has been extensively researched. Several methods have been proposed to solve the RWA problem. These methods differ in their assumptions about the traffic pattern, availability of the wavelength converters, and their objective functions. There are two classes of RWA problems based on the type of traffic demand: static and dynamic. In static RWA, the demand is assumed to be fixed and known. In this case the goal typically is to accommodate the demand while minimizing the number of wavelengths used on all links. In the dynamic case, the demands for light-paths vary over time. If the demands are known beforehand, the RWA problem is an off-line problem. However, if demand is both dynamic and stochastic, the problem is called on-line RWA.

1.3 Overview of Problem Formulations

Even in the simpler static case, the typical formulations for optimal light path establishment turn out to be difficult mixed integer programs. Specifically, the RWA problem over a network without wavelength converters has been proven to be NP-complete [CGK92]. Therefore, relaxed linear programs have been used to obtain bounds on the objective functions [RaS95].

A lot of recent work in WDM networks has been devoted to the maximum-load model [GSKR99], [GeK97], [RaSi98]. In the terminology of this formulation, the link load refers to the number of light paths that pass through each link. The network load is defined as the maximum link load in the network. The objective is to minimize the network load. Network load provides an upper bound on the number of wavelengths required. Often this results in over-designing the network by using more wavelengths than actually necessary.

There has also been significant interest in obtaining the performance of RWA algorithms under dynamic traffic assumptions [BaH96], [KoA96], [SaS96]. For this purpose, stochastic models are employed for the call arrivals and service times. The performance of all-optical networks have been studied when some simple RWA algorithms are used. The main goal in these studies has been to identify important network parameters that affect the blocking performance of the network.

In their paper, Bertsekas and Ozdaglar developed an efficient algorithmic approach for optimal routing and wavelength assignment [BO01], [OB03]. Their approach can be used for networks with no wavelength conversion and easily extends to networks with sparse wavelength conversion. Their general formulation is easily applied to dynamic and stochastically varying demand models, where it is important that present-time decisions take into consideration the effect of the uncertain future demand and availability of resources. In particular, they provide a quasi-static view of the RWA problem based on optimal multi-commodity network flow problems. They take into account the effect of the present time decisions on future resource availability by using a particular form of cost function: A convex piece-wise linear cost function with break points corresponding to integer values of link traffic.

This structure of the cost function is the key aspect of their formulation that distinguishes it from other approaches; the resulting formulation tends to have integer optimal solutions even when the integrality constraints are relaxed. Therefore, their method allows the problem to be solved nearly optimally by fast and highly efficient linear (rather than integer) programming methods in an overwhelming majority of cases. Thus, their methodology is not subject to performance degradations inherent in the alternative heuristic approaches.

Chapter 2

Linear Programming Formulation

2.1 Multi-Commodity Flow Problems

The RWA problem can be mapped into a multi-commodity network flow problem. Therefore, network flow problems provide a convenient notation for formulating RWA problems. The network can be described by a strongly connected graph $G = (V, E)$ where V denotes the set of nodes and E denotes the set of edges, also referred to as links.

2.2 Notation

Based on the multi-commodity flow formulation, we introduce the following symbols.

Symbol	**Definition**
l	A link in the network
w	An OD (Origin-Destination) pair in the network from node i to node j
r_w	Input traffic for OD pair w
P_w	Set of all paths that a particular OD pair w may use
x_p	The flow of path p for some OD pair w
L	Set of links
W	Set of all OD pairs
C	Set of wavelengths available on each link
f_l	The total flow on link l
D_l	Cost function associated with link l which in general is R $\rightarrow$ R

2.3 Integer-Linear Programming Formulation

In general, we have the following optimization problem:

$$\min \sum_{l \in L} D_l(f_l)$$

subject to constraints including conversion of flow. The total flow on link l can be expressed as:

$$f_l = \sum_{\{p|l \in p\}} x_p$$

We also must satisfy the input traffic of an OD pair, which results in the following constraint:

$$\sum_{p \in P_w} x_p = r_w$$

Furthermore, there might be a limit on the number of wavelengths that a link can support, which results in a link capacity constraint:

$$\sum_{p|l \in p} x_p \leq |C|$$

All of the constraints we have introduced so far are linear. Therefore, since the cost function is given by summing over the link cost functions, if we put certain conditions on the individual link cost functions we can formulate the problem as a linear program. Specifically, each of the link cost function maps the amount of flow on the link to a real number. If these cost functions are linear, the formulation becomes completely linear.

In addition to the constraints above we require x_p to be integer. The reason is that in the original problem we either assign a wavelength on a path to an OD pair or not. We can't assign a non-integer amount of wavelength to a path or OD pair. So far, we have obtained a linear program with integer constraints. In general, this is a very complex problem which is NP-hard.

2.4 Piece-Wise Linear Cost Functions

To address this problem, Ozdaglar and Bertsekas have proposed the following conditions on the cost functions [BO01], [OB03]:

a) The cost function of every link is convex, monotonically increasing, and piece-wise linear. Therefore, the cost function can be expressed using a set of linear constraints. Moreover, the marginal cost for a new light-path over a link increases as the traffic over the link increases. As a result, and as our numerical simulations have shown, the optimal solution to the program tends to choose paths with underutilized links and leaves room for future light-paths.

b) The breakpoints of each piece-wise linear link cost function are at the integer points $0, 1, \ldots, |C|$. The cost for flows larger than $|C|$ is ∞, effectively imposing a link capacity constraint. Therefore, the resulting optimal solution tends to be integer, eliminating the need for time-consuming integer programming techniques.

From now on, I will refer to this formulation as the Piece-Wise Linear (PWL) formulation.

Previous computational results show that these constraints on the cost function provide a very effective way to solve the RWA problem [OD03]. Actually, even with these constraints, a network designer can significantly alter the characteristics of a network by modifying the break points and slopes in the piece-wise linear cost function.

In networks with no wavelength conversion or sparse wavelength conversion, the above formulation should be slightly modified. However, the general features of the cost function are still preserved. Since the computational results we have are for networks with full conversion, I will not go over the details of these alternative formulations.

Chapter 3

Rounding

3.1 Rounding Problem

When we use the particular piece-wise linear cost function described in the previous chapter, it is very likely that the optimal solution is integer even without imposing an integrality constraint in the linear programming formulation [OB03]. However, there are cases for which the linear program results in a non-integer solution that needs to be rounded in some manner. In their original paper, Bertsekas and Ozdaglar review some of the special cases for which the potentially non-integer optimal solutions can be rounded without loss of optimality [BO01], [OB03]. Indeed, their papers suggest a heuristic for rounding non-integer solutions for piece-wise linear cost functions in general networks. Their method relies on fixing the traffic pattern for OD pairs which have integer traffic distribution. Based on their method, I used a similar heuristic for rounding that is slightly more general in the sense that it can be applied to a wider class of objective functions (not just sum of individual link costs). The heuristic I used works for all the multi-commodity flow problems that are formulated as linear programs.

3.2 Rounding Algorithm

Suppose we have an arbitrary network with a given set of OD pairs and a given number of wavelength channels on the links. We assume that we have full wavelength conversion capability at all of the nodes of the network. A feasible solution of this problem has the form $x = \{x_p | p \in P_w, w \in W\}$, where W denotes the set of all OD pairs, and P_w denotes the set of paths that some OD pair $w \in W$ may use.

At the start of iteration k, we have the subset S_k of the OD pairs (the w's), which are subject to optimization (that is, the non-integer path flow variables), and the complementary set of the variables not in S_k, which have x_p's permanently fixed at an integer number. Initially, S_0 is the set of all OD pairs. The 0_{th} iteration is basically solving the linear program. The k_{th} iteration of the algorithm consists of the following steps:

1. Choose one of the OD pairs with non-integer path flow variables (for a single OD pair, the number of non-integer path flow variables can not be equal to 1 because the path flow variables should sum up to 1).

2. Fix all the path flow variables (x_p's) for OD pairs not in S_k.

3. Impose the integrality constraint on the OD pair which was selected in step 1.

4. Solve the resulting integer-linear program.

5. Update S_{k+1} by including all the OD pairs with non-integer path flow variables (x_p's). If S_{k+1} happens to be empty, the algorithm is terminated.

At the termination, the resulting solution x specifies a feasible routing. Since this method does not make any specific assumptions about the cost function other than having a linear programming formulation, it can uniformly be applied to alternative cost functions (e.g. the Min-Max and Min Total Distant cost functions that I will introduce shortly).

Chapter 4

Alternative Methods

The ultimate goal of this project is to provide a fair comparison between the performance of the PWL formulation of section 2.4 and other formulations. This chapter introduces these alternative methods that we used for benchmarking the performance of the PWL formulation.

4.1 Universal Algorithms

In these methods, we use a “holistic approach” to the routing problem. That is, the algorithm or method requires us to look at the problem all at once rather than dividing the problem to substructures.

4.1.1 Optimal Integer Programming

In this method, as its title suggests, an integer program is used to obtain the optimal cost of the routing problem. Therefore, this is the best any routing algorithm can do. Our ultimate goal is to achieve the same results as this method. Here we want to solve the integer-linear programming problem defined by:

$$\min \sum_{l \in L} D_l(f_l)$$

So that:

$$f_l = \sum_{\{p|l \in p\}} x_p$$

$$\sum_{p \in P_w} x_p = r_w$$

$$\sum_{p|l \in p} x_p \leq |C|$$

and $x'p$-s are required to be integer.

4.1.2 Minimizing the Maximum Link Load

In this method, we focus on the link in the network that is carrying maximum traffic. We want to minimize the traffic that passes through such a link. The motivation for this method is that the load (traffic) that passes through a link corresponds to the number of the wavelengths that is required to transmit the data. This method can be formulated as

$$\min \max_{l \in L} f_l$$

So that:

$$f_l = \sum_{\{p|l \in p\}} x_p$$

$$\sum_{p \in P_w} x_p = r_w$$

$$\sum_{p|l \in p} x_p \leq |C|$$

Non-integer solutions can be eliminated either through use of the rounding algorithm introduced in Chapter 3 or by directly imposing the integrality constraint within the problem formulation and obtaining a linear-integer program. Of course, using the rounding algorithm requires less computational resources.

4.1.3 Minimizing the Total Link Load

In this method, we want to minimize the sum of the traffic in all the links. Or alternatively, we want to minimize the aggregate distant traveled by the light paths. The linear programming formulation is

$$\min \sum_{l \in L} f_l$$

Such that:

$$f_l = \sum_{\{p|l \in p\}} x_p$$

$$\sum_{p \in P_w} x_p = r_w$$

$$\sum_{p|l \in p} x_p \leq |C|$$

Again, non-integer solutions can be eliminated either through use of the rounding algorithm or by imposing integrality constraint inside the linear programming formulation.

4.2 Sequential Algorithms

In this class of algorithms, the cost objective used for incremental optimization is not guaranteed to be unique and is usually dependent on the ordering of assigning. This means that we look at each OD (Origin-Destination) pair (in an ad-hoc order) and distribute each unit of traffic one at a time. The sequential algorithms are very different from the previous class of algorithms because inherently their outcome depends on the order of demand routing.

4.2.1 Minimum Marginal Cost Routing

In this method, for each unit of traffic, we consider all the feasible paths and choose the path that results in minimal increase in the overall cost. In other words, we choose the path with the minimal marginal cost, where the marginal cost of a path is defined as the sum of marginal costs of the links in the path. Here is a step by step description:

1. Choose an OD pair w of the set of OD pairs that have not been routed yet.
2. For all paths P_w (paths that correspond to the OD pair w), calculate the marginal cost of routing one additional unit of traffic. Only consider the paths that have not been congested.
3. Choose the path with the lowest marginal cost.
4. Go to step 2 until all the demand for the OD pair w is satisfied.
5. Go to step 1 until there are no more OD pairs left to be routed.

4.2.2 Shortest Path Routing

This is the simplest method of all; for each unit of traffic, we use the shortest feasible path where the length of a path is defined as the number of links in it. Note that if we use purely linear costs, this method and the previous method are identical. Again here is a step by step description of this method:

1. Choose an OD pair w of the set OD pairs that have not been routed yet.
2. For all paths P_w (paths that correspond to the OD pair w), find the shortest path that has not been congested.
3. Route all the traffic demand of w through this path until either all the traffic demand for w is satisfied or the path is congested.
4. Go to step 2 until all the demand for the OD pair w is satisfied.
5. Go to step 1 until there are no more OD pairs left to be routed.

Chapter 5

Demand Modeling and Performance Measurement

Since the ultimate goal of this project is to assess the performance of different routing algorithms, it is essential that we come up with tests that are fair measures of the performance of these algorithms. Particular attention must be paid on how the traffic demand for routing is created so that a wide range of traffic patterns is covered. In general, the traffic pattern can be either static or dynamic.

5.1 Static Scenarios

In static scenarios, the traffic demand does not change over time and therefore the optimization should be performed once.

5.1.1 Demand Modeling

To simulate traffic demand in the static case we used a two step model:

1. Each OD (Origin-Destination) pair is determined to be either "on" (with probability p) or "off" (with probability $q = 1 - p$). The demand for OD pairs that are "off" is set to zero.

2. For the OD pairs that are "on", the demand is determined according to a Poisson distribution with factor λ, that is

$$\text{P(k units of demand)} = \frac{\lambda^k}{k!}e^{-\lambda}$$

This way of simulating demand enables us to cover a wide variety of cases ranging from having a lot of demand coming from very few OD pairs in the network (small p and large λ) to having nearly all of OD pair sending a small amount of traffic through the network (large p and small λ).

5.1.2 Performance Measurement

We can then run different optimization schemes on a large number of sample demands and compare their performance. In general, there might be cases that the demand cannot be satisfied; either because the problem is infeasible by nature or because the routing method used is not optimal. Furthermore, different routing methods can yield different values for the cost function. Therefore, we used the following two metrics when comparing our results for the static scenarios.

1. Probability of failure (also known as probability of blocking): This number can be estimated by dividing the number of sample traffic demands which couldn't be satisfied by the total number of samples. This number can be easily calculated for all the routing schemes that were used.

2. Average Cost: As its name implies, the average cost is calculated by taking average of the Piece-Wise Linear (PWL) cost function over sample demands. Particular attention must be paid to the sample space over which this average is calculated. The reason is that not all methods are able to find a routing (feasible solution) for a particular demand pattern. Since our aim is to use the average cost to compare different methods, only the samples that resulted in a feasible solution for all the algorithms should be included in the average.

The first metric determines the success rate of a routing scheme and the second determines the efficacy of that routing scheme in reducing the costs. Also note that in the static case, the methods that are solely based on linear programming have the same probability of blocking because in linear programming formulations feasibility depends only on the constraints (which are the same in all LP formulations) and is independent of the cost function.

5.2 Dynamic Scenarios

In dynamic scenarios, traffic demand changes over time. Each time the traffic demand changes, the routing needs to change to accommodate. We first need to come up with a reasonable framework to generate dynamic demand.

5.2.1 Demand Modeling

I assume that as long as the demand pattern is constant, the routing does not change. Therefore, without loss of generality, the problem can be simulated in discrete time. The reason is that we are only concerned with the changes in traffic pattern. Even in continuous time scenarios, the changes in traffic pattern happen at discrete points in time.

For discrete time demand modeling, I used an "accumulative model" in which *additional* traffic arrives at discrete points in time, and lasts for some interval. This can be modeled by a three step process:

1. Each OD pair is determined to be either have a change in traffic demand (with probability p) or stay constant (with probability $q = 1 - p$). I refer to change in traffic demand by *arrival.*

2. For the OD pairs that are going to have a change in demand, the magnitude (or intensity) of new demand is determined according to a Poisson distribution with factor λ_I, that is

$$\text{P(k units of demand)} = \frac{\lambda_I^k}{k!} e^{-\lambda_I}$$

3. In the absence of new arrivals, the demand of an OD pair will return to zero after a certain time. Again, I use a Poisson process (with parameter λ_T) to describe the duration (the time that it takes for the demand to return to zero).

$$\text{P(the new demand lasts for n samples)} = \frac{\lambda_T^n}{k!} e^{-\lambda_T}$$

5.2.2 Continuity Constraints

Another issue in dynamic scenarios is the continuity constraint. If there are no continuity constraints to satisfy when the traffic pattern changes, the problem reduces to a series of static routings. The continuity constraints make dynamic routing problems more delicate. For the simulations, I required the routing corresponding to OD pairs with constant traffic demand to remain constant. Any change in traffic pattern can only come from creation (or destruction) of demand on an OD pair. Obviously, this is not the only way to incorporate a continuity constraint. For example, one can use a penalty function to allow changes in routing even for OD pairs with unchanged traffic demand. The penalty function (which is to be added to the total cost function) takes into account the loss in quality of service which occurs as a result of re-routing.

5.2.3 On-Line vs. Off-Line Routing

The dynamic routing problem can either be off-line or on-line (real time). In the on-line dynamic case, the actual future traffic demand is not known and arrives in real time. However, in an off-line dynamic routing, the traffic demand for the entire routing time span is known beforehand. Obviously, routing in an off-line scenario can be performed more efficiently.

5.2.4 Performance Measurement

The performance metrics from the static case are also applicable to the dynamic scenarios. Furthermore, in dynamic scenarios we can use the average time that passes until a routing algorithm fails (T_F) as a third performance measure. Therefore, in dynamic RWA problems the following three performance measures will be used:

1. Probability of Failure (also known as probability of blocking): Similar to the static case, this number can be estimated by dividing the sample traffic demands which resulted in blocking by the total number of samples.

2. Average Cost: Again, this measure is identical to the static case: the average cost is calculated by taking average of the cost (according to PWL cost function) over sample demands. Since our aim is to use the average cost to compare different methods, only the samples should be included in the average that have a feasible solution according to all of the algorithms.

3. Time of First Blocking: This is a measure of how long (how many time samples) a method can survive without experiencing any blockings.

5.2.5 Benchmarks in Dynamic Routing Problems

In real-life applications most of the problems require on-line (real-time) routing. In comparing the performance of the different routing schemes, our goal is to find the one that has the best performance in the on-line case. Still, some insights about strengths and weaknesses of routing methods can be obtained by observing their performance under alternative constraints. In my simulations I used three classes of constraints:

1. On-Line Routing Constraints

2. Off-Line Routing Constraints

3. No Continuity Constraints, that is the problem is simplified by disregarding the continuity constraints and treating it as a series of independent static routing problems.

Of course, the cost of off-line routing will be less than the cost of on-line routing and routing under no continuity constraints will be less costly than off-line routing. Furthermore, for the sequential routing algorithms (shortest path and marginal cost), the cost of off-line routing and on-line routing is the same, because the sequential routing methods does not take into account any of the information about upcoming demand patterns.

Chapter 6

Simulation and Simulation Results

In this chapter, along with the simulation results, we will review the software infrastructure that was used for the simulations.

6.1 Overview Simulation Software

The simulations consisted of creating sample demands and then running various routing schemes on them. MATLAB and AMPL were at the heart of my simulation models. First of all, I created a MATLAB code that reads a network as a list of links and then finds all the possible paths for all the OD (Origin-Destination) pairs in the network. For simulation purposes, I only used the five shortest paths. I also used MATLAB to create the demand models. The demand models were created in form of data files that were readable by AMPL. Furthermore, the sequential routing methods were also performed in MATLAB, as they did not require any form of sophisticated optimization.

The rest of optimization was performed with AMPL. For the static case, it was straight forward, though some elaborate coding was required to incorporate the rounding algorithm. In the on-line dynamic routing case, the model was a little bit more complicated as the AMPL code needs to take into account the inter-temporal aspect of the problem. In the off-line dynamic routing case the model was considerably simpler.

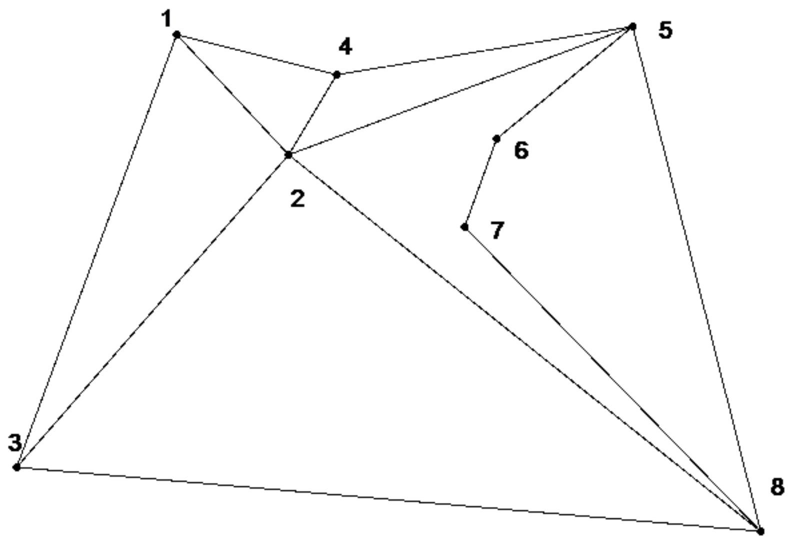

Figure 6-1: The 8 node network used for simulations

6.2 The Network and Cost Function

Figure 6-1 shows the network that was used in the simulations. As the picture shows, we used an 8-node network. Moreover, the slope of the cost function for each link has the form 2^f, where f denotes the traffic load of the link. Each link has a maximum capacity of 8 wavelengths.

We performed additional experiments with several other networks, including a 21-node network. While we did not compile a full set of computational results for these networks, the performance of the different algorithms was qualitatively similar to the case of the 8-node network.

6.3 Static Case

6.3.1 Table Specification

Table 6-1 shows the summary of results for the static case. The data in each row summarizes the simulation results for 100 traffic distributions that were generated by the corresponding pattern (as discussed below and also in chapter 5). Before discussing the data presented in that table, we first provide a column-by-column description of the items in the table:

- **λ:** In simulating the traffic in the network, I assumed that in an active OD pair, the amount of traffic follows a Poisson distribution. λ refers to parameter of the Poisson distribution. In other words, in an active link the magnitude follows:

$$\text{P(k units of demand)} = \frac{\lambda^k}{k!}e^{-\lambda}$$

- **P:** This number denotes the probability of activation for an OD pair. That is, each OD pair is either on (with probability P) or off (with probability $1 - P$). Only the active OD pairs can have traffic demand and their traffic is generated according to the Poisson distribution described above.

- **Average Cost:** This performance measure corresponds to the efficiency of each of the methods tested. As its name implies, this number is obtained by averaging the cost function over different network traffic samples. The costs were calculated using the objective function of PWL formulation.

  Moreover, the averaging of costs was only performed over samples that were successfully routed for all the routing methods. As different algorithms might fail for different traffic patterns, averaging over the intersection of the successful samples ensures fairness in our comparison.

- **P$(Fail)$:** This number corresponds to the number of traffic samples that could not be routed under a certain method divided by the total number of samples.

- **Integer Programming:** The data which comes under this heading was obtained by using an exact optimal methodology. In particular, an integer programming algorithm was used to solve Piece-Wise Linear, Min-Max and Total Distance formulations (all three methods are described below and also in chapter 4). Since the feasibility conditions for these methods are the same, they have the same probability of failure.

- **Linear Programming:** The data which comes under this heading was obtained by using a linear programming relaxation methodology with rounding. The rounding algorithm is described in Chapter 3. A linear programming algorithm with rounding was used to solve Piece-Wise Linear, Min-Max and Total Distance formulations (all three methods are described below and also in chapter 4). Due to rounding effects, in general the probability of failure under linear programming can be different under these three methods.

- **Sequential Routing:** The data that comes under this heading corresponds to methods that route traffic sequentially or "one at a time," as described in chapter 4. Basically, an OD pair is selected and after routing each unit of traffic, its path will be fixed and the next unit will be routed. We continue until all the traffic in the network is routed. In particular, I used sequential routing using both marginal costs and shortest paths. Both of these methods are described below (as well as in chapter 4).

- **PWL:** This refers to use of Piece-Wise Linear functions as the cost function, as described by Bertsekas and Ozdaglar in their papers [BO01], [OB03].

- **Min-Max:** This method minimizes the network load. Network load is defined as the maximum traffic that links in the network carry.

- **Min Total Dist.:** This method minimizes the total distance traveled by data in the network. Distance of travel is defined as the number of links (or nodes) that a unit of traffic needs to go through to reach its destination. This method is equivalent to minimizing the sum of link loads.

- **Marg. Cost:** This sequential method looks at the marginal cost associated with each path and then chooses the path with the least marginal cost. As mentioned above, after routing each unit of traffic, its path is fixed and we proceed to routing another unit of demand till all the traffic demand is met.

- **Shortest Path:** This sequential method chooses the path with the shortest distance (as measured by the number of links in a path). As mentioned above, after routing each unit of traffic, its path is fixed and we proceed to routing another unit of demand till all the traffic demand is met.

λ	P	*Integer Programming*				*Linear Programming With Rounding*						*Sequential Routing*			
		PWL	Min-Max	Min Total Dist.	P($Fail$)	PWL		Min-Max		Min Total Dist.		Marg. Cost		Shortest Path	
		Average Cost	Average Cost Cost	Average Cost		Average Cost	P($Fail$)	Average Cost	P($Fail$)	Average Cost	P($Fail$)	Average Cost	P($Fail$)	Average Cost	P($Fail$)
1	0.1	13.9	14.2	22.9	0	13.9	0	15.6	0	23.8	0	14.3	0	24.1	0
1	0.2	34.4	34.7	62.7	0	34.4	0	40.5	0	65.0	0	37.0	0	63.7	0
1	0.3	51.8	52.3	87.6	0	51.8	0	59.2	0	103.5	0	57.2	0	110.2	0
1	0.4	94.2	95.2	158.2	0	94.2	0	113.6	0	166.9	0	107.1	0	182.2	0
1	0.5	141.8	142.1	244.3	0.01	141.8	0.01	175.9	0.01	259.5	0.01	163.1	0.01	267.1	0.01
1	0.6	193.0	194.9	359.7	0	193.0	0	246.3	0	398.0	0	223.1	0	413.6	0.01
1	0.7	239.2	240.0	431.8	0.05	239.2	0.05	308.6	0.05	474.7	0.05	287.2	0.06	495.0	0.05
1	0.8	334.1	337.4	613.3	0.06	334.2	0.06	437.6	0.06	635.0	0.06	421.6	0.07	664.5	0.09
1	0.9	447.9	448.5	792.6	0.12	448.0	0.12	615.7	0.12	840.9	0.12	568.8	0.13	870.8	0.14
1	1	573.2	573.4	963.2	0.26	573.2	0.26	800.2	0.26	1011.7	0.26	771.7	0.27	1048.5	0.29
2	0.1	43.4	44.2	94.8	0	43.4	0	55.2	0	97.5	0	47.8	0	99.1	0
2	0.2	114.9	114.9	256.9	0	114.9	0	148.1	0	264.5	0	140.3	0.01	273.3	0
2	0.3	235.0	237.5	473.0	0.03	235.1	0.03	305.5	0.03	490.9	0.03	299.3	0.08	513.6	0.07
2	0.4	368.6	370.5	725.0	0.1	368.6	0.1	487.8	0.1	747.6	0.1	487.1	0.13	774.9	0.14
2	0.5	533.2	540.1	1035.2	0.32	533.3	0.32	753.5	0.32	1063.6	0.32	777.8	0.37	1099.8	0.35
2	0.6	759.2	764.9	1294.5	0.57	759.2	0.57	1066.1	0.57	1341.5	0.57	1095.9	0.68	1420.9	0.61
2	0.7	873.4	881.8	1384.9	0.78	873.4	0.78	1143.7	0.78	1406.0	0.78	1351.3	0.84	1492.2	0.86
3	0.1	90.4	93.1	226.4	0.01	90.4	0.01	118.5	0.01	245.2	0.01	102.1	0.01	248.8	0.03
3	0.2	257.7	261.0	564.3	0.08	257.7	0.08	341.8	0.08	588.1	0.08	321.9	0.12	619.7	0.11
3	0.3	465.7	470.2	903.8	0.39	465.7	0.39	617.8	0.39	927.6	0.39	620.6	0.47	970.7	0.49
3	0.4	606.0	606.0	1158.3	0.58	606.0	0.58	810.1	0.58	1193.4	0.58	923.9	0.75	1257.7	0.71
3	0.5	787.1	787.1	1367.1	0.78	787.1	0.78	1111.3	0.78	1450.8	0.78	1168.4	0.88	1622.2	0.89
4	0.1	167.8	171.6	376.9	0.07	167.8	0.07	224.8	0.07	417.6	0.07	201.5	0.07	446.1	0.09
4	0.2	469.9	472.5	896.3	0.33	469.9	0.33	626.2	0.33	939.0	0.33	655.0	0.38	1015.5	0.44
4	0.3	796.6	796.6	1321.5	0.57	796.6	0.57	1088.1	0.57	1343.0	0.57	1187.8	0.73	1455.5	0.68
4	0.4	982.0	982.0	1608.0	0.87	982.0	0.87	1480.8	0.87	1500.0	0.87	1505.4	0.94	1650.6	0.91
5	0.1	259.3	267.0	598.0	0.09	259.3	0.09	353.7	0.09	640.6	0.09	314.7	0.18	686.5	0.16
5	0.2	650.7	656.1	1177.5	0.61	650.7	0.61	868.8	0.61	1196.8	0.61	858.8	0.69	1272.5	0.7
5	0.3	748.4	796.4	1182.4	0.82	748.4	0.82	1029.0	0.82	1231.0	0.82	1087.4	0.92	1316.5	0.87

Table 6.1: Summary of simulation results for the static routing problems. Each row shows data corresponding to 100 samples. Use of piece wise linear cost function is clearly superior to other methods.

6.3.2 Discussion of Results

The data in table 6-1 proves the superiority of Piece-Wise Linear (PWL) method compared to other algorithms. It should be noted, however, that the cost function that was used in measuring average cost of algorithms was the PWL cost.

First, the data from integer programming formulations shows that PWL routing and Min-Max method outperform Minimum Total Distance formulation. Still, the performance of PWL and Min-Max method are relatively close. Note, however, that the integer programming formulation is impractical due to its excessive computational requirements for practical size networks.

Therefore, we need to look at the data from linear programming formulation (with rounding) to distinguish between the two methods. Table 6-1 clearly shows that using a linear program formulation causes significant deterioration in the performance of the Min-Max algorithm while the performance of PWL routing doesn't change very much. Since our goal is to come up with a fast routing algorithm, the data from linear programming formulation with rounding is more relevant. Based on this data, the PWL clearly outperforms any of the other methods.

The reason for the significant deterioration in the performance of Min-Max algorithm is that it results in non-integer solutions more often than the PWL formulation.

Moreover, the data from sequential routing has some interesting features. This method is much faster and less computationally intensive than the methods that use linear programming. Still, sequential routing using Marginal Costs out performs the Min Max method (with linear programming with rounding formulation). Naturally, using sequential routing with Shortest Path allocation has the worst performance of all, as it tends to over load certain links in the network.

6.4 Dynamic Case

Table 6-2 shows the simulation results for the dynamic case. The data in each row summarizes the simulation results for 100 traffic distributions samples, each consisting of 50 discrete-time samples, that were generated according to the corresponding pattern (as discussed below and also in chapter 5). Before discussing the results of simulation for the dynamic RWA problem, I will provide a description of data shown in table 6-2.

6.4.1 Table Specification

- **λ_I:** In simulating the dynamic traffic in the network, I assumed that when an OD pair is activated, a random amount of traffic demand is created and stays there for a random amount of time. The magnitude of traffic demand follows a Poisson distribution. λ_I refers to the parameter of this Poisson distribution. In other words, in an active link the increase in the magnitude of traffic is described by:

$$\text{P(k units of demand)} = \frac{\lambda_I^k}{k!} e^{-\lambda_I}$$

- **λ_T:** In simulating the dynamic traffic in the network, I assumed that when an OD pair is activated, a random amount of traffic demand is created and stays there for a random amount of time. The duration of the stay (in discrete time) follows a Poisson distribution. λ_T refers to the parameter of this Poisson distribution. In other words, in an active link the duration of the incremental traffic is given by:

$$\text{P(the increase lasts for t samples)} = \frac{\lambda_T^t}{t!} e^{-\lambda_T}$$

- **P:** This number denotes the probability of activation for an OD pair. That is, each OD pair is either on (with porbability P) or off (with probability $1 - P$). Only an active OD pairs can have increase in traffic demand. The increase in traffic is generated according to the Poisson distributions described above.

- **Cont.:** This column denotes to the type continuity constraint that was imposed on the network traffic. In a dynamic traffic problem, there might be some constraints that limit the change in traffic routing between consecutive samples (for example, the traffic distribution of an OD pair could be required to stay the same after it was initially routed). Moreover, the router might have certain information about future demand. The continuity constraint refers to the conditions on traffic patterns in consecutive time samples and the knowledge of the router about future demand. In my simulations, I use three types of continuity constraints:

1. **on-line:** In an on-line routing problem, the router does not have any knowledge about the upcoming traffic demand. Moreover, the router cannot change the routing for an OD pair unless the traffic demand on that OD pair changes.
2. **off-line:** In an off-line routing problem, the router has complete knowledge about the upcoming traffic demand. Still, the router cannot change the routing for an OD pair unless the traffic demand on that OD pair changes.
3. **no const.:** In this case, the constraints on traffic patterns between consecutive time samples are dropped. Therefore, the router's knowledge about about future demand becomes irrelevant as it will always choose a routing that minimizes the cost for the current traffic demand. Basically, the problem collapses to a series of static routings.

In general, we expect the results from *no const.* formulation to be better compared to *off-line* formulation as the router faces fewer constraints in the *no const.* formulation. Also, the results from *off-line* routing are expected to be better than the results of *on-line* routing as the router has more information in the *off-line* method.

Finally, note that for sequential routing methods, there is no difference between on-line and off-line problems as the knowledge about future traffic demand will not influence the decision of the router.

- **Average Cost:** This performance measure corresponds to the efficiency of each of the methods tested. As its name implies, this number is obtained by averaging the cost function over different network traffic samples. The costs were calculated by summing the cost function of PWL formulation over time samples and then taking the average.

Moreover, the averaging of costs was only performed over the samples that were successfully routed for all the routing methods. As different algorithms might fail for different traffic patterns, averaging over the intersection of the successful samples ensures fairness in our comparison.

- P($Fail$)**:** This number corresponds to the ratio of traffic samples that could not be routed under a certain method to the total number of samples.

- **Avg.** T_F**:** For the online routing problems, this number refers to the average number of time samples that passed before the network experienced its first failure. The average is calculated over the traffic demands that resulted in a failure. Since in off-line and no-continuity-constraint scenarios the notion of time is not clearly defined, this number is calculated only for the on-line routing.

- **Integer Programming:** The data which comes under this heading was obtained by using an exact integer programming methodology. Integer Programming was used to solve Piece-Wise Linear, Min-Max, and Total Distance formulations (all three methods are described below and as well as chapter 4).

- **Linear Programming:** The data which comes under this heading was obtained by using a linear programming relaxation methodology with rounding. The rounding algorithm that was used is described in Chapter 3. Linear programming with rounding was used to solve Piece-Wise Linear, Min-Max and Total Distance formulations (all three methods are described below and also in chapter 4). Due to rounding effects, in general the probability of failure under linear programming can be different under these three methods.

- **Sequential Rounding:** The data that comes under this heading corresponds to methods that route traffic sequentially or "one at a time", as described in chapter 4. Basically, an OD pair is selected and after routing each unit of traffic, its path will be fixed and the next unit will be routed. We continue until all the traffic in the network is routed. In particular, I used sequential routing using marginal costs and sequential routing using shortest path. Both of these methods are described below (as well as in chapter 4).

- **PWL:** This refers to use of Piece-Wise Linear functions as the cost function, as described by Bertsekas and Ozdaglar in their papers [BO01], [OB03].
- **Min-Max:** This method minimizes the network load. Network load is defined as the maximum traffic that links in the network carry (Chapter 4).
- **Min Total Dist.:** This method minimizes the total distant traveled by data in the work. Distance of travel is defined as the number of links (or nodes) that a unit of traffic needs to go through to reach its destination. This method is equivalent to minimizing the sum of link loads.
- **Marg. Cost:** This sequential method looks at the marginal cost associated with each path and then chooses the path with the least marginal cost. As mentioned above, after routing each unit of traffic, its path is fixed and we proceed to routing another unit of demand till all the traffic demand is met.
- **Shortest Path:** This sequential method chooses the path with the shortest distance (as measured by the number of links in a path). As mentioned above, after routing each unit of traffic, its path is fixed and we proceed to routing another unit of demand till all the traffic demand is met.

λ_I	λ_T	P	Cont.	*Integer Programming*									*Linear Programming With Rounding*									*Sequential Routing*					
				PWL			Min-Max			Min Total Dist.			PWL			Min-Max			Min Total Dist.			Marg. Cost			Shortest Path		
				Avg. Cost	P_F	Avg. T_F	Avg. Cost	P_F	Avg. T_F	Avg. Cost	P_F	Avg. T_F	Avg. Cost	P_F	Avg. T_F	Avg. Cost	P_F	Avg. T_F	Avg. Cost	P_F	Avg. T_F	Avg. Cost	P_F	Avg. T_F	Avg. Cost	P_F	Avg. T_F
1	1	0.3	on-line	3021.6	0	N/A	3725.8	0	N/A	5616.4	0.02	32.0	3024.2	0	N/A	4156.2	0	N/A	5590.0	0.01	43.0	3211.3	0.02	26.0	5964.8	0.12	23.8
			off-line	2931.6	0	N/A	3619.8	0	N/A	5597.4	0	N/A	2931.6	0	N/A	3583.7	0	N/A	5675.2	0	N/A						
			no cont.	2878.8	0	N/A	3586.9	0	N/A	5611.0	0	N/A	2878.8	0	N/A	3543.6	0	N/A	5679.3	0	N/A	3187.7	0.01	N/A	5997.5	0.08	N/A
1	2	0.2	on-line	4766.5	0.1	28.9	6124.6	0.12	30.6	8895.8	0.17	27.6	4776.3	0.11	27.3	7067.4	0.14	28.0	8779.9	0.17	25.2	5112.8	0.14	26.3	9510.5	0.26	25.9
			off-line	4535.5	0.09	N/A	5816.3	0.09	N/A	8899.4	0.09	N/A	4535.7	0.09	N/A	5492.2	0.09	N/A	8890.7	0.09	N/A						
			no cont.	4403.1	0.09	N/A	5618.7	0.09	N/A	8945.1	0.09	N/A	4403.1	0.09	N/A	5476.5	0.09	N/A	8959.5	0.09	N/A	5016.9	0.16	N/A	9638.2	0.21	N/A
1	5	0.05	on-line	2264.8	0	N/A	2784.7	0	N/A	3905.7	0.03	28.7	2256.3	0	N/A	3338.8	0	N/A	3573.2	0.02	40.0	2282.9	0	N/A	4167.4	0.03	28.7
			off-line	2117.3	0	N/A	2600.5	0	N/A	3862.4	0	N/A	2117.4	0	N/A	2500.6	0	N/A	3941.4	0	N/A						
			no cont.	2053.4	0	N/A	2502.8	0	N/A	3869.8	0	N/A	2053.4	0	N/A	2479.3	0	N/A	3948.5	0	N/A	2215.7	0.01	N/A	4239.8	0	N/A
1	10	0.04	on-line	4355.5	0.1	22.3	5530.6	0.12	25.0	8067.1	0.16	27.9	4380.5	0.1	22.3	7125.1	0.11	24.2	7138.7	0.12	24.5	4437.1	0.1	22.1	8340.8	0.17	28.5
			off-line	4002.7	0.09	N/A	4981.2	0.09	N/A	7901.7	0.09	N/A	4002.7	0.09	N/A	4748.8	0.09	N/A	7977.4	0.09	N/A						
			no cont.	3849.3	0.09	N/A	4805.5	0.09	N/A	7887.4	0.09	N/A	3849.3	0.09	N/A	4756.3	0.09	N/A	8054.2	0.09	N/A	4357.0	0.09	N/A	8644.4	0.12	N/A
2	2	0.05	on-line	2174.3	0.1	18.8	2778.6	0.1	18.9	4894.8	0.1	21.5	2179.7	0.1	18.8	3036.2	0.11	21.0	4472.2	0.09	19.0	2218.7	0.11	20.5	4956.8	0.15	21.6
			off-line	2069.8	0.08	N/A	2545.0	0.08	N/A	4850.8	0.08	N/A	2069.8	0.08	N/A	2538.0	0.08	N/A	4744.9	0.08	N/A						
			no cont.	2014.6	0.08	N/A	2503.5	0.08	N/A	4843.1	0.08	N/A	2014.6	0.08	N/A	2510.7	0.08	N/A	4757.5	0.08	N/A	2183.4	0.09	N/A	5049.7	0.13	N/A
3	5	0.03	on-line	7922.4	0.8	20.2	11251.3	0.82	20.3	17444.3	0.83	19.3	7906.3	0.81	20.3	13314.9	0.83	20.6	17054.7	0.83	20.7	8255.3	0.82	20.2	17149.9	0.85	19.2
			off-line	7160.6	0.68	N/A	8751.9	0.68	N/A	16764.6	0.68	N/A	7161.7	0.68	N/A	8594.4	0.68	N/A	16622.7	0.68	N/A						
			no cont.	6733.4	0.68	N/A	9141.6	0.68	N/A	17655.1	0.68	N/A	6733.4	0.68	N/A	9141.6	0.68	N/A	17802.4	0.68	N/A	8015.7	0.82	N/A	18139.3	0.86	N/A
5	5	0.01	on-line	4234.3	0.51	23.8	5996.6	0.55	23.9	11695.0	0.53	23.4	4230.5	0.51	23.8	6288.9	0.56	24.5	11263.4	0.51	22.1	4240.2	0.53	23.5	11697.2	0.61	23.4
			off-line	3916.9	0.46	N/A	4899.4	0.46	N/A	11636.1	0.46	N/A	3916.9	0.46	N/A	4864.3	0.46	N/A	11553.7	0.46	N/A						
			no cont.	3740.5	0.46	N/A	5060.8	0.46	N/A	11784.4	0.46	N/A	3740.5	0.46	N/A	5089.6	0.46	N/A	11593.9	0.46	N/A	4124.1	0.54	N/A	12014.6	0.59	N/A

Table 6.2: Summary of simulation results for the dynamic routing problems. Each row shows data corresponding to 100 traffic distribution samples, each consisting of 50 time samples. Use of piece-wise linear cost function is clearly superior to other methods.

6.4.2 Discussion of Results

The data in table 6-2 shows similar patterns as in table 6-1. It proves the superiority of Piece-Wise Linear (PWL) cost function based routing method compared to other algorithms. Again, it should be noted that the cost function that we used in measuring average cost of algorithms was the PWL cost function.

First, the data from integer programming formulations shows that PWL routing and Min-Max method outperform Minimum Total Distance formulation. However, the difference between the performance of PWL and Min-Max method is not as significant under integer programming.

Therefore, we can look the at the data from linear programming formulation (with rounding) to distinguish between the two methods. Table 6-2 unequivocally shows that using a linear program formulation with rounding causes significant deterioration in the performance of the Min-Max algorithm while the performance of PWL routing doesn't change very much. Since our goal is to come up with a fast routing algorithm, the data from linear programming formulation with rounding is more relevant. Based on this data, the PWL clearly outperforms any of the other methods.

The reason for the significant deterioration in the performance of Min-Max algorithm is that it results in non-integer solutions more often than the PWL formulation.

Moreover, the data from sequential routing has some interesting features. This method is much faster and less computationally intensive than use of linear programming. Still, sequential routing using Marginal Costs out performs the Min-Max method. Of course, using a simple sequential rouging with Shortest Path allocation has the worst performance of all, as it tends to over load certain links in the network.

Finally, as expected, *no const.* formulation resulted in better numbers in comparison to the *off-line* method

Chapter 7

Summary and Suggestions for Future Work

7.1 Summary and Conclusion

We introduced a framework for comparing the performance of various routing algorithms. The framework was extended to cover both static and dynamic routing problems. Moreover, due to the statistical characteristics of the demand model that was used, a wide range of traffic demand possibilities was tested. Based on the simulation results, the performance obtained by using piece-wise linear cost functions is very close to optimal for both dynamic and static problems. Another relatively easy to implement, yet reasonably well performing, method was sequential routing with respect to marginal costs of paths.

7.2 Suggestions for Future Work

We can think of extending the project in three areas:

7.2.1 Using Alternative Cost Functions as the Performance Metric

Our results suggests that using Piece-Wise Linear cost function with heuristic rounding performs much better than the Min-Max method with rounding, specially under the dynamic case. One reason could be that Min-Max requires rounding more often than the PWL formulation. This might suggest that for the dynamic routing problem, even when we use the average Maximum Link Load as the comparison benchmark, the result of routing with PWL and rounding could be better than using the Min-Max method itself. A series of simulations can be performed to test this hypothesis.

7.2.2 Using Penalty Functions Instead of Hard Continuity Constraints

In these simulations, while performing the online dynamic routing, we required the traffic routing corresponding to an OD pair to stay constant as long as the demand on that OD pair remained constant. We can relax this constraint and replace it with a penalty function that takes effect when the continuity constraint is violated. This penalty function can capture the loss in quality of service that users suffer when their traffic is re-routed.

7.2.3 Using Penalty Functions Instead of Hard Limits on Link Loads

In these simulations, we used a hard constraint to limit the traffic on the nodes. Instead, one can use a penalty function to capture the costs of increasing the bandwidth on the links.

Bibliography

[OB03] A. E. Ozdaglar, A., and Bertsekas, D., "Optimal Solution of Integer Multicommodity Flow Problems with Application in Optical Networks," Proc. of Symposium on Global Optimization, Santorini, Greece, June 2003

[BaH96] Barry, R., and Humblet, P., "Models of Blocking Probability in All-Optical Networks with and without Wavelength Changers," IEEE J. Select Areas Commun., vol. 14, no. 5, pp. 858-867, June 1996.

[BO01] Bertsekas, D., and Ozdaglar, A., "Routing and Wavelength Assignment in Optical Networks," LIDS Report P-2535, December 2001; IEEE/ACM Transactions on Networking, no. 2, 2003, pp. 259-272.

[CGK92] Chlamtac, I., Ganz, A., and Karmi, G., "Lightpath Communications: An Approach to High-Bandwidth Optical WANs," IEEE Trans. Commun., vol. 40, pp. 1171-1182, July 1992.

[GSKR99] Gerstel, O., Sasaki, G., Kutten S., and Ramaswami, R., "Worst-Case Analysis of Dynamic Wavelength Allocation in Optical Networks," IEEE/ACM Transactions on Networking, vol. 7, no. 6, pp. 833-845, Dec. 1999.

[GeK97] Gerstel, O., and Kutten S., "Dynamic Wavelength Allocation in All-Optical Ring Networks," in Proc. ICC , pp. 432-436, 1997.

[KoA96] Kovacevic, M., and Acampora A., "Benefits of Wavelength Translation in All-Optical Clear-Channel Networks," IEEE J. Select. Areas Commun., vol. 14, pp. 868-880, 1996.

[RaS95] Ramaswami, R., and Sivarajan, K. N., "Routing and Wavelength Assignment in All-Optical Networks," IEEE/ACM Transactions on Networking, vol. 3, pp. 489-499, Oct. 1995.

[RaSi98] Ramaswami, R., and Sivarajan, K. N., "Optical Networks: A Practical Perspective," CA: Morgan Kaufmann, 1998.

[SAS96] Subramaniam, S., Azizoglu, M., and Somani, A. K., “All-Optical Networks with Sparse Wavelength Conversion,” IEEE/ACM Transactions on Networking, vol. 4, no. 4, pp. 544-557, June 1996.