1-1-2003

**Dynamic establishment of restorable lightpaths in WDM optical networks**

Haibo Luo  
*Iowa State University*

Follow this and additional works at: [https://lib.dr.iastate.edu/rtd](https://lib.dr.iastate.edu/rtd)

**Recommended Citation**  
[https://lib.dr.iastate.edu/rtd/19491](https://lib.dr.iastate.edu/rtd/19491)

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Dynamic establishment of restorable lightpaths in WDM optical networks

by

Haibo Luo

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Lu Ruan, Major Professor
Soma Chaudhuri
Arun Somani

Iowa State University
Ames, Iowa
2003

Copyright © Haibo Luo, 2003. All rights reserved.
This is to certify that the master's thesis of

Haibo Luo

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>vii</td>
</tr>
<tr>
<td>CHAPTER 1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Two Problems</td>
<td>4</td>
</tr>
<tr>
<td>1.2 Thesis Contribution</td>
<td>4</td>
</tr>
<tr>
<td>CHAPTER 2 TERMINOLOGY</td>
<td>6</td>
</tr>
<tr>
<td>CHAPTER 3 FAST RESTORATION METHOD</td>
<td>8</td>
</tr>
<tr>
<td>3.1 Key Idea</td>
<td>8</td>
</tr>
<tr>
<td>3.2 Performance Analysis</td>
<td>9</td>
</tr>
<tr>
<td>3.2.1 Restoration Time</td>
<td>9</td>
</tr>
<tr>
<td>3.2.2 Routing and Wavelength Assignment</td>
<td>11</td>
</tr>
<tr>
<td>3.2.3 Capacity Requirement</td>
<td>12</td>
</tr>
<tr>
<td>3.3 Numerical Results</td>
<td>14</td>
</tr>
<tr>
<td>3.4 Summary</td>
<td>18</td>
</tr>
<tr>
<td>CHAPTER 4 EFFICIENT LIGHTPATH ESTABLISHMENT</td>
<td>19</td>
</tr>
<tr>
<td>4.1 Previous Work</td>
<td>19</td>
</tr>
<tr>
<td>4.1.1 Centralized Algorithm (CA)</td>
<td>19</td>
</tr>
<tr>
<td>4.1.2 Simple Distributed Algorithm (SDA)</td>
<td>20</td>
</tr>
<tr>
<td>4.1.3 Other Related Work</td>
<td>21</td>
</tr>
<tr>
<td>4.1.4 Blocking Types</td>
<td>22</td>
</tr>
<tr>
<td>4.2 Load Balancing Heuristics</td>
<td>22</td>
</tr>
<tr>
<td>4.2.1 Random Version of Dijkstra’s Algorithm</td>
<td>22</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 3.1 Characteristics of Test Networks ........................................ 14
Table 3.2 Total Capacity Used and Average Backup Path Length in Network 1 . 16
Table 3.3 Total Capacity Used and Average Backup Path Length in Network 2 . 17
Table 3.4 Total Capacity Used and Average Backup Path Length in Network 3 . 17
Table 4.1 Blocked Demand Numbers By Each Version of Dijkstra's Algorithm . 28
Table 4.2 Blocked Demand Numbers With Different δ .............................. 29
Table 4.3 Blocked Demand Numbers With Different r and ν ...................... 30
Table 4.4 Blocked Demand Numbers By Each Algorithm .......................... 31
Table 5.1 Shortest Paths by Array Version and Heap Version .................... 34
Table A.1 Blocked Demand Numbers under Different r and ν without Primary Path Selection Heuristic ......................................................... 37
Table B.1 Blocked Demand Numbers under Different r and ν with Primary Path Selection Heuristic(δ=3) ......................................................... 38
LIST OF FIGURES

Figure 1.1 An Example of Backup Multiplexing .................................. 3

Figure 2.1 Supportable Graphs of Channel \( \lambda_{(ac),1} \) .......................... 7

Figure 3.1 One Primary Lightpath with Two Backup Lightpaths ............. 9

Figure 3.2 Backup Capacity Requirement: 2-backup v.s 1-backup. .......... 13

Figure 3.3 Test Network 1 ................................................................. 15

Figure 4.1 Original and Random Version of Dijkstra's Algorithm ........... 23

Figure 4.2 \( r \)'s Effect on Backup Path Selection ............................... 26
ABSTRACT

Lightpath provisioning for dynamic traffic is an important issue in WDM optical networks. Meanwhile, in order for a lightpath to survive a network failure, both a primary lightpath and a backup lightpath need to be found for each demand. A demand will be blocked if either can not be provided. Fast restoration and efficient lightpath establishment are two capabilities sought by service providers. In this thesis, we provide our approaches to address these two issues.

Current path-based proactive restoration scheme precomputes a backup lightpath when a primary lightpath is setup. Better spare capacity utilization can be achieved using backup multiplexing technique; however, long restoration time is inevitable because of the end-to-end signaling required to setup the backup lightpath upon failure. We present a new proactive lightpath restoration method that computes one primary lightpath and two backup lightpath segments for each traffic demand. Simulation results showed that the new method could significantly reduce the restoration time with only minor increase in capacity requirement.

We also propose an efficient establishment scheme of restorable lightpaths. Three heuristic ideas are proposed to exploit the wavelength usage information and to make link channels across the network used more evenly, leading to lower blocking probability. Simulation shows that, the load balancing routing algorithm (LBA), with our heuristic cost functions for primary path selection and backup path selection, achieves comparable performance as the centralized algorithm does. However, LBA asks for much less information to be disseminated, and therefore is more scalable.
CHAPTER 1  INTRODUCTION

To accommodate the increasing traffic, optical networks employing wavelength division multiplexing (WDM) [1] technology are becoming the backbone transport networks for the next generation Internet. WDM divides the tremendous bandwidth of a fiber into many non-overlapping wavelength channels, with each channel carrying data at the speed of several gigabits per second. A WDM network consists of optical cross-connects (OXC}s) interconnected by fiber links. By configuring the OXCs across the network, a lightpath – a point-to-point all-optical communication path occupying one wavelength channel in each link along its route – can be established to accommodate the traffic demand between the source and destination nodes. In the absence of wavelength converters, a lightpath must occupy the same wavelength on all the fiber links throughout the route of data transmission; this property is known as wavelength continuity constraint. OXCs without wavelength conversion capability are called wavelength selective cross-connects; on the other hand, OXCs with wavelength conversion capability are known as wavelength interchanging cross-connects.

Given a set of demands, the problem of setting up lightpaths by finding routes and assigning wavelengths to them is called the routing and wavelength assignment (RWA) problem. Typically, the traffic demands can be either static or dynamic. With static traffic, the entire set of demands is known in advance, and the problem is then to set up lightpaths for these demands in a global fashion while minimizing network resources such as the number of wavelengths or the number of fibers in the network. Alternatively, one may attempt to set up as many of these demands as possible for a given fixed number of wavelengths. The RWA problem for static traffic is known as the static lightpath establishment.

In a dynamic traffic pattern, a lightpath is set up for each connection request as it arrives,
and the lightpath is released after some finite amount of time. The objective in the dynamic traffic case is to set up lightpaths and assign wavelengths in a manner which minimizes the amount of blocking demands, or which maximizes the number of demands that are established in the network at any time. This problem is referred as the *dynamic lightpath establishment* problem.

Future communication infrastructure will consist of two layers: IP layer and WDM layer. In this IP-over-WDM architecture, IP routers are interconnected by lightpaths (also called logical links) provisioned by a WDM optical network. Since WDM networks carry high volumes of traffic, it is imperative that these networks be survivable, i.e., be able to reroute the disrupted traffic upon network failure. Although IP layer can reroute traffic flows around failures by routing table updates, the restoration time is in the order of minutes. Another problem in IP restoration is that a single fiber link failure can lead to a high number of simultaneous logical link failures in the IP layer, causing a large number of routing table updates. This calls for WDM layer restoration, which is much faster and simpler than IP layer restoration. In WDM layer restoration, the failed lightpaths are directly replaced by backup lightpaths at the WDM layer, so that IP layer will not notice the logical link failures at all.

Many lightpath restoration schemes have been proposed and a detailed classification was given in [2]. In general, restoration can be approached in two ways to recover from a link or node failure: *reactive* and *proactive*. The reactive restoration methods reroute the affected traffic after failure occurrence by using the available capacity in the network. It may fail if there is not enough spare capacity. In contrast, the proactive restoration methods reserve capacity for the *backup lightpath* while establishing the *primary lightpath*. During normal operation, traffic is carried on the primary lightpath; on detecting failure, the source OXC switches the traffic to the backup lightpath. Therefore, the protection schemes can ensure 100% survivability. Moreover, the restoration time is shorter than that of reactive method, which starts searching backup lightpath after failure occurs.

A proactive method can be either *link-based* or *path-based*. Link-based method finds a new path connecting the two end nodes of the failed link. This new path, along with the unaffected
portion of the primary path, form the backup path used to restore the traffic. On the other hand, path-based method finds an end-to-end backup path between the source and destination of the failed primary path. Link-based method is unattractive since it limits the choice of the backup paths and requires more spare capacity than path-based method [3].

Wavelength channels on the backup path in path-based method, can be either dedicated or shared. In the dedicated method, wavelength channels assigned to a backup path can’t be assigned to other backup paths. In the shared method, if two primary paths do not fail simultaneously, their backup paths can share wavelength channels, this is also known as backup multiplexing [4]. Under the single link failure assumption, two primary paths can share backup wavelength channels if they are link-disjoint because they will not fail simultaneously.

Backup multiplexing is illustrated in Figure 1.1. It shows two primary paths $P_1$ and $P_2$, and their respective backup lightpaths $B_1$ and $B_2$, all on the same wavelength, say $\lambda_1$. Since $P_1$ and $P_2$ are link-disjoint, they would not fail at the same time. Therefore, $B_1$ and $B_2$ can share wavelength $\lambda_1$ on link (ac). This shared channel will be used by $B_1$ when link (ab) fails, or by $B_2$ when either link (ad) or (de) fails. Better resource utilization can be achieved using backup multiplexing. The work in [5] shows that the total capacity requirement for the dedicated backup method is 260-265% of the capacity requirement without lightpath protection, and it reduced to 186-195% when backup multiplexing is used.

![Figure 1.1 An Example of Backup Multiplexing](image)
1.1 Two Problems

Although backup multiplexing greatly improves the spare capacity utilization, it requires longer restoration time than the dedicated backup method. In dedicated backup, both the primary lightpath and the backup lightpath are setup by configuring the cross-connects along their paths when a demand is accepted. If backup multiplexing is employed, wavelength channels are reserved along the backup lightpath, but the cross-connects can’t be configured at the connection setup time in order to allow for sharing of backup wavelengths. The configuration of the cross-connects along the backup path is done only after a failure occurs and it requires an end-to-end signaling that causes long delay in traffic restoration.

On the other hand, although dynamic lightpath establishment allows service providers to respond quickly to customer demands [6], the service providers also expect to fully utilize their network resources to bring the maximum profits. In addition, as we discussed earlier, the survivability should be provided as well. In this thesis, we study the dynamic routing of restorable lightpaths. When a lightpath demand arrives at an OXC node, a routing algorithm computes both primary and backup paths for it. When handling restorable lightpath establishment, we cannot but face the following two problems:

1. How can a restoration method achieve rapid restoration when a link failure occurs?
2. How can an efficient routing algorithm maximize the number of demands accommodated by the network?

1.2 Thesis Contribution

We give our solutions to each of the two questions in this thesis.

For question 1, we propose a new proactive restoration method in Chapter 3, which divides a primary path into two segments and finds backup paths for each individual segment separately. We call this new method 2-backup method and the conventional method that uses a single end-to-end backup path 1-backup method. We will show that compared to 1-backup method, the
new 2-backup method substantially reduces the restoration time with only a minor increase in capacity requirement.

The objective of question 2 can be redefined to select the “best” primary-backup path pair for each incoming demand, such that the network can accommodate a maximum number of demands; or equivalently, the blocking probability of arriving demands is minimized. In Chapter 4 we present an efficient distributed routing algorithm aiming to achieve this goal. The algorithm judiciously assigns link costs to the network graph when computing primary and backup paths for a given demand. Taking the wavelength usage information of each link into account, the algorithm balances the traffic load in terms of wavelength consumption across the network; as a result, low blocking probability is achieved.

Note that within both solutions we assume the use of wavelength-interchange cross connects, so the wavelength continuity constraint is released. In addition, only single link failure is assumed, however both works can be extended to allow single node failure as well.

Some terminologies are explained in Chapter 2. Then our solutions to the two problems are discussed in Chapter 3 and Chapter 4. Finally we present some ideas for future work in Chapter 5.
CHAPTER 2 TERMINOLOGY

A channel may be occupied by a primary path, reserved by one or more backup paths (recall that backup multiplexing allows more than one backup paths to share a wavelength channel), or has not been used yet. We call the channel in these three different cases as primary channel, backup channel and unused channel respectively.

Suppose we have a backup channel $\lambda$, a satisfied demand $d_s$ and an incoming demand $d_i$. $d_s$’s primary lightpath is said to be supported by $\lambda$ if $d_s$’s backup lightpath is now reserving $\lambda$. We call the graph consisting of all nodes and those links within the primary paths that are supported by the backup channel $\lambda$ as $\lambda$’s supported graph. Define the complement of the backup channel $\lambda$’s supported graph as $\lambda$’s supportable graph. (Obviously, the link which includes $\lambda$ should be removed in its supportable graph as well.) If the incoming demand $d_i$’s primary path can be contained in $\lambda$’s supportable graph, then it must be link-disjoint with the existing primary lightpaths supported by $\lambda$, therefore the link containing $\lambda$ can be used in the backup path for $d_i$ “freely”. We define the number of links in the $\lambda$’s supported graph out of the number of all links in the network graph as the backup channel $\lambda$’s supported ratio.

For example, assume the network in Figure 1.1 has accommodated one demand $d_1$, with the primary path $P_1$ (a-b) using the wavelength $\lambda_1$ and the backup path $B_1$ (a-c-b) reserving $\lambda_1$. Consider an incoming demand $d_2$ and the backup channel $\lambda_{(ac),1}$ in link (ac). We choose $P_2$ (a-d-e) as $d_2$’s primary path, which is included in $\lambda_{(ac),1}$’s supportable graph (the left graph of Figure 2.1). We say that, the link (ac) where $\lambda_{(ac),1}$ exists, can be used by the backup path for $d_2$ freely. In this example, $\lambda_{(ac),1}$ is assigned to $B_2$ indeed. The supportable graph of $\lambda_{(ac),1}$ after $d_2$ is satisfied is shown in the right graph of Figure 2.1. And its’ supported ratio change from $\frac{1}{6} (16.7\%)$ before establishing $d_2$ to $\frac{3}{6} (50.0\%)$ after establishing $d_2$. 
It is helpful to distinguish two types of links: *P-only* and *PB-only*. A link is called P-only if it has only primary channels. A link is PB-only if it has backup channels and primary channels (the number of primary channels could be 0), but no unused channels. Since both P-only and PB-only links have no unused channels, they cannot be used in any incoming demand's primary path. In addition, P-only links cannot be used in the backup path; however, a PB-only link could be used in a backup path if its primary path can be supported by a backup channel within the link.
CHAPTER 3  FAST RESTORATION METHOD

In this chapter, we present the 2-backup proactive restoration method that provides shorter restoration time than 1-backup method and achieves 100% restoration guarantee for dynamic traffic demands under single link failure assumption. First we introduce the key idea of the new 2-backup restoration method in section 3.1. Then in section 3.2 we analysis its restoration time and capacity requirement in comparison with 1-backup method. We present simulation results and evaluate the performance of the proposed method in section 3.3. Finally, we summarize this restoration method in section 3.4.

3.1 Key Idea

Given a traffic demand between source $s$ and destination $d$, we compute one primary lightpath and two backup lightpaths, with each backup lightpath protecting a part of the primary lightpath. An example is shown in Figure 3.1. The primary lightpath between $s$ and $d$ is divided into two segments $P_1$ and $P_2$ by the node $m$ in the middle of the primary lightpath. $P_1$ has a link-disjoint backup lightpath $B_1$ and $P_2$ has a link-disjoint backup lightpath $B_2$ (though $P_1$ and $B_2$ are not necessarily link-disjoint; neither are $P_2$ and $B_1$). Under normal condition, traffic is carried along the primary lightpath. Upon a link failure, one of the two backup lightpaths is activated based on the location of the failure: If a link fails on $P_1$, $B_1$ is activated and traffic is restored over $B_1$ and then continues on $P_2$. If a link fails on $P_2$, $B_2$ is activated and traffic first follows $P_1$ and then is restored over $B_2$. This is a failure dependent method as opposed to the failure independent 1-backup method since two different end-to-end restoration routes exist for the primary lightpath (i.e., $B_1\cdot P_2$ and $P_1\cdot B_2$) and the choice of which one to use depends on the location of the failure.
3.2 Performance Analysis

3.2.1 Restoration Time

Restoration time is the elapsed time from the instant a failure occurs to the instant the backup path is activated. The major advantage of the 2-backup method is that it leads to much faster restoration than 1-backup method.

3.2.1.1 Restoration Time of 1-backup Method

In 1-backup method, each primary lightpath has a single end-to-end backup lightpath. Two phases are involved in setting up the backup lightpath upon a failure.

- **PHASE 1** — Failure Notification: Upon a link failure, the end nodes of the failed link detect the failure and the node that is closer to the destination sends a failure notification message to the destination along the primary path. Let $h_n$ be the number of hops the failure notification message needs to traverse to reach the destination. Assuming each link on the primary path fails with equal probability, then on the average $h_n = h_p/2$ where $h_p$ is the length (in terms of number of hops) of the primary path.

- **PHASE 2** — Backup Path Activation: When the destination receives the failure notification message, it sends a setup message to the source node along the backup path. Upon receiving the setup message, each intermediate node along the path configures its cross-connects and passes the message onto the next node. Let $h_s$ be the number of hops the setup message needs to traverse to reach the source, then $h_s = h_b$ where $h_b$ is the
length of the backup path. After the source receives the setup message, it begins to send the disrupted traffic along the backup path, completing the restoration process.

The restoration time is the sum of the time taken in the two phases, and it is proportional to \( h_n \) and \( h_s \).

### 3.2.1.2 Restoration Time of 2-backup Method

Next we show that 2-backup method reduces both \( h_n \) and \( h_s \) and therefore reduces the restoration time. Similar as 1-backup method, two phases are involved in the 2-backup method.

- **PHASE 1** — Failure Notification: Using the example in Figure 3.1, if the failure occurs in \( P_1 \), the failure notification message will be sent to \( m \). If the failure occurs in \( P_2 \), the failure notification message will be sent to \( d \). On the average, the failure notification message traverses \( h_p/4 \) hops, thus \( h_n = h_p/4 \).

- **PHASE 2** — Backup Path Activation: If the failure occurs in \( P_1 \), node \( m \) will receive the failure notification message and send a setup message back to source \( s \) along backup path \( B_1 \). If the failure occurs in \( P_2 \), destination \( d \) will receive the failure notification message and send a setup message back to \( m \) along backup path \( B_2 \). In both cases, the nodes along the backup path will configure their cross-connects and forward the setup message to the next hop. Let \( h_{b1} \) be the length of \( B_1 \) and \( h_{b2} \) be the length of \( B_2 \). On the average, the setup message will traverse \( (h_{b1} + h_{b2})/2 \) hops, thus \( h_s = (h_{b1} + h_{b2})/2 \). Once \( s \) (or \( m \)) receives the setup message, it begins to send the traffic along the backup path \( B_1 \) (or \( B_2 \)), completing the restoration process.

Comparing \( h_n \) and \( h_s \) values of 1-backup and 2-backup methods, we can easily find that, in phase 1, 2-backup method reduces the failure notification time by half; in phase 2, it's conceivable that 2-backup method takes less time to activate the backup path because in general the average length of the two backup paths (each protecting half of the primary path) is less than the length of a single end-to-end backup path. The simulation results in section 3.3 will show that \( h_s \) value of 2-backup method can be as low as 60% of \( h_s \) in 1-backup method in
a network with average nodal degree 5. Thus, by reducing the time taken in both of the two phases, 2-backup method achieves much faster restoration.

3.2.2 Routing and Wavelength Assignment Algorithm

In this section, we describe our routing and wavelength assignment algorithm that computes a primary lightpath and two backup lightpaths for a given traffic demand and assigns wavelength channels to these three paths. The algorithm solves the routing and wavelength assignment problem in two separate steps. The routing algorithm uses Dijkstra’s algorithm to compute a primary and two backup paths for the given demand. The wavelength assignment algorithm assigns wavelength channels to the primary and backup paths in a way that tries to share backup wavelength channels among the current demand and the existing demands as much as possible. Because we assume that the network nodes have wavelength conversion capabilities, a lightpath is allowed to occupy different wavelength channels along its route.

3.2.2.1 Routing

Given a demand between source $s$ and destination $d$, we first find the shortest path $P$ between $s$ and $d$ as the primary path. Suppose $P$ is $h$ hops long, we then find a node $m$ on $P$ that divides $P$ into two segments $P_1$ and $P_2$, where $P_1$ is $\lceil h/2 \rceil$ hops long and $P_2$ is $\lfloor h/2 \rfloor$ hops long. Next we find the shortest path between $s$ and $m$ that is link-disjoint with $P_1$ as the backup path for $P_1$ and find the shortest path between $m$ and $d$ that is link-disjoint with $P_2$ as the backup path for $P_2$.

3.2.2.2 Wavelength Assignment With Backup Multiplexing

After routing is done, wavelength channels are assigned to the primary path and two backup paths. For the primary path, we assign one unused channel to each link along the path. For the two backup paths, we try to reuse backup channels (i.e., use backup multiplexing). In order to check the possibility of backup multiplexing, for each backup channel $\lambda$ on link $l$, the algorithm maintains a set $S(\lambda)$ containing all primary segments that use $\lambda$ on their backup
paths. Note that every two primary segments in $S(\lambda)$ must be link-disjoint; otherwise they can't share the same backup channel $\lambda$. Given a backup path $B$ of a primary segment $P$, the algorithm assigns wavelength channels to $B$ as follows. For each link $l$ on $B$, we check the backup channels within link $l$ one by one. Let $\lambda$ be the backup channel currently being checked, if all the primary segments in $S(\lambda)$ are link-disjoint with $P$, assign $\lambda$ to $B$. If none of backup channels can be assigned to $B$, i.e., sharing backup channels with existing backup paths is not possible, try to assign a unused channel to $B$. Such computation is equivalent to the maintenance of each backup channel’s supportable graph. The pseudo-code for the backup wavelength assignment algorithm is given below:

PROCEDURE Backup-Wavelength-Assignment($P, B$)
/* INPUT: $P$ - a primary segment, $B$ - the backup path for $P$ */
/* OUTPUT: wavelength assignments to $B$ */
1 FOR each link $l$ along $B$
2 FOR each backup channel $\lambda$ in $l$
3 IF $P$ is contained in $\lambda$'s supportable graph, THEN assign $\lambda$ to $B$ and GOTO 1
4 IF an unused channel exists in $l$, THEN assign it to $B$, ELSE block

Note that our algorithm handles the routing and wavelength assignment in two separate steps. To find the optimal routing of primary and backup paths that use the minimum number of free wavelength channels, routing and wavelength assignment have to be solved jointly and the problem is NP-hard [7].

3.2.3 Capacity Requirement

It has been shown in section 3.2.1 that 2-backup method reduces the restoration time by reducing the number of hops the failure notification message and the backup path setup message need to traverse. It is natural to ask whether this reduction in restoration time is achieved at the price of requiring more capacity. The intuitive answer is yes since in 2-backup method each demand has two backup paths and the total length of the two backup paths must be longer than or equal to the length of a single end-to-end backup path. But a careful observation shows that in some cases, 2-backup method requires less spare capacity than 1-backup method.
An example is given in Figure 3.2. Suppose there are two primary paths $P_1$ (a-b-c-d-e) and $P_2$ (a-i-c-d-e). Both have the same middle point $c$, which divides each of the them into two segments. Since segment (a-b-c) of $P_1$ and segment (a-i-c) of $P_2$ are link-disjoint, they can share wavelength channels on backup path (a-f-c), i.e., only one wavelength channel needs to be reserved on link (af) and (fc). On the other hand, segment (c-d-e) of $P_1$ and segment (c-d-e) of $P_2$ are not link-disjoint, they can use the same backup path (c-h-e), but two wavelength channels must be reserved on link (ch) and (he). Therefore, in 2-backup method, one backup channel is reserved on link (af) and (fc), two backup channels are reserved on link (ch) and (he), a total of 6 backup channels are reserved for the two primary paths $P_1$ and $P_2$. Now suppose 1-backup method is used and both $P_1$ and $P_2$ use (a-f-g-h-e) as their backup path. Since $P_1$ and $P_2$ are not link-disjoint, two backup channels must be reserved on the backup path (a-f-g-h-e), leading to a total of 8 backup channels used. This example shows that the spare capacity required in 2-backup method is not necessarily larger than in 1-backup method.

The reason is that in 2-backup method, backup multiplexing is done among primary segments instead of among the entire primary paths. That is, as long as two primary segments are link-disjoint, they can share backup channels, even though the two corresponding primary
paths are not link-disjoint. Since a primary segment is only half as long as the primary path, the chance of two primary segments being disjoint is higher than the chance of two primary paths being disjoint, resulting in a greater chance of sharing. On the other hand, the total length of the two backup paths for a given demand in 2-backup method is typically greater than a single end-to-end backup path, leading to larger capacity requirement. The combination of the two effects (i.e., higher chance of backup sharing and longer backup paths) on the capacity requirement is studied by simulation in the next section. The results show that for a large number of random demands, 2-backup method does require more capacity than 1-backup method, but the increase in capacity requirement is small.

### 3.3 Numerical Results

We compare the performance of 2-backup method with that of 1-backup method in this section. Three different networks are used to carry out the simulation. Their characteristics including number of nodes, number of links, average nodal degree and network diameter are given in Table 3.1. Network 1 and 3 are test networks used in [8] and [9] respectively. Network 2 is an Italian network given in [10]. For reference, Network 1 is illustrated in Figure 3.3. Each simulation run has 1000 demands (with a random source and a random destination) generated one by one. Link capacities are set to infinity so that no demand is rejected.

<table>
<thead>
<tr>
<th>Network</th>
<th>Nodes</th>
<th>Links</th>
<th>Avg Nodal Degree</th>
<th>Diameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32</td>
<td>51</td>
<td>3.19</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>70</td>
<td>4.38</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>125</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>

Two metrics are used to measure the performance of each method: total capacity used and average backup path length (measured by hop count). Total capacity used is the number of primary channels and backup channels. Let $C_1$ and $C_2$ denote the total capacity used by 1-backup method and 2-backup method respectively. Average backup path length is the average length of the backup paths for the 1000 demands in each simulation run. Let $L_1$ and $L_2$ denote the average backup path length for 1-backup method and 2-backup method respectively. For
Figure 3.3 Test Network 1

1-backup method,

\[ L_1 = \frac{\sum_{i=1}^{1000} b_i}{1000} \]

where \( b_i \) is the length of the backup path of demand \( i \). For the 2-backup method, the backup path length of a demand is the average length of its two backup paths because upon a failure only one of the backup paths is activated. Thus,

\[ L_2 = \frac{\sum_{i=1}^{1000} \frac{b_{i1} + b_{i2}}{2}}{1000} \]

where \( b_{i1} \) and \( b_{i2} \) are the length of the two backup paths for demand \( i \).

In each simulation run, 1000 demands are loaded to the test network, \( C_1, C_2, L_1 \) and \( L_2 \) values are then determined to compare the performance of 1-backup and 2-backup methods. For the 2-backup method, routing and wavelength assignment are done for each demand as explained in section 3.2.2. For the 1-backup method, routing and wavelength assignment algorithms are the same as 2-backup method except that a single end-to-end shortest path that is link-disjoint with the primary path is chosen to be the backup path for each demand.

It's clear that 2-backup method can be used only when the primary path is at least 2 hops long. And intuitively, the longer the primary path is, the better is the performance of 2-backup method. It can be explained as follows: as the primary path becomes shorter, the choice of
the two backup paths for a demand is more localized, thus increasing the average backup path length and capacity usage relative to 1-backup method. To evaluate the effect of primary path length on the two performance metrics, we conduct simulations for different minimum primary path lengths for each test network. In a simulation with minimum primary path length equal to \( k \), all the demands loaded to the network have primary paths at least \( k \) hops long.

Table 3.2 shows the simulation results for Network 1. The first column gives the minimum primary path length. The next three columns show the total capacity used for 1-backup method and 2-backup method respectively, and then percentage increase of the total capacity used by 2-backup method relative to 1-backup method. The last three columns give the average backup path length for 1-backup method and 2-backup method respectively, and then, the percentage decrease of the average backup path length for the 2-backup method compared to the 1-backup method. We can find that, 2-backup method always uses more capacity than 1-backup method, and the percentage increase in capacity requirement varies from 12.55% to 14.42%. In addition, 2-backup method has shorter average backup path length, and the percentage decrease relative to 1-backup method varies from 26.7% to 30.7%.

<table>
<thead>
<tr>
<th>Min Primary Hop</th>
<th>( C_1 )</th>
<th>( C_2 )</th>
<th>( \frac{C_1 - C_2}{C_1} \times 100 )</th>
<th>( L_1 )</th>
<th>( L_2 )</th>
<th>( \frac{L_1 - L_2}{L_1} \times 100 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6361</td>
<td>7278</td>
<td>14.42</td>
<td>5.32</td>
<td>3.90</td>
<td>26.7</td>
</tr>
<tr>
<td>3</td>
<td>7085</td>
<td>7974</td>
<td>12.55</td>
<td>5.81</td>
<td>4.07</td>
<td>29.9</td>
</tr>
<tr>
<td>4</td>
<td>8086</td>
<td>9125</td>
<td>12.85</td>
<td>6.28</td>
<td>4.35</td>
<td>30.7</td>
</tr>
<tr>
<td>5</td>
<td>9036</td>
<td>10171</td>
<td>12.56</td>
<td>6.72</td>
<td>4.66</td>
<td>30.7</td>
</tr>
<tr>
<td>6</td>
<td>10381</td>
<td>11757</td>
<td>13.26</td>
<td>7.30</td>
<td>5.08</td>
<td>30.4</td>
</tr>
</tbody>
</table>

Table 3.3 shows the simulation results for Network 2. Same as the results for Network 1, 2-backup method uses more capacity and has shorter average backup path length than 1-backup method. The percentage increase in total capacity used varies from 4.28% to 8.01% and the percentage decrease in average backup path length varies from 34.4% to 41.0%. The advantage of 2-backup method over 1-backup method is more dramatic in Network 2 since it has larger average backup path decrease with smaller capacity increase compared to Network 1. The reason for this performance improvement is that Network 2 is denser (i.e., has higher
average nodal degree) than Network 1, thus the backup paths tend to be shorter and use lesser capacity.

Table 3.3 Total Capacity Used and Average Backup Path Length in Network 2

<table>
<thead>
<tr>
<th>Min Primary Hop</th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$\frac{C_2-C_1}{C_1} \times 100$</th>
<th>$L_1$</th>
<th>$L_2$</th>
<th>$\frac{L_2-L_1}{L_1} \times 100$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5906</td>
<td>6379</td>
<td>8.01</td>
<td>4.25</td>
<td>2.79</td>
<td>34.4</td>
</tr>
<tr>
<td>3</td>
<td>7130</td>
<td>7559</td>
<td>6.02</td>
<td>4.83</td>
<td>3.02</td>
<td>37.5</td>
</tr>
<tr>
<td>4</td>
<td>9199</td>
<td>9617</td>
<td>4.54</td>
<td>5.71</td>
<td>3.40</td>
<td>40.5</td>
</tr>
<tr>
<td>5</td>
<td>10744</td>
<td>11204</td>
<td>4.28</td>
<td>6.36</td>
<td>3.75</td>
<td>41.0</td>
</tr>
</tbody>
</table>

Table 3.4 shows the simulation results for Network 3. The percentage increase in total capacity used varies from 2.17\% to 4.61\% and the percentage decrease in average backup path length varies from 38.5\% to 42.7\%. Among the 3 test networks, Network 3 demonstrates the greatest advantage of 2-backup method over 1-backup method, i.e., it achieves the biggest backup path length decrease at the cost of the smallest capacity increase. It is as the result of that Network 3 is the densest among the three test networks.

Table 3.2 — 3.4 also show that as the minimum primary path length increases, the percentage decrease in average backup path length becomes larger and the percentage increase in capacity used becomes smaller. Therefore, 2-backup method is more efficient in decreasing restoration time when the primary paths are longer, conforming to our intuition.

In brief, 2-backup method reduces the average backup path length (therefore reducing the restoration time) at the cost of increasing the total capacity used. As the network becomes denser, the reduction in average backup path length becomes greater and the increase in capacity used becomes smaller. Therefore, 2-backup method can dramatically decrease the

Table 3.4 Total Capacity Used and Average Backup Path Length in Network 3

<table>
<thead>
<tr>
<th>Min Primary Hop</th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$\frac{C_2-C_1}{C_1} \times 100$</th>
<th>$L_1$</th>
<th>$L_2$</th>
<th>$\frac{L_2-L_1}{L_1} \times 100$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6826</td>
<td>7130</td>
<td>4.53</td>
<td>4.80</td>
<td>2.95</td>
<td>38.5</td>
</tr>
<tr>
<td>3</td>
<td>7508</td>
<td>7840</td>
<td>4.42</td>
<td>5.28</td>
<td>3.13</td>
<td>40.7</td>
</tr>
<tr>
<td>4</td>
<td>8703</td>
<td>9054</td>
<td>4.03</td>
<td>5.95</td>
<td>3.43</td>
<td>42.4</td>
</tr>
<tr>
<td>5</td>
<td>9776</td>
<td>9988</td>
<td>2.17</td>
<td>6.44</td>
<td>3.69</td>
<td>42.7</td>
</tr>
<tr>
<td>6</td>
<td>11075</td>
<td>11586</td>
<td>4.61</td>
<td>7.17</td>
<td>4.13</td>
<td>42.4</td>
</tr>
</tbody>
</table>
service disruption time with only a minor extra requirement in capacity.

3.4 Summary

Current path-based proactive lightpath restoration method finds one primary lightpath and one backup lightpath to provide 100% resoration guarantee for each demand. Backup multiplexing is used to reduce the spare capacity requirement by allowing backup lightpaths to share wavelength channels provided that their primary lightpaths are link-disjoint. However, the restoration time is long due to the end-to-end signaling required to activate the backup lightpath upon failure. In this chapter, we have proposed a new proactive lightpath restoration method to achieve faster restoration, where a primary lightpath is protected by two backup lightpaths, each protecting one segment of the primary lightpath. Depending on the location of the failure, one of the two backup paths will be activated to restore the traffic. The new method achieves faster restoration by reducing the time spent on both failure notification and backup path activation. We have presented a routing and wavelength assignment algorithm for establishing a primary and two backup lightpaths for a given demand. Simulations are conducted on three test networks to compare the new 2-backup method with 1-backup method. The results showed that 2-backup method can significantly reduce the restoration time with only minor increase in the capacity requirement. It's also shown that 2-backup method achieves better performance as the network topology becomes denser and the primary paths become longer.
CHAPTER 4 EFFICIENT LIGHTPATH ESTABLISHMENT

To approach the second question, we propose an efficient distributed routing algorithm for dynamic lightpath establishment in this chapter, which is organized as follows. Section 4.1 presents two algorithms used for comparison with our distributed routing algorithm and briefly reviews similar works. In section 4.2, we discuss the load balancing heuristics, including random version of Dijkstra’s algorithm, and link cost functions for both primary and backup path selection. The control mechanism for this algorithm is also presented. We study the impact of each load balancing heuristic and give the simulation results in section 4.3, and a conclusion remark is given in section 4.4.

4.1 Previous Work

We first present two existing dynamic routing algorithms which will be compared with our algorithm later in the simulation. Other related works using wavelength usage information in the path selection are introduced next. At the end of this section a classification of reasons for demand blocking is given.

4.1.1 Centralized Algorithm (CA)

In centralized scenario, when a demand arrives at a network node, the node sends the demand information to a network management system (NMS) [11], which in turn runs a centralized routing algorithm to compute a primary-backup path pair for the demand. The NMS has a complete view of the network, including channel types (i.e., primary, backup or unused) in each link and the supportable graph of each backup channel.

A centralized routing algorithm is proposed in [12]. During primary path selection, all links
without unused channels are set with the cost of infinity, the other links are set with the cost of 1. Dijkstra's algorithm is then run on this weighted graph. After choosing the shortest path as the primary path, the algorithm removes all links along the primary path and P-only links, and set the cost of the remaining links as follows: if a link contains a backup channel that can support the primary path (by checking it's supportable graphs), set the link cost $\alpha$, where $0 \leq \alpha < 1$; otherwise, if the link contains an unused channel, set the cost 1, else set the cost infinity (which means the link is PB-only and all backup channels can not support the chosen primary path). Then the shortest path in this weighted graph is chosen as the backup path. Here $\alpha$ provides a trade-off between capacity efficiency and backup path length [12]. When $\alpha$ is set to 0, the backup path selection is to use "free" links as possible as it can, which leads to larger hop count and longer restoration latency. In the following simulation, we let $\alpha$ be 0.5.

The algorithm adopts first-fit wavelength assignment when signaling along the backup path. That is, the first backup channel that can support the primary path is assigned to the backup path; If no such backup channel exists, assign the first unused channel to the backup path. Note that first-fit wavelength assignment is used by all algorithms discussed in the chapter.

4.1.2 Simple Distributed Algorithm (SDA)

In distributed scenario, the source node is responsible for computing the primary and backup path for arriving demands. Since it's not feasible for each node to maintain the complete view of the network status as the centralized algorithm does, each node only maintains a limited network status information and uses these information for path selection. In the simple algorithm presented here, each node only knows whether each link has unused channels. The primary path selection is the same as in centralized algorithm. During the backup path selection, the algorithm excludes links along the chosen primary path, P-only links as well as PB-only links, then sets the cost of each remaining link to 1. The shortest path in the weighted graph is chosen to be the backup path.

As mentioned in previous subsection, first-fit wavelength assignment scheme is also used here to try to share those backup channels first. Note that if P-only links are excluded instead
of both P-only and PB-only links during the backup path selection, the backup path may be blocked during the setup signaling. This occurs when any link in the chosen path is PB-only but none of backup channels can support the primary path. However, there is a trade-off on the blocking probability because sometimes, we can not find a backup lightpath connecting the source and destination nodes without considering those PB-only links, especially when the traffic becomes heavier.

4.1.3 Other Related Work

The work in [13] proposed a concept of the Least Congested Path (LCP) routing, which basically is a min-max computation. The number of unused channels in a link determines the degree of congestion of the link. When a demand arrives, the numbers of unused channels of all links in all feasible paths for this demand are collected. Then the minimum value of the numbers of unused channels of links along each path is computed and the path with the maximum of such numbers is chosen as the least congested path to accommodate the demand. However the paper did not consider the backup path selection. Besides, it takes a relatively long time to enumerate feasible paths for each demand and compute the min-max value.

Distributed routing algorithm proposed in [14] also uses information about the number of unused and backup channels to guide path selection. To integrate the sharability of a backup channel into the link cost estimation for backup paths, a probabilistic link cost model is proposed. The OXC node locally estimates the "sharability" for each adjacent link, based on which, it's link cost for backup path selection is computed and disseminated to all other OXCs. A link's sharability is evaluated as the probability that the link contains a backup channel that can support any primary path. At first a random pair sample is chosen. For each pair in the sample, a set of shortest cost path are computed as potential primary paths, and the fraction of these paths which the link can support is determined using the local information. The average is then taken over all pairs in the sample to obtain the possibility that this link contains a sharable backup channel. As mentioned in their simulation results, this model does not lead to significant savings in restoration capacity. What is more, the computation complexity is
prohibitive when a large sample is applied.

4.1.4 Blocking Types

A demand request may be blocked due to two reasons. One is due to resource limitation, the other is due to algorithm. The former happens when between the source and destination nodes there is no feasible path to provide primary path or backup path because the available network resource can’t accommodate two link-disjoint paths. In this case, the demand must be blocked no matter what algorithm is used. Block due to algorithm happens when the routing algorithm chooses an infeasible backup path even though a feasible backup path does exist in the network. It is mainly caused by improper cost assignment to the PB-only link (as explained in SDA). Note that retry scheme [15] could relieve blocking due to algorithm but with longer response latency.

4.2 Load Balancing Heuristics

In this section, three load balancing heuristics that can be adopted by dynamic routing algorithms are presented. Our goal is to use link channels evenly, leading to low blocking probability for demands. We first discuss the effects of different implementations of Dijkstra’s Algorithm. Then we describe our link cost assignment strategies for primary path selection and backup path selection. Finally the control mechanism for lightpath establishment is discussed and the information disseminated for path selections is also noted.

4.2.1 Random Version of Dijkstra’s Algorithm

Routing modules in some Internet protocols, such as OSPF, use Dijkstra’s algorithm [16] to compute one shortest path from the source node to the destination node. In this paper, we adopt Dijkstra’s algorithm as well.

Given a source-destination node pair and a weighted graph, ordinary Dijkstra’s algorithm always provides a fixed path; this is due to the invariant order of extracting to-be-relaxed nodes from a priority queue (implemented by array or heap). However, given a unit-weighted graph,
it is often the case that more than 2 shortest paths exist between a pair, particularly when the network graph becomes denser. For simplicity, suppose that we only consider providing primary lightpaths for each demand, and there are 2 shortest paths for a given pair. After several requests for the same pair are satisfied, the first fixed shortest path can not be chosen because of no unused channels, so the other shortest path will be selected. But it may bring inefficiency to establish connections starting from or ending in those nodes involved in the first shortest path.

We proposed a *random* version of Dijkstra’s algorithm, which is to randomly choose one of those nodes with minimum shortest-path estimate in the priority queue to make relaxation in each round. This allows us to make use of all possible shortest paths evenly as the primary or backup lightpath for demands with the same source and destination. Figure 4.1 shows 2 cases using original version and random version respectively to compute primary paths only in the unit-weighted network graph. Assume each link has 2 wavelength channels. The incoming

![Figure 4.1 Original and Random Version of Dijkstra's Algorithm](image)

 demands are in order of \((a, d), (a, d), (c, d)\). Suppose the original version always chooses path \((a-b-c-d)\) for the first 2 demands. When dealing with the third demand \((c, d)\), the path \((c-f-d)\) has to be chosen since link \((cd)\) has no unused channel. However, random version may choose \((a-b-c-d)\) and \((a-c-f-d)\) as the first 2 demands’ primary paths, therefore allowing the third demand to use \((c-d)\) as its primary path. Thus one channel is saved in the later case.

Note that though certain improvement can be made, it is difficult to influence path selection
explicitly by such random factor. Nevertheless, we can construct judicious link cost function, as explained in the following two subsections, which would be more effective in controlling path selection.

4.2.2 Primary Path Selection

The basic rule of primary path selection is to follow the shortest path as in the unit-weighted graph as long as every link along it has enough unused channels. When some links have unused channels less than a threshold, these links become "critical" and we take precautionary step to avoid using them. Accordingly, the chosen path will be a little longer than that chosen in the same graph with unit weight. Unlike the backup channel, which may be shared with other demand, primary channel can never be shared with future demands. Thus a higher weight is assigned to a link if the link has no more than \( \delta \) unused channels, where \( \delta \) is a constant (called critical index). Note that those links with no unused channels should be excluded during the primary path selection.

Formally, given a link \( l \), we define its link cost for primary path selection as follows:

\[
\text{pcost}(l) = \begin{cases} 
\infty & \text{if } l \text{ has no unused channels} \\
\eta & \text{if } l \text{ has } \leq \delta \text{ but } > 0 \text{ unused channels} \\
1 & \text{if } l \text{ has } > \delta \text{ unused channels}
\end{cases}
\]

where \( \eta \) is a constant and \( \eta > 1 \). Dijkstra's algorithm can then be run on the weighted graph to compute the shortest path as the primary path. When \( \delta = 0 \), the primary path selection becomes the same as that in the CA and SDA.

4.2.3 Backup Path Selection

Like the primary path selection, we use the number of unused wavelength channels to guide the backup path search. In addition, we use the information about backup channels. Since P-only links and those links along the chosen primary path can not be used by the backup path, the cost of these links are set to infinity.
One observation we get from simulation is that a backup channel's supported ratio is about 30% on average under the dynamic traffic, though the maximal value could reach 90%. In our scheme, those backup channels with supported ratio larger than \( \nu \) (called venture index) are considered not able to support more primary paths, therefore is not considered as a backup channel candidate for new demands. Within a link, those backup channels with supported ratio less than \( \nu \) and those unused channels comprise this link's backup channel candidates for new demands. Intuitively, the backup path should prefer links with more backup channel candidates to those with less backup channel candidates. So our link cost function is the reciprocal of the number of the link's backup channel candidates.

Formally, given a link \( l \), we denote \( \text{cand}(l) \) and \( \text{bcost}(l) \) as the number of the backup channel candidates and the link cost used for backup path selection, then \( \text{bcost}(l) \) is defined as follows:

\[
\text{bcost}(l) = \begin{cases} 
\infty & \text{if } \text{cand}(l) = 0 \text{ or } l \text{ is in the primary path} \\
\frac{1}{\text{cand}(l)^r} & \text{otherwise}
\end{cases}
\]

where \( r \) could be any number no less than 1. In our simulation, we use 1, 2, 3, 4, 5 as \( r \)'s values.

The reason we introduce the exponent \( r \) is that when \( r = 1 \) the difference between two link costs would be too small to reflect the preference in choosing backup lightpath, even though a sizable gap exists between the backup channel candidate numbers of these two links. For example, see Figure 4.2, which is part of a network with 16 wavelengths per link. We try to find the path for demand \((s, d)\), which will passes through \( s' \) and \( d' \). Suppose link \( l_1 \) currently has 7 unused channels and 2 backup channels with supported ratio less than \( \nu \), then the number of backup channel candidates \( \text{cand}(l_1) \) is 9. The channel status of \( l_2 \) is the same as of \( l_1 \) (i.e., \( \text{cand}(l_2) = 9 \)). As for link \( l_3 \), \( \text{cand}(l_3) = 5 \) but only 3 unused channels left. When \( r = 1 \), \( \text{bcost} \) values of \( l_1, l_2 \) and \( l_3 \) are \( \frac{1}{9}(0.111), \frac{1}{9}(0.111) \) and \( \frac{1}{5}(0.200) \) respectively; the path selection algorithm will choose link \( l_3 \). Consequently \( l_3 \) will only have 2 unused channels (or still 3 thanks to the existence of a supportable backup channel). When \( r = 2 \), the 3 \( \text{bcost} \) values are 0.012, 0.012 and 0.040; then \( l_1 \) and \( l_2 \) will be chosen to connect \( s' \) and \( d' \). In this case, \( l_3 \) remains 3 unused channels, \( l_1 \) and \( l_2 \) may have 6 or 7 unused channels, achieving better
balanced wavelength consumption.

Figure 4.2  r's Effect on Backup Path Selection

4.2.4 Control Mechanism

Multi-protocol Lambda Switching (MPλS)[17], extended from MPLS, is now concentrating on the lightpath establishment without restoration functionality. [18] examines challenges faced in achieving restorable lightpath under current MPλS framework and presents where particular MPλS enhancements for control mechanism are needed. Here we introduce the lightpath establishment procedure briefly.

For the primary lightpath setup, a reservation message is sent along the primary path. Each intermediate node examines the requested resources and reserves one unused channel. After the destination receives the reservation message, it will send an acknowledgment message back to previous node. Each intermediate node then configures the switch on receiving it, and then forward the message to its previous node. Concurrently, similar work is done for the backup lightpath setup, except that for each link, first-fit wavelength assignment is used to choose a backup channel or an unused channel for the backup lightpath. Note that the primary path information is included in the reservation message for backup lightpath setup. If one backup channel is chosen, the primary path is added to the channel’s supported graph. If an unused channel is chosen, the channel status becomes “backup” with its supported graph containing only the primary path. Besides, the switches need not be configured at this time. When a failure occurs, the restoration activation architecture presented in [19] can be applied here and we omit the discussion on it.
Network topology and wavelength usage information (including the numbers of 3 types of channels, supported ratios of backup channels) can be disseminated to all OXC nodes in the network using an augmented interior gateway protocol (IGP) with appropriate extensions (such as [20]) to its link state advertisement (LSA) messages. With these local information, the routing module in the source node can determine primary and backup lightpaths, as presented in the previous 2 subsections.

4.3 Numerical Results

We evaluate each load balancing idea on a network of 32 nodes and 51 links, which is studied in [8], and also used in previous work (see Figure 3.3). We assume that each link has 16 wavelengths. Although not shown, similar results are also obtained with 8 wavelengths in each link.

The traffic pattern is dynamic and is uniformly distributed among all node pairs. The arrival of traffic to the whole network follows Poisson distribution with \( \lambda \) demand requests per second and the holding time for a demand is exponentially distributed with the mean of \( 1/\mu \). The traffic load of the network measured in Erlang is \( \lambda/\mu \). The same set of 10000 demands is run for each test case. This network model and traffic model are applied to all simulations, except for the first one on Dijkstra’s algorithm.

4.3.1 On Dijkstra’s Algorithm

The first simulation is run on the test network with unlimited wavelength. The traffic is incremental, that is, once a demand request is satisfied, it will remain active forever. Also the traffic is uniformly distributed among all node pairs. The numbers of used channels (including primary and backup channels) of all links are recorded after each run by the simple distributed algorithm (SDA), which is equipped with array version or random version of Dijkstra’s algorithm for the shortest path computation. We compare the maximum and minimum channels used in one link with array version and random version. They are 462 and 17 with array version whereas 457 and 58 with random version. We also compare the standard deviations
among used channels of each node's adjacent links. 21 out of 32 nodes have lower deviation with random version. Both results show that link channels across the network are used more evenly.

Table 4.1 lists the blocked demand numbers by SDA equipped with 3 versions of Dijkstra's Algorithm. Note that the traffic model and network model are described at the beginning of this section. The result of random version is computed on average with 20 runs. As we expected, random version causes less blocked demand number than array version. But the heap version outperforms both random version and array version. It may be attributable to dynamic features of binary tree embedded in the heap.

<table>
<thead>
<tr>
<th>Load</th>
<th>40</th>
<th>50</th>
<th>53</th>
<th>57</th>
<th>60</th>
<th>63</th>
<th>65</th>
<th>67</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array Version</td>
<td>0</td>
<td>30</td>
<td>42</td>
<td>107</td>
<td>160</td>
<td>209</td>
<td>230</td>
<td>302</td>
<td>498</td>
<td>803</td>
<td>1534</td>
<td>1991</td>
</tr>
<tr>
<td>Random Version</td>
<td>3</td>
<td>19</td>
<td>26</td>
<td>91</td>
<td>129</td>
<td>184</td>
<td>223</td>
<td>284</td>
<td>474</td>
<td>793</td>
<td>1501</td>
<td>1952</td>
</tr>
<tr>
<td>Heap Version</td>
<td>1</td>
<td>11</td>
<td>23</td>
<td>88</td>
<td>128</td>
<td>142</td>
<td>201</td>
<td>269</td>
<td>466</td>
<td>767</td>
<td>1451</td>
<td>1941</td>
</tr>
</tbody>
</table>

Since the difference among the results corresponding to these 3 versions is insignificant and heap version has the lowest running time, we use heap version in the following experiments.

### 4.3.2 On Primary Path Selection

We compare the blocked demand numbers for different δ values and loads, as shown in Table 4.2. When a link's unused channel number reaches δ, the link cost is adjusted from 1 to 1.5 (i.e., \( \eta = 1.5 \)). To study the impact of our cost function for the primary path selection, the backup path selection procedure of SDA is employed to provide the backup path. Note that when δ is 0, the algorithm becomes original SDA.

It is clear that the blocked demand number is increasing along with the increasing traffic load and increasing much quickly under the heavy load. For a fixed load larger than 60, we find a tendency that the blocked demand number decreases and then increases as δ increases. Proper δ choice can lower 29% blocked demands on average compared with SDA (i.e., \( \delta=0 \)), and generally the value of δ = 3 or 4 makes the best improvement. Results also show that
Table 4.2 Blocked Demand Numbers With Different $\delta$

<table>
<thead>
<tr>
<th>Load</th>
<th>40</th>
<th>50</th>
<th>53</th>
<th>57</th>
<th>60</th>
<th>63</th>
<th>65</th>
<th>67</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
<th>120</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta = 0$/SDA</td>
<td>1</td>
<td>11</td>
<td>23</td>
<td>88</td>
<td>128</td>
<td>142</td>
<td>201</td>
<td>269</td>
<td>466</td>
<td>767</td>
<td>1451</td>
<td>1941</td>
<td>2616</td>
</tr>
<tr>
<td>$\delta = 1$</td>
<td>0</td>
<td>8</td>
<td>14</td>
<td>67</td>
<td>92</td>
<td>140</td>
<td>174</td>
<td>238</td>
<td>414</td>
<td>704</td>
<td>1350</td>
<td>1804</td>
<td>2545</td>
</tr>
<tr>
<td>$\delta = 2$</td>
<td>0</td>
<td>5</td>
<td>13</td>
<td>60</td>
<td>87</td>
<td>119</td>
<td>151</td>
<td>208</td>
<td>388</td>
<td>665</td>
<td>1265</td>
<td>1823</td>
<td>2574</td>
</tr>
<tr>
<td>$\delta = 3$</td>
<td>0</td>
<td>4</td>
<td>13</td>
<td>53</td>
<td>75</td>
<td>115</td>
<td>161</td>
<td>218</td>
<td>372</td>
<td>632</td>
<td>1340</td>
<td>1793</td>
<td>2579</td>
</tr>
<tr>
<td>$\delta = 4$</td>
<td>1</td>
<td>8</td>
<td>6</td>
<td>61</td>
<td>68</td>
<td>134</td>
<td>161</td>
<td>205</td>
<td>386</td>
<td>674</td>
<td>1331</td>
<td>1775</td>
<td>2689</td>
</tr>
<tr>
<td>$\delta = 5$</td>
<td>1</td>
<td>2</td>
<td>11</td>
<td>56</td>
<td>82</td>
<td>132</td>
<td>165</td>
<td>220</td>
<td>375</td>
<td>707</td>
<td>1371</td>
<td>1882</td>
<td>2634</td>
</tr>
<tr>
<td>$\delta = 6$</td>
<td>1</td>
<td>9</td>
<td>8</td>
<td>60</td>
<td>80</td>
<td>131</td>
<td>172</td>
<td>221</td>
<td>417</td>
<td>715</td>
<td>1437</td>
<td>1837</td>
<td>2674</td>
</tr>
</tbody>
</table>

Heavy traffic load asks for later precautionary action (i.e., smaller $\delta$ value). It can be explained as follows: later action allows most of primary paths taken as the shortest paths in the unit weight graph, saving the wavelength resources and thus accepting more subsequent demands.

### 4.3.3 On Backup Path Selection

Due to the space limitation, Table 4.3 only shows the blocked demand numbers for different $\nu$ values and loads in 2 cases, i.e., $r = 2$ and 3. (More results on different $r$ values can be found in the Appendix A). Here, in order to see the impact of our link cost function for backup path selection, we do not adopt the precautionary step during the primary path selection; in other words, set $\delta$ to be 0 for the primary path selection. For reference, the blocked demand numbers by CA and SDA are listed in the table as well. It should be mentioned that the primary path selections of the three algorithms are currently the same and the difference lies only in the backup path selection.

The numbers in bold stand for the smallest blocked demands with different $\nu$ values, given a fixed traffic load and $r$ value. The result confirms that larger $r$ value reduces the number of blocked demands with other sensible settings. But given a specified load and $\nu$ value, results with $r = 4$ and 5 (not shown in the table) do not bring considerable reduction on the blocked demand number.

It also verifies that overestimating the contribution of backup channels would result in worse performance. For example, when load is 50 and $\nu$ is only 20% (i.e., backup channels with supported ratio less than 20% are included in the candidate set), the algorithm ends with 31($r = 2$) or 18($r = 3$) blocked demands whereas only 11 demands are blocked by SDA.
Table 4.3 Blocked Demand Numbers With Different $r$ and $\nu$

<table>
<thead>
<tr>
<th>Load</th>
<th>40</th>
<th>50</th>
<th>53</th>
<th>57</th>
<th>60</th>
<th>63</th>
<th>65</th>
<th>67</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDA</td>
<td>1</td>
<td>11</td>
<td>23</td>
<td>128</td>
<td>142</td>
<td>201</td>
<td>269</td>
<td>466</td>
<td>767</td>
<td>145</td>
<td>194</td>
<td>183</td>
</tr>
<tr>
<td>CA</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>15</td>
<td>12</td>
<td>44</td>
<td>31</td>
<td>82</td>
<td>155</td>
<td>334</td>
<td>915</td>
<td>1393</td>
</tr>
</tbody>
</table>

$r = 2$

<table>
<thead>
<tr>
<th>$\nu$</th>
<th>0%</th>
<th>5%</th>
<th>10%</th>
<th>15%</th>
<th>20%</th>
<th>25%</th>
<th>30%</th>
<th>40%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>24</td>
<td>60</td>
<td>89</td>
<td>108</td>
<td>196</td>
</tr>
<tr>
<td>$\nu$</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>34</td>
<td>58</td>
<td>95</td>
<td>113</td>
<td>161</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>9</td>
<td>44</td>
<td>50</td>
<td>82</td>
<td>89</td>
<td>135</td>
<td>251</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>31</td>
<td>82</td>
<td>90</td>
<td>123</td>
<td>152</td>
<td>185</td>
<td>273</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>26</td>
<td>52</td>
<td>96</td>
<td>106</td>
<td>173</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>22</td>
<td>50</td>
<td>60</td>
<td>67</td>
<td>117</td>
<td>233</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>31</td>
<td>60</td>
<td>109</td>
<td>101</td>
<td>152</td>
<td>170</td>
<td>244</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>34</td>
<td>36</td>
<td>95</td>
<td>107</td>
<td>156</td>
<td>206</td>
<td>250</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>27</td>
<td>37</td>
<td>112</td>
<td>118</td>
<td>165</td>
<td>200</td>
<td>258</td>
</tr>
</tbody>
</table>

$r = 3$

<table>
<thead>
<tr>
<th>$\nu$</th>
<th>0%</th>
<th>5%</th>
<th>10%</th>
<th>15%</th>
<th>20%</th>
<th>25%</th>
<th>30%</th>
<th>40%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>29</td>
<td>59</td>
<td>100</td>
<td>82</td>
<td>177</td>
</tr>
<tr>
<td>$\nu$</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>26</td>
<td>52</td>
<td>96</td>
<td>106</td>
<td>173</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>22</td>
<td>50</td>
<td>60</td>
<td>67</td>
<td>117</td>
<td>233</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>11</td>
<td>44</td>
<td>44</td>
<td>79</td>
<td>91</td>
<td>139</td>
<td>230</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>18</td>
<td>53</td>
<td>75</td>
<td>90</td>
<td>97</td>
<td>160</td>
<td>238</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>17</td>
<td>27</td>
<td>70</td>
<td>78</td>
<td>121</td>
<td>121</td>
<td>169</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>22</td>
<td>23</td>
<td>69</td>
<td>88</td>
<td>130</td>
<td>146</td>
<td>181</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>22</td>
<td>30</td>
<td>89</td>
<td>106</td>
<td>154</td>
<td>194</td>
<td>219</td>
</tr>
</tbody>
</table>

When $\nu = 0$, the algorithm takes advantage of the number of unused channels only. Compared with SDA, certain improvement is also made, but is not good enough especially under heavy load. Besides, the blocked demand numbers in bold express a trend, that is, the cost estimation function should "venture" those backup channels with higher supported ratio as the traffic load increases, namely, a larger $\nu$ value is preferred.

4.3.4 Overall Comparison

In this section, we combine the primary path selection heuristic and the backup path selection heuristic together, called load balancing routing algorithm (LBA). LBA is compared with SDA and CA, and the blocked demand numbers by these 3 algorithms are shown in Table 4.4. In LBA, we choose $\delta = 3$ while computing primary path and $r = 4$ while computing backup path. For better performance of LBA, when the traffic load changes from range [40, 70], [80, 90] to 100, the venture index $\nu$ is adjusted from 10%, 20% to 30% correspondingly. The details can be found in the Appendix B.
Table 4.4  Blocked Demand Numbers By Each Algorithm

<table>
<thead>
<tr>
<th>Load</th>
<th>40</th>
<th>50</th>
<th>53</th>
<th>57</th>
<th>60</th>
<th>63</th>
<th>65</th>
<th>67</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDA</td>
<td>1</td>
<td>11</td>
<td>23</td>
<td>88</td>
<td>128</td>
<td>142</td>
<td>201</td>
<td>269</td>
<td>466</td>
<td>767</td>
<td>1451</td>
<td>1941</td>
</tr>
<tr>
<td>LBA</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>15</td>
<td>22</td>
<td>48</td>
<td>41</td>
<td>94</td>
<td>189</td>
<td>408</td>
<td>1020</td>
<td>1428</td>
</tr>
<tr>
<td>CA</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>15</td>
<td>12</td>
<td>44</td>
<td>31</td>
<td>82</td>
<td>155</td>
<td>334</td>
<td>915</td>
<td>1393</td>
</tr>
</tbody>
</table>

Under light traffic loads, LBA performs almost as well as CA does. The precautionary step in the primary path selection could be used to explain why LBA has no blocked demands while CA has 1 blocked demands under the loads 40 and 50. Within load range [65, 100], SDA causes 39%-548% more blocked demands compared with CA. However, the new algorithm LBA only adds 2%-32% more blocked demands, and the absolute blocked demand number is much closer to that by CA.

4.4  Summary

In this chapter, we present three load balancing ideas: random version of Dijkstra’s algorithm, the heuristic cost function for primary path selection and the heuristic cost function for backup path selection. Random version of Dijkstra’s algorithm running on the unit-weight network graph does not make significant improvement on lowering the blocked demand number. Our primary path selection characterizes the critical link by comparing the link’s unused channel number with the critical index, and assigns a higher weight to it. The number of unused channels is also used to guide backup path selection. Furthermore, the venture index is proposed to integrate the contribution of backup channels to the cost function for backup path selection.

Simulation results show that load balancing routing algorithm (LBA), including both heuristic cost functions, achieves almost the same performance as centralized algorithm (CA) under light traffic loads, and very close to it under heavy loads. However, relative to CA, LBA only need few information to be flooded in the network, thus the control traffic load is light, making the distributed algorithm more scalable. Besides, the computation for both paths in LBA is simple.

To achieve better performance, the venture index as well as the critical index need to be
adjusted with the changing load. One approach to achieve that is to let each node in the network can monitor the traffic and then disseminate the traffic information along with wavelength usage information in the link state advertisement messages. Another feasible approach is to make statistical analysis on daily traffic, based on which the two indexes can be adjusted.
SUMMARIES OF BOTH WORK ARE PRESENTED IN SECTION 3.4 AND SECTION 4.4. ALTHOUGH WE GIVE OUR SOLUTIONS TO THE TWO PROBLEMS AND BOTH SIMULATION RESULTS SHOW THAT BETTER PERFORMANCE IS ACHIEVED, THERE ARE STILL SOME IMPROVEMENTS TO BE MADE IN THE FUTURE WORK.

FIRST OF ALL, THE 2-BACKUP RESTORATION METHOD AND THE EFFICIENT LOAD BALANCING ROUTING ALGORITHM HANDLE 2 DIFFERENT ASPECTS OF THE PROBLEM OF DYNAMIC ESTABLISHMENT OF RESTORABLE LIGHTPATHS. THEY ARE 2 RELATIVELY INDEPENDENT APPROACHES, AND WE HAVE NOT INVESTIGATED THE EFFECT OF EMPLOYING THESE 2 SOLUTIONS TOGETHER ON RESTORABLE LIGHTPATH ESTABLISHMENT; HOWEVER THE COMBINATION COULD BE EXPECTED TO BE A GOOD SOLUTION WHEN BOTH THE RESTORATION TIME AND BLOCKING PROBABILITY ARE CRITICAL. NEXT WE DISCUSS SOME IDEAS FOR THE FUTURE WORK.

5.1 GENERALIZED SEGMENTED RESTORATION

THE 2-BACKUP RESTORATION METHOD, WHICH IS BETWEEN LINK-BASED AND PATH-BASED RESTORATION, PROVIDES A CERTAIN DEGREE OF QUALITY OF RESTORATION. THIS IDEA CAN BE EXTENDED WITH SEVERAL DIRECTIONS:

- To divide each primary path into several non-overlapping protection segment, to protect against single link failure only;

- To divide each primary path into several overlapping protection segments, to protect against single node failure as well;

- To route each primary path via one or more than one nodes in a preselected set. Only nodes in the set can be used as “middle points”. Nodes with high degree could be better candidates.
Basically, 2-backup restoration method is a special case of which using non-overlapping protection segments. Short Leap Shared Protection proposed in [21], did the second extension. Recently multi-segment protection approaches are gaining attention in the context of shared risk link groups [21, 22] and some other related research can be found in [23, 24].

5.2 More about Dijkstra’s Algorithm

When presenting the results of different versions of Dijkstra’s algorithm in section 4.3.1, we claim that the dynamic feature within heap version brings better performance on blocking probability than array version. Table 5.1 gives some shortest paths computed by array version and heap version based on Figure 3.3.

<table>
<thead>
<tr>
<th>(s, d)</th>
<th>array version</th>
<th>heap version</th>
</tr>
</thead>
<tbody>
<tr>
<td>v5, v22</td>
<td>v5v9v12v16v21v22</td>
<td>v5v9v12v16v21v22</td>
</tr>
<tr>
<td>v5, v24</td>
<td>v5v9v12v16v21v22v24</td>
<td>v5v9v14v15v25v28v24</td>
</tr>
<tr>
<td>v5, v31</td>
<td>v5v9v12v16v21v22v24v31</td>
<td>v5v9v14v13v26v29v30v31</td>
</tr>
<tr>
<td>v17, v8</td>
<td>v17v16v21v15v14v8</td>
<td>v17v16v12v9v14v8</td>
</tr>
<tr>
<td>v17, v13</td>
<td>v17v16v21v15v14v13</td>
<td>v17v16v12v9v14v13</td>
</tr>
<tr>
<td>v17, v14</td>
<td>v17v16v21v15v14v14</td>
<td>v17v16v12v9v14</td>
</tr>
<tr>
<td>v17, v29</td>
<td>v17v16v21v15v25v26v29</td>
<td>v17v16v20v23v27v30v29</td>
</tr>
<tr>
<td>v17, v26</td>
<td>v17v16v21v15v25v26</td>
<td>v17v19v21v15v25v26</td>
</tr>
<tr>
<td>v17, v28</td>
<td>v17v16v21v15v25v28</td>
<td>v17v19v21v15v25v28</td>
</tr>
<tr>
<td>v17, v25</td>
<td>v17v16v21v15v25</td>
<td>v17v19v21v15v25</td>
</tr>
<tr>
<td>v17, v15</td>
<td>v17v16v21v15</td>
<td>v17v19v21v15</td>
</tr>
<tr>
<td>v17, v21</td>
<td>v17v16v21</td>
<td>v17v19v21</td>
</tr>
<tr>
<td>v17, v22</td>
<td>v17v16v21v22</td>
<td>v17v19v21v22</td>
</tr>
<tr>
<td>v17, v24</td>
<td>v17v16v21v22v24</td>
<td>v17v19v21v22v24</td>
</tr>
<tr>
<td>v17, v31</td>
<td>v17v16v21v22v24v31</td>
<td>v17v16v20v23v27v31</td>
</tr>
</tbody>
</table>

Suppose \( P = v_1v_2 \ldots v_i \ldots v_n \) is the shortest path from \( v_1 \) to \( v_n \), any path starting from \( v_1 \) via \( v_2, \ldots, v_{i-1} \) to \( v_i \), must also be a shortest path from \( v_1 \) to \( v_i \) computed by the same heap (or array) version of Dijkstra’s algorithm. It may not hold if random version is used.

When making relaxation each round in array version, the same preference order is used. Instead, in heap version, the flip operations for maintaining the heap could change the vertex
order, based on which, a different "root" node may be chosen and relaxed when more than one nodes have the same minimum estimate so far. Examples illustrated above show this difference. But the performance by the random version is unexpected and hard to explain.

5.3 The Least Congested Shortest Path

The motivation behind the random version of Dijkstra’s Algorithm is to achieve balanced wavelength consumption in a unit-weight graph; however, as mentioned before, the simulation result is out of expectation.

The following 2 approaches could be used to achieve the goal: to find and then use the “least congested” shortest path:

1) Borrowing the idea from the implementation of the Least Congested Path [13], we can pre-select a set of shortest paths for each pair, and choose the least congested one on request. The congestion of a path is measured by the minimum value of the unused channel numbers among all links along the path. The disadvantage of this approach is that the options of the shortest path are limited.

2) Do on-line “least congested” shortest path selection. When the network graph becomes denser, it is possible that there are dozens of shortest paths for each source destination pair. Considering the on-line routing requires low computation complexity, an algorithm with limited search is preferred, thus this result could be just an approximation. The basic idea is to find one shortest path from the original network graph, and then, try other paths again on the graph by removing some of those most congested links in the chosen path. Pseudo-code could be like as follows:

PROCEDURE Approx-Least-Congested-Shortest-Path-Selection(G, s, d, k)
/* INPUT: network graph G, source node s, destination node d, search depth limit k */
/* OUTPUT: one shortest path Q */
1 compute a shortest path P between (s, d) in G by Dijkstra's Algorithm;
2 let $sd = ||P||$
3 let $cgn = \min_{e \in P} \{\text{unused-channel-number}(e)\}$
let $CE = \{e : e \in P$ and unused-channel-number(e) = cgn$\}$

$SP[end = cur = 0] = \{\emptyset, cgn, P, 0\}$

let $CE_1, CE_2, \ldots$ be $2^{[CE]} - 1$ elements in the power set of $CE$ (except the empty set);

FOR $i = 1$ TO $2^{[CE]} - 1$ DO $SP[end++] = \{CE_i, -1, NULL, 1\}$;

DO

$G' = G - SP[++cur].rmEdges$;

compute a shortest path $P'$ between $(s, d)$ in $G'$ by Dijkstra’s Algorithm;

IF $\|P'\| > sd$, CONTINUE;

let $cgn' = \min_{e \in P'} \{\text{unused-channel-number}(e)\}$

$SP[cur].cgn = cgn'$

$SP[cur].path = P'$

IF $SP[cur].depth > k$, CONTINUE;

let $CE' = \{e : e \in P'$ and unused-channel-number(e) = cgn'$\}$

let $CE'_1, CE'_2, \ldots$ be $2^{[CE']} - 1$ elements of the power set of $CE'$;

FOR $i = 1$ TO $2^{[CE']} - 1$ DO

IF $\forall j SP[j].rmEdges \neq CE'_i,$

$SP[end++] = \{CE'_i \cup SP[cur].rmEdges, -1, NULL, SP[cur].depth + 1\}$;

WHILE $(cur < end)$

RETURN any $Q \in \{SP[x].path : SP[x].cgn \text{ is maximum}\}$

$SP$ is an array of a structure with 4 fields: $rmEdges$, $path$, $cgn$ and $depth$. $rmEdges$ is an edge set which will be removed from the network graph $G$. The shortest path computed on the remaining graph is stored in $path$, and $cgn$ records the unused channel number in the most congested link along $path$. We use $depth$ to limit the search space, as line 15 indicated.

Line 1-7 find one shortest path on the original graph $G$. Then we do a local search by eliminating some of those most congested links and computing the shortest path on it. Shown in line 3 and 12, the evaluation criteria currently used is the minimum unused channel number among the links (i.e., the most congested link). In line 8-21, those with the same shortest path distance $sd$ are qualified (line 11) and appended to the array $SP$, waiting for evaluation. Finally, the one with the maximum $SP.cgn$ number is returned; if there is tie, just choose one at random.
APPENDIX A RESULTS DETAILS I

Table A.1 gives the blocked demand numbers under different $r$ and $\nu$ values without the primary path selection heuristic.

Table A.1 Blocked Demand Numbers under Different $r$ and $\nu$ without Primary Path Selection Heuristic

<table>
<thead>
<tr>
<th>Load</th>
<th>40</th>
<th>50</th>
<th>53</th>
<th>57</th>
<th>60</th>
<th>63</th>
<th>65</th>
<th>67</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDA</td>
<td>1</td>
<td>11</td>
<td>23</td>
<td>88</td>
<td>128</td>
<td>142</td>
<td>201</td>
<td>269</td>
<td>466</td>
<td>767</td>
<td>1451</td>
<td>1941</td>
</tr>
<tr>
<td>CA</td>
<td>1</td>
<td>1</td>
<td>15</td>
<td>12</td>
<td>44</td>
<td>31</td>
<td>82</td>
<td>155</td>
<td>334</td>
<td>915</td>
<td>1393</td>
<td></td>
</tr>
</tbody>
</table>

$r = 1$

$v = 0\%$ | 2  | 5  | 10 | 49 | 75 | 104| 121| 207| 372| 678| 1446| 1874|
$v = 5\%$ | 2  | 3  | 7  | 43 | 78 | 129| 127| 217| 375| 634| 1388| 1990|
$v = 10\%$| 2  | 37 | 26 | 84 | 100| 133| 169| 222| 355| 610| 1247| 1703|
$v = 15\%$| 2  | 48 | 51 | 125| 157| 193| 208| 237| 384| 666| 1175| 1592|
$v = 20\%$| 8  | 50 | 52 | 143| 152| 210| 269| 287| 449| 685| 1207| 1692|
$v = 25\%$| 8  | 54 | 61 | 154| 156| 231| 259| 288| 478| 692| 1237| 1682|
$v = 30\%$| 10 | 53 | 66 | 163| 171| 214| 268| 320| 486| 744| 1313| 1691|
$v = 40\%$| 9  | 47 | 67 | 172| 192| 242| 262| 333| 483| 728| 1340| 1749|

$r = 2$

$v = 0\%$ | 1  | 3  | 7  | 24 | 60 | 89 | 108| 196| 341| 666| 1294| 1827|
$v = 5\%$ | 1  | 4  | 2  | 34 | 58 | 95 | 113| 161| 318| 657| 1390| 1937|
$v = 10\%$| 1  | 8  | 9  | 44 | 50 | 82 | 89 | 135| 251| 535| 1167| 1720|
$v = 15\%$| 0  | 28 | 31 | 82 | 90 | 123| 152| 185| 273| 505| 1136| 1614|
$v = 20\%$| 0  | 31 | 31 | 84 | 105| 140| 156| 205| 304| 567| 1158| 1668|
$v = 25\%$| 0  | 31 | 36 | 109| 101| 152| 170| 244| 328| 581| 1156| 1591|
$v = 30\%$| 0  | 34 | 36 | 95 | 107| 156| 206| 250| 341| 591| 1142| 1546|
$v = 40\%$| 4  | 27 | 37 | 112| 118| 185| 200| 258| 360| 609| 1110| 1625|

$r = 3$

$v = 0\%$ | 0  | 3  | 2  | 29 | 59 | 100| 82 | 177| 340| 633| 1349| 1850|
$v = 5\%$ | 0  | 1  | 5  | 26 | 52 | 96 | 106| 173| 308| 627| 1308| 1934|
$v = 10\%$| 1  | 1  | 2  | 22 | 39 | 60 | 67 | 117| 233| 525| 1212| 1718|
$v = 15\%$| 0  | 11 | 14 | 43 | 44 | 79 | 91 | 139| 230| 485| 1188| 1624|
$v = 20\%$| 0  | 18 | 26 | 53 | 75 | 90 | 97 | 160| 238| 531| 1125| 1636|
$v = 25\%$| 0  | 17 | 27 | 70 | 78 | 121| 121| 169| 274| 527| 1083| 1622|
$v = 30\%$| 0  | 22 | 23 | 69 | 88 | 130| 146| 181| 291| 599| 1089| 1575|
$v = 40\%$| 1  | 22 | 30 | 89 | 106| 154| 194| 219| 360| 558| 1114| 1559|

$r = 4$

$v = 0\%$ | 0  | 3  | 3  | 29 | 54 | 99 | 98 | 163| 341| 646| 1337| 1876|
$v = 5\%$ | 0  | 1  | 5  | 25 | 53 | 96 | 105| 172| 325| 674| 1377| 1863|
$v = 10\%$| 0  | 1  | 2  | 23 | 38 | 70 | 60 | 118| 267| 515| 1238| 1724|
$v = 15\%$| 0  | 6  | 11 | 28 | 41 | 62 | 68 | 122| 229| 464| 1162| 1575|
$v = 20\%$| 0  | 12 | 5  | 33 | 59 | 72 | 96 | 120| 218| 483| 1136| 1594|
$v = 25\%$| 0  | 14 | 14 | 50 | 56 | 83 | 90 | 170| 225| 492| 1117| 1599|
$v = 30\%$| 1  | 13 | 21 | 61 | 77 | 115| 123| 171| 298| 547| 1079| 1506|
$v = 40\%$| 3  | 12 | 26 | 81 | 91 | 143| 171| 240| 309| 563| 1157| 1558|
APPENDIX B  RESULTS DETAILS II

Table B.1 gives the blocked demand numbers under different $r$ and $\nu$ values with the primary path selection heuristic.

Table B.1  Blocked Demand Numbers under Different $r$ and $\nu$ with Primary Path Selection Heuristic ($\delta=3$)

<table>
<thead>
<tr>
<th>Load</th>
<th>40</th>
<th>50</th>
<th>53</th>
<th>57</th>
<th>60</th>
<th>63</th>
<th>65</th>
<th>67</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDA</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CA</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>15</td>
<td>12</td>
<td>44</td>
<td>31</td>
<td>82</td>
<td>155</td>
<td>334</td>
<td>915</td>
<td>1393</td>
</tr>
</tbody>
</table>

$r=1$

| $\nu=0\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 2  | 0  | 0  | 0  | 0  | 0  |
| $\nu=5\%$ | 0  | 0  | 1  | 1  | 1  | 1  | 16 | 28 | 28 | 61 | 61 | 61 |
| $\nu=10\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 16 | 28 | 28 | 61 | 61 | 61 |
| $\nu=15\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 16 | 28 | 28 | 61 | 61 | 61 |
| $\nu=20\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 16 | 28 | 28 | 61 | 61 | 61 |
| $\nu=25\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 16 | 28 | 28 | 61 | 61 | 61 |
| $\nu=30\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 16 | 28 | 28 | 61 | 61 | 61 |
| $\nu=40\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 16 | 28 | 28 | 61 | 61 | 61 |

$r=2$

| $\nu=0\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=5\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=10\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=15\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=20\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=25\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=30\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=40\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

$r=3$

| $\nu=0\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=5\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=10\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=15\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=20\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=25\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=30\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=40\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

$r=4$

| $\nu=0\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=5\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=10\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=15\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=20\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=25\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=30\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| $\nu=40\%$ | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
BIBLIOGRAPHY


ACKNOWLEDGMENT

First of all, I would like to give special thanks to my adviser Dr. Lu Ruan, for her guidance, patience, trust and financial support during this thesis research.

Secondly, I would like to offer my gratitude to Prof. Arun Somani. It is my great honor to have him serving at my Program of Study committee. Thanks also give to Prof. Soma Chaudhuri, who gave me a lot of her time, advice and encouragement throughout my entire Master’s Study.

I want to take this opportunity to thank many faculty, staff members and graduate students at the Department of Computer Science, especially to our Graduate Secretary, Melanie Eckhart, for her continuous help and encouragement. I wish them all the best.

Last but certainly not least, I am forever indebted to the love and caring of my family and my wife Tianying.