2003

Traffic Engineering in Multiprotocol Label Switching networks

Chung-Yu Wei
Louisiana State University and Agricultural and Mechanical College, cwei1@lsu.edu

TRAFFIC ENGINEERING IN MULTIPROTOCOL LABEL SWITCHING NETWORKS

A Thesis

Submitted to the Graduate Faculty of
the Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

in

The Department of Electrical and Computer Engineering

by
Chung-Yu Wei
B.S., I-Shou University, Taiwan, 1997
December 2003

Acknowledgements

I would like to express my gratitude to my major professor, Dr. Morteza Naraghi-Pour. Without him, I could never find the direction on my graduate study. Also, I would like to thank the committee members, Dr. Jorge L. Aravena and Dr. David M. Koppelman, and all the professors who have helped me.

Table of Contents

List of Tables

List of Figures

Abstract

The goal of Traffic Engineering is to optimize the resource utilization and increase the network performance. Constraint-based routing has been proposed as an networks effective approach to implement traffic engineering in Multiprotocol Label Switching. In this thesis, we review several algorithms on constraint-based routing from the literature and point out their advantages and disadvantages. We then propose several algorithms to overcome some of the shortcomings of these approaches. Our algorithms are specificity suitable for large densely connected networks supporting both Quality of Service traffic and the Best Effort traffic. In large networks the size of the MPLS label space in a node may become extremely large [3]. Our algorithms allow for control on the size of the label space for each node in the network. In addition, explicit routes can be accommodated supporting both node and link affinity. We address an algorithm that implements the node and link affinity correctly. If the QoS traffic has stringent delay requirements, a path length limit can be imposed so that the number of hops on the path for such traffic is limited. Finally, we propose the $1 + 1$ and $1 : 1$ path protection mechanisms using the constraint-based routing in MPLS and establish backup for the working path carrying the primary traffic. Our approach appropriately overcome the problems and the result are satisfying.

Chapter 1

Introduction

In this chapter, we introduce the idea of Traffic Engineering in core MPLS in Section 1.1. The architecture and operation of Multiprotocol Label Switching and the online and off-line constraint-based routing problem is described in Section 1.2. In Section 1.3, we discuss the MPLS supported Quality of Service to support end-to-end QoS. The path protection in MPLS is introduced in Section 1.4.

1.1 Traffic Engineering

Traffic Engineering intends to map the traffic flows onto the existing physical network topology in order to optimize resource utilization and network performance. Specifically, traffic engineering reduces the congestion resulting from inefficient resource allocation. In the early 1990s, internet service providers mapped Internet traffic flows onto the physical network topology in an ad-hoc manner. The traffic flows simply followed the shortest path calculated by the ISPs' Interior Gateway Protocol (IGP). Using the shortest paths attempts to conserve network resources, but since the bandwidth availability and traffic characteristics are not taken into account, this approach often leads to overutilization.

For example, two paths may exist between two nodes; one is a T1 line with a bandwidth of 1.5 Mbps and a path length of 3 hops, the other one is an OCS line with a bandwidth of 155 Mbps and a path length of 5 hops. Clearly, the 5-hop path is preferred. An IGP that simply counts the hops and chooses the shortest path would select the first path as the preferred path without regard to the available bandwidth. An alternative to using the IGP protocols for TE is the use of overlay networks. An overlay network allows the establishment of an end-to-end path (within the ISP's core network). By setting up a path around the bottlenecks, traffic can be routed so as to avoid the points of congestion in the network.

By using the overlay network we in fact convert the IP network (which is packet switched) into a virtual circuit switched network. The first example of overlay networks were the ATM and frame relay networks. This lead to IP over ATM and IP over frame relay. These IP over ATM was implemented in core networks. However, IP over ATM had many limitations including a high overhead due to cell tax. More recently, Multiprotocol Label Switching (MPLS) has been proposed as an alternative overlay network.

1.2 Multiprotocol Label Switching

Multiprotocol Label Switching provides mechanisms for Traffic Engineering (TE) to maximize resource usage and optimize traffic flow by establishing end-to-end routes. Some other features include increased packet forwarding performance by using label swapping, path protection to offer reliable service, and path prioritization to provide service differentiation. We describe the operation of MPLS and the constraint-based routing problem in the follows.

1.2.1 MPLS Operation

MPLS is responsible for directing a flow of IP packets along a predetermined path across a network. This path is called a label switched path (LSP). An LSP is created by the concatenation of one or more label switched hops from one label switching router (LSR) to another LSR across the MPLS domain. An LSR is a router that support MPLS-based forwarding.

In an IP network, incoming packets are classified into Forwarding Equivalence Classes (FECs) based upon the maximum match prefix look-up with the destination IP address. The FEC is a set of IP packets that are forwarded in the same manner. The classification for an IP network is performed at each hop. In the MPLS domain, the classification of packets into an FEC is done only at the entry into the domain (e.g., the ingress LSR). At subsequent LSRs, no packet classification needs to be done [13].

In MPLS, the packet forwarding process at each LSR is based on the concept of label swapping. After the classification into the FECs in the ingress LSR, a 4-byte MPLS header is added to the beginning of IP packet (including 20-bit label value, 4-bit experimental use, 1-bit bottom of stack , and 4-bit time to live). In the subsequent node of the MPLS domain, the LSR's forward the packet based on the label value. When a packet containing a label arrives at an LSR, the LSR examines the label and uses it as an index into its MPLS forwarding table. The incoming label is replaced with the outgoing label and the packet is forwarded to the next LSR in the LSP. The labelled packet is forwarded along the LSP by each LSR until it reaches the egress LSR. In the egress LSR, the label is removed and the packet is forwarded based on the IP destination address again. This label swapping operation results in high-speed switching of the IP packet through the MPLS domain [15].

MPLS effectively superimposes connections of LSP on the connectionless IP network. Packet forwarding occurs on the LSPs. LSPs are established prior to data transmission or upon detection of a certain flow of data. The labels are distributed using label distribution protocols such as LDP, RSVP-TE, CR-LD [13].

The MPLS protocol architecture supports explicit routing for selecting the LSP's, and two different explicit routing methods are used [7]. For an explicitly routed LSP, if an LSR (usually the ingress LSR or the egress LSR) pre-specifies some of the LSR's on that LSP, the LSP is said to be loosely explicitly routed. If the LSR's along the entire LSP are specified, the LSP is said to be strictly explicitly routed.

1.2.2 Constraint-Based Routing

Constraint-based Routing uses the information of traffic patterns, networks resources, and topology state to compute routes for the traffic [2]. It greatly reduces the amount of administrative path configuration and manual intervention required to achieve Traffic Engineering objectives. MPLS Traffic Engineering (MPLS TE) associates a set of attributes with network resources for constraint-based routing. The attributes associated with the network resources include link capacity, maximum reservable bandwidth and the available bandwidth, as well the administration policies [2]. These attributes can be used to constrain the placement of LSP's.

Constraint-based routing can be online or off-line. Online constraint-based routing takes resource constraints into account and calculates one LSP at a time. Every ingress LSR in the network finds paths for its LSPs separately based on its own information. The challenge with this approach is that it is not deterministic. The order of the calculated LSPs plays a critical role in determining paths across the network. The LSPs calculated earlier have more resources available to them than LSPs calculated later. If the order of the calculated LSPs is changed, the resulting set of paths for the LSPs can also change [16].

Off-line constraint-based routing examines each link's resource constraint and the requirement of the traffic demand to obtain routes for all the demand simultaneously. This approach takes more time to complete, but since it performs global calculations, the output of the off-line calculation is a set of LSPs that can optimize utilization of network resources. Then the set of LSPs is installed into the ingress LSRs.

The constraint-based routing can be used to compute the paths periodically, e.g., daily or weekly. In the next chapter, we'll examine several off-line constraint-based routing problems, review the existing work in the literature, and propose new approach to these problems.

1.3 Quality of Service Support with MPLS

Many applications require end-to-end Quality of Service support. An end-to-end path can be divided into two parts: the path between the end users and the service provider's edge router, and the path across the network core between the ingress and the egress edge routers. To maintain end-to-end QoS, each part is required to support end-to-end service.

MPLS and Differentiated Services (Diffserv) make scalable QoS support in the core possible. Diffserv takes the IP TOS (type of service) field (denoted Diffserv code point or DSCP) and uses it to carry information about the requirements of the IP packet. In the Diffserv domain, packets are buffered and scheduled in accordance to their DS-fields. The ingress LSR can classify the packets and marks them with DS Code Points (DSCP) that match the QoS requirements of the application. The packets are then forwarded along LSPs that meet the DS class of service requirements. Label inferred LSPs (L-LSP) and EXP inferred LSPs (E-LSP) have been proposed for aggregating DS marked packets into MPLS tunnels.

Label inferred LSPs associate a specific MPLS label with the DS code point in the IP header. The ingress LSR examines the DSCP in the IP packet header and selects an LSP that has been provisioned for that QoS level. Different LSPs are used for different classes of traffic. Each LSR on that path will then determine the QoS treatment for the packets from their incoming label. The LSRs then implement packet scheduling consistent with the inferred DSCP as defined in the DS architecture. We review the QoS-based routing that differentiates the service class for each LSP in the next chapter [15].

EXP inferred LSPs use the experimental bits in the MPLS header to convey the QoS requirements of the packets to the LSRs. There are three bits in the EXP part of the MPLS header, which can support up to eight DS code points. The mapping between the DSCP and the EXP bits is made at the ingress LSR. Packets marked with the EXP bits receive per-hop forwarding treatment [15].

1.4 MPLS Path Protection

With the migration of real-time and high-priority traffic to IP networks, the future IP networks are required to provide high levels of availability and reliability. However, the IP networks which rely on routing protocols to maintain network connectivity can take a substantial amount of time to recover from a failure. Though robust and survivable, the routing tables in the IP protocol may take several seconds to minutes to converge after a failure, and this can cause serious disruption of services in the interim.

The traffic engineering feature of MPLS supports explicit routing, and this allows MPLS networks to pre-establish backup LSP's with the same amount of bandwidth as the primary LSP. Currently, new failure recovery mechanisms are being designed for MPLSs which allow the fail over time to be considerable less than that achievable by routing protocols. In the following, two classification of recovery models according to [12] and [13] are presented.

1.4.1 Rerouting vs. Protection Switching

The path protection in MPLS is based on a backup or recovery path to carry the traffic while the primary or working path has failed. An LSR is referred to as path switch LSR (PSL) if the working path and backup path diverge at this node, and referred to as the path merge LSR (PML) if the working path and the backup path converge at that node. The PSL is the origin of the traffic recovery, but may not be the origin of the working path; similarly, the PML may not be the destination.

Failure recovery is often classified into rerouting and protection switching. Rerouting is defined as establishing a new path or path segment when a failure occurs and restoring the traffic as required. No resources are reserved in advance until the fault occurs and the location of the fault is known. Consequently, the rerouting mechanism provides higher resource utilization than the other protection switching mechanism.

However, rerouting mechanisms are slow due to the fact that upon detection of a fault, paths or path segments to by-pass the fault are established through signaling, and MPLS signaling relies on IP protocol. Re-signaling a new LSP can be time consuming and requires that routing tables be updated.

Protection switching recovery mechanisms pre-establish a backup path (or path segment) for each working path (or path segment) based on network routing policies, bandwidth requirement and administrative requirements. The backup path can be link or node disjoint with the working path. For the link disjoint scheme, no links can be shared between the working path and the backup path. And for the node disjoint scheme, no nodes and links can be shared between the working path and the backup path. Clearly, if the backup path shares any resources with the working path, the reliability of the failure recovery will be degraded.

Another option is for the backup paths to be configured in advance without reserving the resources. Upon failure of the working path, resource resource reservation must be signaled to all the nodes on the backup path. However, this increases the latency of the restoration mechanism and the resources on the configured paths may not be available when a fault occurs.

1.4.2 Path Protection

Path protection is intended to protect against any link or node fault on a path or a segment of a path. The resources on the backup LSP are fully reserved, and two types of path protection have been proposed.

1.4.2.1 One-plus-one (1+1) Path Protection

Among the path protection mechanisms, the one-plus-one $(1 + 1)$ path protection scheme is the fastest. In this scheme, the backup path carries the same traffic as the working path. Therefore, the path merge LSR (PML) receives two copies of the traffic and for reads only the traffic from the working path. Upon the detection of the fault, the PML only needs to read the data from the backup path. The fault detection and protection switching at the same time, so it's fast and simple.

1.4.2.2 One-to-one (1:1) Path Protection

In one-to-one $(1 : 1)$ path protection, resources of the backup path are available to pre-emptible low-priority traffic. Upon the detection of a failure, a notification to the PSL would trigger the recovery action; therefore this type of path protection is PSL-oriented. The recovery action involves preempting the low-priority traffic, sending a notification to the PML to update its label mapping, and switching the protected traffic to the backup path. This approach can be extend to m-to-n $(m : n)$ protection, where m recovery paths can be shared to protect n working paths.

Chapter 2

Routing and Admission Control

2.1 Introduction

In this chapter we review several off-line constraint-based routing algorithms from the literature, including routing for the QoS and Best Effort service classes, routing with label constraints for large scale network, and the use of resource attribute for node and link affinities. We will also discuss some of the drawbacks and shortcomings of these algorithms. Next several alternative approaches will be presented to overcome the shortcomings of these algorithms in Section 2.3. The experimental results comparing our results with the previous works in the case of several example networks are demonstrated in Section 2.4.

2.2 Previous Work

2.2.1 Multiservice, Multipriority Traffic Engineering Design

In [4], Mitra and Ramakrishnan proposed a technique for traffic engineering in QoS supported data networks using multi-commodity flow (MCF) problem. In their approach, the QoS traffic is routed first and is allowed to use all of the available link capacities. Multiple QoS service classes can be accommodated. The BE traffic is routed next, and uses only the unutilized or residual capacities. In the following, we describe the approach of [4] in detail and point out some of its drawbacks.

2.2.1.1 QoS Traffic

For the QoS service classes, an admissible route set is pre-selected for each origin-destination (OD) pair, and the final routes are chosen from this set. For a given service class and any given OD pair, policy concerns can be taken into account in the selection of the route set. For example, in the real-time services such as voice and video, the admissible routes are restricted in the number of hops, and for the delay insensitive services, the hop count can have a less stringent restriction.

Quality of Service routing is achieved in two stages with a concomitant multi-commodity flow problem for each stage. In the first stage, a network revenue is defined. By maximizing this revenue, the network utilization is maximized. In the second stage, the objective is to conserve the resources utilized by the QoS traffic. The details are given as follows.

A network is represented by a graph $G = (V, E)$. The set of routers in the network form the vertices of the graph V. The communication links connecting the routers are represented by directed links in the graph and form the edge set E. Consider a connected network with N nodes, M directed links and capacity C_l bits/sec on link l, $l = 1, 2, ..., M$. Denote the amount of traffic of class s from source node σ_s to destination node σ_t by $D_{s,\sigma}$, where $\sigma = (\sigma_s, \sigma_t)$. Define $D_s = [D_{s,\sigma}]$ as the $N \times N$ demand matrix of class s. Additional notation is given in Table 2.2.1.

In [4], the network revenue for QoS traffic is defined as follows.

Table 2.1: Notation of MCF problem for QoS traffic.

V	The set of vertices (routers) in the network
E	The set of edges (directed links) in the network
$G = (V, E)$	The Network
$l \in E$	Link
C_l	Capacity on link l
$D_{s,\sigma}$	Bandwidth demand for class s and OD pair σ, $\sigma = (\sigma_s, \sigma_t)$
$R(s, \sigma)$	The pre-selected route set for class s and OD pair σ
S_{QoS}	The set of service classes for QoS traffic
$e_{s,r}$	The earning per unit carried traffic on route r, and class s
W_{QoS}	Network revenue for QoS traffic
$X_{s,r}$	Allocated bandwidth for class s on route r

$$W_{QoS} = \sum_{s \in S_{QoS}} \sum_{\sigma} \sum_{r \in R(s,\sigma)} e_{s,r} X_{s,r},$$

where $e_{s,r}$ and $X_{s,r}$ are given in Table 2.2.1. Note that if $e_{s,r} = 1$ for all s and r, then W_{QoS} represents the amount of QoS traffic that the network will carry. For each $s \in S_{QoS}$ and σ we wish to determine the allocated bandwidth $X_{s,r}$ on route $r \in R(s, \sigma)$. The optimization problem addressed in [4] is described below.

MCF problem P1:

$$maximize \;\; W_{QoS}, \tag{2.1}$$

subject to

$$\left.\begin{array}{l} \sum_{r \in R(s,\sigma)} X_{s,r} \le D_{s,\sigma} \\ X_{s,r} \ge 0, \forall r \in R(s, \sigma) \end{array}\right\} \forall s \in S_{QoS}, \forall \sigma, \tag{2.2}$$

$$\sum_{s \in S_{QoS}} \sum_{\sigma} \sum_{r \in R(s,\sigma): l \in r} X_{s,r} \leq C_l, \forall l. \tag{2.3}$$

In (2.1), the objective is to maximize the revenue W_{QoS} which is the sum of the earnings over all classes $s \in S_{QoS}$, and all OD pairs σ and all routes. The constraint in (2.2) ensures that the allocated bandwidth for class s and OD pair σ does not exceed the demand $D_{s,\sigma}$, and that the allocated bandwidths should be non-negative. The constraint in (2.3) ensures that the total allocated bandwidth on link l does not exceed its capacity C_l.

The above constrained optimization is a multicommodity flow (MCF) problem, in which the commodities are differentiated by their origin-destination pairs and traffic classes, and share the common link capacities. The *MCF problem P1* is a linear programming problem. Thus we can use any linear programming solver, such as the commercial linear programming package CPLEX [10] to obtain a solution. Upon solving the constrained optimization problem *P1*, we obtain the maximum revenue W^*_{QoS}.

In the second stage, the objective is to conserve the resources utilized by the QoS traffic, so the problem is to minimize the bandwidth-hops utilized. In [4], the formulation is not described in detail, so we surmise the formulation as follows. Let $h_{s,r}$ be the number of hops on route $r \in R(s,\sigma)$.The problem is described below.
MCF problem P2:

$$minimize \sum_{s \in S_{QoS}} \sum_{\sigma} \sum_{r \in R(s,\sigma)} h_{s,r} X_{s,r}, \tag{2.4}$$

subject to

$$\left.\begin{array}{l} \sum_{r \in R(s,\sigma)} X_{s,r} \leq D_{s,\sigma} \\ X_{s,r} \geq 0, \forall r \in R(s,\sigma) \end{array}\right\} \forall s \in S_{QoS}, \forall \sigma,$$

$$\sum_{s \in S_{QoS}} \sum_{\sigma} \sum_{r \in R(s,\sigma): l \in r} X_{s,r} \leq C_l, \forall l,$$

$$W^*_{QoS} = \sum_{s \in S_{QoS}} \sum_{\sigma} \sum_{r \in R(s,\sigma)} e_{s,r} X_{s,r}. \tag{2.5}$$

In (2.4), the objective is to minimize the total bandwidth-hops utilized by the QoS traffic. This is the sum of the bandwidths consumed by all the routes in the network over all classes $s \in S_{QoS}$ and all OD pairs σ. In (2.5), the revenue in the second stage is confined to be the same as the revenue from the first stage W^*_{QoS}. Using CPLEX [10] to solve the *MCF problem P2*, we will obtain the allocated bandwidth $X_{s,r}$ on route $r \in R(s,\sigma)$.

2.2.1.2 Routing of Best Effort Traffic

The routing of BE traffic is link-based in that no pre-selected route sets are chosen for any given OD pair. Unlike the route-based QoS traffic, BE traffic problem attempts to determine the allocated bandwidth on links instead of the bandwidth on routes. In [4], the network revenue for BE traffic is defined as follows.

$$W_{BE} = \sum_{\sigma} e_{BE,\sigma} F_{BE,\sigma},$$

where $F_{BE,\sigma}$ is the total Best Effort traffic carried for OD pair σ, and $e_{BE,\sigma}$ is the earning per unit carried traffic of OD pair σ.
Here the objective is to maximize the network revenue W_{BE} using the residual link capacities left by the QoS traffic. The notation and the problem formulation are given in the following.

Table 2.2: Notation of MCF problem for BE traffic.

W_{BE}	Network revenue for BE traffic
$e_{BE,\sigma}$	The earning per unit carried traffic for OD pair σ
$F_{BE,\sigma}$	Total carried traffic for OD pair σ
$D_{BE,\sigma}$	Bandwidth demand for OD pair σ
$Y_{\sigma,l}$	Allocated bandwidth for OD pair σ on link l
C_l^{res}	Residual capacity on link l
$L_{in}(n)$	Set of links directed into node n
$L_{out}(n)$	Set of links directed out of node n
σ_s	Source node of OD pair σ
σ_t	Destination node of OD pair σ

MCF problem P3:

$$maximize \;\; W_{BE}, \tag{2.6}$$

subject to

$$0 \leq F_{BE,\sigma} \leq D_{BE,\sigma}, \forall\sigma, \tag{2.7}$$

$$\sum_{l\in L_{in}(n)} Y_{\sigma,l} - \sum_{l\in L_{out}(n)} Y_{\sigma,l} = \left\{ \begin{array}{ll} F_{BE,\sigma} & \text{if } n=\sigma_t \\ -F_{BE,\sigma} & \text{if } n=\sigma_s \\ 0 & otherwise \end{array} \right\} \forall n, \forall\sigma, \tag{2.8}$$

$$Y_{\sigma,l} \geq 0, \forall\sigma, \forall l, \tag{2.9}$$

$$\sum_{\sigma} Y_{\sigma,l} \leq C_l^{res}, \forall l. \tag{2.10}$$

The objective function in (2.6) is the network revenue. The constraint in (2.7) ensures that the total carried traffic $F_{BE,\sigma}$ for OD pair σ does not exceed the traffic demand $D_{BE,\sigma}$ for this OD pair. The flow conservation constraint in (2.8) sets the total traffic leaving the source node and the total traffic entering the destination node equal to the carried traffic $F_{BE,\sigma}$. For an intermediate node, the total traffic entering the node should be the same as the total traffic leaving the node. The constraint in (2.9) ensures that the allocated bandwidths are non-negative. The constraint in (2.10) ensures that the total allocated bandwidths on link l does not exceed its residual capacity C_l^{res}.

For the above link-based formulation, the allocated bandwidth $Y_{\sigma,l}$ is obtained for each link l and each OD pair σ. Unfortunately, the $Y_{\sigma,l}$ do not indicate the utilized paths. Possibly more than one path may exist for an OD pair and such multiple paths may share a number of links. A procedure called flow decomposition is used to identify the routes or the paths from the solution for the BE traffic. The flow decomposition procedure is as follows.

The following procedure is repeated for all OD pairs σ. For each OD pair σ, a network $\mathcal{N}'$ whose graph is a subgraph of the original network graph defined and used in the procedure as described in the following.

Flow-decomposition:

1. Fix an OD pair σ.
2. Generate the graph for $\mathcal{N}'$ by removing all the links l such that $Y_{\sigma,l} = 0$ and denote the remaining graph by $\mathcal{G}$. Note that the BE traffic for OD pair σ only used the links in graph $\mathcal{G}$.
3. Trace a route r in $\mathcal{G}$ from the source σ_s to the destination σ_t using the depth first search method. Let $Y_{BE,r} = min_{l \in r} Y_{\sigma,l}$. Compute
   $F_{BE,\sigma} = F_{BE,\sigma} - Y_{BE,r}$,
   and
   $Y_{\sigma,l} = Y_{\sigma,l} - Y_{BE,r}, \forall l \in r$.
4. End if $F_{BE,\sigma} = 0$
   else goto 2

Repeat for all OD pairs σ.

2.2.1.3 Drawbacks of the Mitra's Approach

As mentioned previously, the route-based formulation is used in [4] for the QoS traffic. This requires that a route set be preselected for each OD pair. If the route sets are selected to have only a few routes, the selection process will not be too complicated. However, the final solution from *MCF problem P1* may not be desirable in that the network revenue W^*_{QoS} (or carried traffic when $e_{s,r}$ is one for all s and r) is too small. Larger route sets result in better solutions for *MCF problem P1*, but they are much harder to select prudently particularly in large mesh-like networks.

One drawback of the *MCF problem P3* applied to BE traffic is that the resulting solution of the LP problem may contain cycles or loops. The objective of the formulation is to maximize the network revenue regardless of the resources that are utilized. For the same amount of revenue, there could be many possible feasible solutions including some paths with cycles. There is no guarantee that the *MCF problem P3* will not choose a path with cycles. We propose a solution which accounts for the utilized resources in the objective function. The details are given in Section 2.3.2.

In [3] the authors have pointed out another problem with the approach in the *P3* for routing BE traffic. The approach in *P3* can result in many paths for a given OD pair. This may result in extremely large routing tables in the network routes, and consequently limit the scalability of the network. For example, in Figure 2.1, there are S sources, T destinations and P parallel paths connected between node V and node U. Assume each source wants to send traffic to each destination. A possible solution of *P3* for an OD pair is to spread the traffic evenly over the P parallel paths. Using flow decomposition for this OD pair, we will trace a route (or label switched path) on each parallel path, and result to P LSP's. Thus for the entire demand set, $S \times T \times P$ LSP's might be found. In particular, at the bottleneck V, we need $S \times T \times P$ labels for each of the LSP's leaving the node. In [3], authors address the problem of large routing tables and propose some solutions. Their approach is described next.

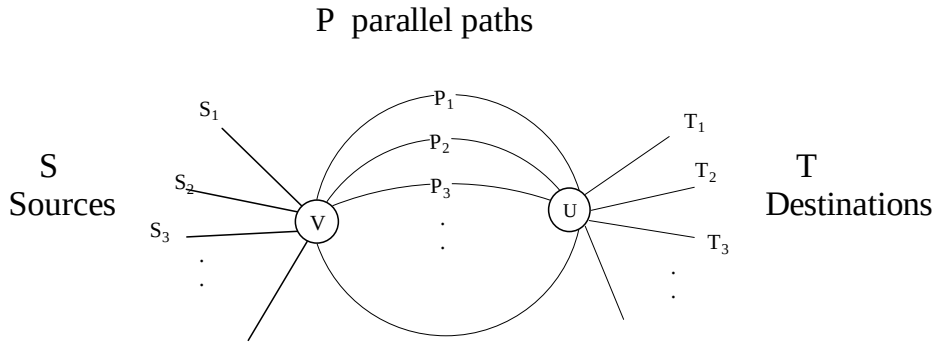


Figure 2.1: Worst case of labels

2.2.2 MPLS Routing with N+M Labels

For a network with N nodes and M edges, Applegate and Thorup proposed some algorithms in [3] to reduce the number of MPLS labels to $N + M$ in the worst case. Their approach is based on the technique of to-trees and rerouting described in the following.

Multiple LSP's can be merged at a specific node if packets from these LSP's are forwarded in the same manner, e.g, over the same downstream path with the same forwarding treatment [7]. As a result, the label based forwarding mechanism of MPLS can be used to route packets along multi-point-to-point trees. A to-tree R is defined as a multi-point-to-point tree rooted in destination node t, and with all links oriented toward t. The simplest way to implement the to-tree in MPLS is to use a single label l_R for each to-tree R.

Given a routing solution Φ, rerouting Φ means to find a solution Φ' which is at least as good as Φ in that no link is assigned a larger load than in Φ. The load in Φ is used as the capacity C_l on link l for the rerouting problem. The problem is feasible because of the existence of Φ.

We describe the approach of [3] in detail and point out some of its drawbacks. The algorithm can be classified into three stages. In the first stage, a modified version of the MCF problem in [4] is modified to find the solution Φ. In the second stage, a similar MCF problem is used to remove cycles, whereby a solution Φ without cycles is achieved. In the last stage, the solution Φ is rerouted into to-trees to find a solution Φ' that forms less labels for to-trees. An algorithm is proposed to find the to-tree R including how to send as much traffic as possible in R. More details are given below.

A network topology is presented as a graph $G = (V, E)$, where the vertex set V is the set of routers, and the edge set E is the set of directed links. We have the sources $S \subseteq V$ and destinations $T \subseteq V$. Each link $l \in E$ has a capacity C_l. A demand matrix D specifying the traffic demand $D_{(s,t)}$ from each source $s \in S$ to each destination $t \in T$ is given. Only one class of service is considered here.

In the first stage, the goal is to minimize the dropped demand. For each destination $t \in T$, and link $l = (u, v) \in E$, the algorithm determines the flow of traffic $F_l^t \geq 0$ on link l toward destination t from all the sources $s \in S$. Different sources are not distinguished if they have a demand for the same destination. The notation and the problem formulation is given in the following [3].

MCF problem P4: [3]

$$maximize \quad \sum_{t \in T} \sum_{(u,t) \in E} F_{(u,t)}^t \tag{2.11}$$

subject to

$$\sum_{(s,w) \in E} F_{(s,w)}^t - \sum_{(u,s) \in E} F_{(u,s)}^t \leq D_{(s,t)}, \forall t \in T, \forall s \in S \tag{2.12}$$

$$\sum_{(v,w) \in E} F_{(v,w)}^t - \sum_{(u,v) \in E} F_{(u,v)}^t = 0, \forall t \in T, \forall v \notin T, S \tag{2.13}$$

$$\sum_{t \in T} F_l^t \leq C_l, \forall l \tag{2.14}$$

$$F_l^t \geq 0, \forall l, \forall t \tag{2.15}$$

In (2.11), the objective is to carry as much traffic as possible. Maximizing the traffic entering all the destination nodes is equivalent to maximizing the carried traffic. The flow conservation constraint in (2.12) ensures that the total traffic leaving a source node $s \in S$ destined for a node t should not exceed the traffic demand $D_{(s,t)}$. Note that $D_{(s,t)}$ is non-negative. For an intermediate node, the flow conservation constraint in (2.13) ensures that the total traffic entering the node should be the same as the total traffic leaving this node. In (2.14), the constraint ensures that the total allocated bandwidth on link l does

Table 2.3: Notation for the N+M label problem

V	Set of routers or nodes
E	Set of directed links
S	Set of sources
T	Set of destinations
l	Link $l = (u, v)$ represent link between node u and node v
F_l^t	Carried traffic on link l toward destination t
C_l	Capacity on link l
D	Traffic demand matrix
$D_{(s,t)}$	Traffic demand of source s and destination t

not exceed its capacity C_l. The constraint in (2.15) ensures that the allocated bandwidths should be non-negative. A solution $\Phi = \{F_l^t\}$ of the amount of bandwidth allocated on each link $l \in E$ for each destination node $t \in T$ will be obtained after solving the *MCF problem P4*.

In the second stage, a linear programming problem base on solution Φ is formulated to remove the cycles. The capacity C'_l is the total allocated bandwidth on link l in the solution Φ. $D'_{(v,t)}$ is the total carried traffic from node v to t in solution Φ. The problem is as follows [3].
MCF problem P5:

$$minimize \quad \sum_{t \in T} \sum_{(u,v) \in E} F^t_{(u,v)} \tag{2.16}$$

subject to

$$\sum_{(v,w) \in E} F^t_{(v,w)} - \sum_{(u,v) \in E} F^t_{(u,v)} = D'_{(v,t)}, \forall t \in T, \forall v \notin T \tag{2.17}$$

$$\sum_{t \in T} F_l^t \leq C'_l, \forall l$$

$$F_l^t \geq 0, \forall l, \forall t$$

The objective in (2.16) is to reduce the total bandwidth allocated in the network. As a result, the cycles will be removed. Two flow conservation constraints in (2.12)and (2.13) is replaced by (2.17). The constraint ensures that the same amount traffic should be carried as that in solution Φ. Note that $D'_{(v,t)} = 0$ if $v \notin S$.

A solution $\widetilde{\Phi}$ without cycles is obtained from the two stages above. Next,to-trees are found from the solution $\widetilde{\Phi}$, and the flows are rerouted to the destination nodes t in Φ on the to-trees.

In the last stage, an algorithm is used to distribute all the flows to destination node t on trees to t. The algorithm for destination node t is described as follows [3].
Flows-to-Trees(t)

1. End if there is no demand from any source $s \in S$ to t, e.g, $D_{(s,t)} = 0, \forall s \in S$
2. Take only links with flows to t ($F_l^t > 0$), and construct a tree R to t spanning all sources with demand to t.
3. Move as much flow as possible to R. Then reduce the amount of traffic carried in R on $D_{(s,t)}, \forall s \in S$ and $F_l^t, \forall l \in E$
4. goto 1

The tree R to t in step 2 can be found as follows:
For each node v, if v has an outgoing link with flow to destination node t, then include an arbitrary such link in R. To claim that R is a tree to t spanning all nodes with demand to destination node t, we notice that from (2.17), a node has outgoing flow to destination node t if it has incoming flow to t or demand to t. Thus we'll include all the links for all nodes $v \neq t$ with demand to destination node t or the links with an incoming flow.

Moving as much flow as possible to R in step 3 can be formulated as a Linear Programming problem. Define D_v^R as the non-negative demand that can be carried from node v in R, and F_l^R as the non-negative flow on link l in R. For a specific R and each link $l \in E$, we want to find the maximum flow that can be carried in R. The formulation is as follows [3].
MCF problem P6:

$$maximize \quad \sum_{v \in R} D_v^R \tag{2.18}$$

subject to

$$\sum_{(v,w) \in R} F_{(v,w)}^R - \sum_{(u,v) \in R} F_{(u,v)}^R = D_v^R, \forall v \neq t \tag{2.19}$$

$$D_v^R \leq D_{(v,t)}, \forall v \tag{2.20}$$

$$F_l^R \leq F_l^t, \forall l \tag{2.21}$$

In (2.18), the objective is to maximize the total traffic demand that can be carried in R from each node v. The flow conservation constraint in (2.19) ensures that the total traffic leaving node v are the sum of the traffic entering node v and the demand from v. In (2.20), for each node v the carried demand D_v^R in R should not exceed the carried demand $D_{(v,t)}$ to t. In (2.21), the carried traffic F_l^R in R on link l also can't exceed the carried traffic F_l^t to t on link l. After finding the flows in R, D_v^R is subtract from $D_{(v,t)}$, and for link l in R, F_l^R is subtract from F_l^t. The process repeated until $D_{(v,t)} = 0$ and $F_l^t = 0$.

The flow in tree R is maximal, so we will remove at least one link with flow to t. Thus the algorithm will find at most $|E|$ to-trees (where for a set A, $|A|$ is the cardinality of A). For all destinations, $|T| \times |E|$ to-trees will be found. The property of basic solutions in linear programming is then used to prove that the algorithm will generate only up to $|T| + |E|$ to-trees [3].

The drawback of the approach in [3] is that the analysis is for the worst case scenario which does not happened in realistic networks. A network with 300 nodes and 1056 edges might result in 95×10^6 ($|S| \times |T| \times |E|$) labels using the *MCF problem P3* in the worst case. However, the maximum number of labels generated in their experiment is only 141k. Using the approach in [3], the number of labels is reduced from 141k to 303. This however, is an overkill that creates additional burdens and constraints in the network due to the deployment of to-trees. Today's routers can handle table sizes of 200k labels without any problem and need not reduce to table size to such small values. The number of labels could be a problem if *MCF problem P3* is deployed on large scale networks, so we propose another approach to overcome its scaling limitation. We'll discuss our approach in Section 2.3.1.

2.2.3 Explicit Route

MPLS Traffic Engineering associates attributes with network resource. Two of the resource class attributes are link affinity and node affinity. With these affinities, the network administrator can control the LSP's to include or exclude specific groups of links or nodes. In particular, we can specify the desired path partially or completely into an included node list and use this list to formulate the constraints in the MCF problem. For the partially listed nodes the effect is the same as the loosely explicit route. For the completely listed nodes, it's the same as imposing the strictly explicit route. In addition, excluded nodes can also be included in the formulation.

In [5], the authors propose a solution for the node affinity in the constraint-based routing problem. Additional constraints are added to the MCF problem to allow for included or excluded nodes on the paths. We use our notation base on *MCF problem P3* to show their approach. The solution is as follows. Let $N_e(\sigma)$ be the set of excluded nodes for the OD pair σ, and let $N_i(\sigma)$ be the set of included nodes for the OD pair σ.

Excluded node: The sum of outgoing and incoming flows to the excluded nodes should be zero.

$$\sum_{l \in L_{in}(n)} Y_{\sigma l} + \sum_{l \in L_{out}(n)} Y_{\sigma l} = 0, \forall n \in N_e(\sigma)$$

Included node: The outgoing or incoming flows to the included nodes should be greater than zero.

$$\sum_{l \in L_{in}(n)} Y_{\sigma l} \geq 0, \forall n \in N_i(\sigma)$$

While the excluded node constraint results in a correct solution, the included node constraint can result in erroneous behavior. Consider an included node. Suppose a shorter path exists between the source node and the destination node that when the traffic is routed on this path a higher network revenue is achieved. In this case, the path is likely to avoid the included node. Now due to the included node constraint, the incoming flows or

outgoing flows of the included node must be greater than zero. This will result in a cycle around the included node, which clearly undesirable and not the intend goal.

In Section 2.3.3 we propose an algorithm for the included node which either routes the path through the included node or reject it.

2.3 Our Approach

We propose several alternative approaches to deal with the shortcomings of the previous algorithms. In Section 2.3.1, label constraint is imposed on the *MCF problem P3* to allow scaling to large networks. In Section 2.3.2, the objective function is modified in order to account for the utilized resources in order to eliminate the cycles. In Section 2.3.3, we propose an algorithm to implement the node and link affinity. In Section 2.3.4, a modified flow decomposition process is used to reduce the number of LSP's. In Section 2.3.5, a link-based QoS approach with hop-constraint is presented.

2.3.1 Label Constraint

The entry of a table in MPLS comprises of incoming label and outgoing label. Here we assume a global label space. In order to control the number of entries used in the routing table of a node, we can limit the number of incoming labels. In other word, we can control the number of LSP's entering the node. Since MPLS supports label merge, the number of outgoing label won't exceed the incoming labels.

The link-based *MCF problem P3* for the BE Traffic presented in [4] does not prevent the splitting of the demand on multiple routes. Consequently, the number of routed LSP's can not be predicted. If it is desired that the traffic not be split over multiple LSP's, a non-bifurcation requirement must be included in the problem setup which often leads to an integer programming problem (IP). In [8], the author address this problem in MPLS which allows only one LSP for each demand. In [5], the author generalized the idea of traffic bifurcation by forming a Mixed Integer Programming (MIP) problem. They define a parameter called granularity g ($0 \leq g \leq 1$) which specifies how coarsely a traffic demand can be divided into multiple LSP's. In fact the inverse of granularity denotes the number of LSP's that can be set up to carried the traffic for one OD pair. We employ the granularity in the case of BE Traffic in order to control the split of demand into multiple LSP's. Base on this parameter, the number of LSP's entering each node can be estimated, and now the label constraint on each node can be used. The constraint is as follows.

$$Y_{\sigma,l} = M_{\sigma,l} \times (D_{BE,\sigma} \times g), \forall\sigma, \forall l \tag{2.22}$$

$$\sum_{\sigma} \sum_{l \in L_{in}(n)} M_{\sigma,l} \leq L_{max}(n), \forall n \tag{2.23}$$

$$M_{\sigma,l} \in Z, \ \textit{where} \ Z \ \textit{is the set of integers.} \ 0 \leq M_{\sigma,l} \leq \lfloor 1/g \rfloor, \forall\sigma, \forall l \tag{2.24}$$

In (2.22), $(D_{BE,\sigma} \times g)$ is the basic unit of flow that can be allocated to $Y_{\sigma,l}$, and $M_{\sigma,l}$ is an integer variable corresponding to $Y_{\sigma,l}$. The constraint in (2.22) ensures that the traffic carried on link l for OD pair σ can only be an integer multiple of the basic unit traffic $(D_{BE,\sigma} \times g)$. In (2.23), $L_{max}(n)$ is the maximum number of labels allowed on node n. The constraint in (2.23) ensures the maximum number of incoming labels $\sum_{\sigma} \sum_{l \in L_{in}(n)} M_{\sigma,l}$ to node n does not exceed the label bound $L_{in}(n)$.

When $g = 1$, the demand between an OD pair can not be split between multiple LSP's. Only a single LSP can be used to carry this demand. In this case $M_{\sigma,l}$ is 0 or 1 for all σ and l. If $g < 1$, the range of $M_{\sigma,l}$ increases, which means that the demand can be split up to at most $\lfloor 1/g \rfloor$ different LSP's.

The *MCF problem P3* will be now augmented with the constraints in (2.22)-(2.24) to form a mixed integer programming (MIP) problem. A MIP problem has the same form as an LP problem with a linear objective function and subject to linear constraints. However, some of the variables are constrained to be integers where as others may be real numbers. This results in a MIP problem. A popular method for solving MIP problems is branch and bound method. The Branch and Bound method begins by finding the optimal solution without the integer constraints. If the variables whose values are constrained to be integers already have integer values, then it stops. If one or more integer variables have non-integral values, one fractional variable is chosen for branching, and two new subproblems are generated where the variable is more tightly constrained. In the branch and bound method, a series of LP subproblems are solved, and a tree of subproblem is built. Solving such problems may require far more computing time than the same problem without the integer constraints. The commercial package CPLEX can also be used to solve MIP problems. This approach however, is extremely time consuming. In Section 2.4, we propose a method for getting around the complexity of solving MIP problems.

2.3.2 Loop Elimination

As mentioned previously, the approach in [4] may result in cycles. There are three ways to remove the cycles. First, we can minimize the resources by modifying the objective function as to account for the utilized resources. Another approach is to perform another optimization in order to minimize the resources used such as in *MCF problem P5*. The third approach is to find the loops in the flow decomposition procedure. While we are tracing a path from the source to the destination, if a node is re-visited before reaching the destination, it implies that a loop has been formed. We can reduce the resources utilized by the loop, and simply ignore the nodes in the loop. The first approach is the most efficient method as it has a much lower complexity than the other two methods. In the following, we implement the first approach by modifying the network revenue as follows.

$$W_{BE} = \sum_{\sigma} e_{BE,\sigma} F_{BE,\sigma} - \varepsilon \sum_{\sigma} \sum_{l} Y_{\sigma,l} \tag{2.25}$$

In (2.25),$\sum_\sigma e_{BE,\sigma} F_{BE,\sigma}$ is the total reward for the network resulting from the carried traffic. The term $\sum_\sigma \sum_l Y_{\sigma,l}$ is the total bandwidth (resource) consumed in the network and may be viewed as the cost of carrying the traffic. By subtracting this cost from the reward, we penalize the network revenue if the utilized bandwidth is increased without increasing the total reward (as in the case of cycles). The parameter ε must be chosen appropriately and its effect on the final solution will be discussed in detail in Section 2.4.

2.3.3 Node Affinity

We present our algorithm to support the included nodes in the off-line constraint-based routing problem. Suppose an included node list $N_i(\sigma) = \{n_1, n_2, ..., n_k\}$ is given for an OD pair $\sigma = (\sigma_s, \sigma_t)$. Here the nodes $n_1, n_2, ..., n_k$ are intermediate nodes that are required to be on the path from σ_s to σ_t. We replace the original OD pair σ by $k+1$ new OD pairs $\lambda_i^\sigma = (n_i, n_{i+1})$ for $i = 0, 1, ..., k$, where $n_0 = \sigma_s$ and $n_{k+1} = \sigma_t$. Each OD pair has a demand equal to the original demand of σ, namely $D^\sigma_{\lambda_i,s} = D_{\sigma,s}$ for $i = 1, 2, ..., k+1$ and $s \in S_{QoS}$. Note that the original demand $D_{\sigma,s}$ is then set to zero. The revenues of the $k+1$ OD pairs are defined by variables $F_{\lambda_1}, F_{\lambda_2}, ..., F_{\lambda_{k+1}}$, and the following constraint is added to the set of constraints. $F_{\lambda_1} = F_{\lambda_2} = ... = F_{\lambda_{k+1}}$. This constraint ensures that the allocated bandwidth for all OD pairs are equal so that the OD pair σ is allocated the same bandwidth on the path from σ_s to σ_t. Let Ψ_σ denote the set of OD pairs created for a given OD pair σ. If σ does not have an included node list, then $\Psi_\sigma = \{\sigma\}$. For an OD pair σ with included node list $N_i(\sigma) = \{n_1, ..., n_k\}$, $\Psi_\sigma = \{\lambda_1^\sigma, \lambda_2^\sigma, ..., \lambda_{k+1}^\sigma\}$, where $\lambda_i^\sigma = (n_i, n_{i+1})$ for $i = 0, 1, ..., k$, where $n_0 = \sigma_s$ and $n_{k+1} = \sigma_t$. Further more, denote Λ by $\Lambda = \cup_\sigma \Psi_\sigma$.

We list the notation for our algorithm and the modified MCF in the following including label constraint, loop elimination and node affinity.

Table 2.4: Notation of our approach.

Γ	Original set of OD pairs
$L_{max}(n)$	Maximum number of labels allowed on node n
$N_i(\sigma)$	Set of included nodes for the OD pair σ
$\lvert N_i(\sigma)\rvert$	Number of nodes in the $N_i(\sigma)$
$N_e(\sigma)$	Set of excluded nodes for the OD pair σ
$M_{\sigma,l}$	An integer variable corresponding to $Y_{\sigma,l}$
Λ	Set of OD pairs for σ and λ
Ψ_σ	The OD pair created for σ

MCF problem P7:
Input: G=(V,E), $0 \leq g \leq 1, D_{BE,\sigma}, \{C_l : l = 1, 2, ..., |E|\}, L_{max}(n), N_i(\sigma), N_e(\sigma), \varepsilon$

$$maximize \;\; W_{BE} = \sum_{\sigma \in \Lambda} e_{BE,\sigma} F_{BE,\sigma} - \varepsilon \sum_{\sigma \in \Lambda} \sum_{l} Y_{\sigma,l} - \sum_{\sigma \in \Lambda} |N_i(\sigma)| e_{BE,\sigma} F_{BE,\sigma} \tag{2.26}$$

subject to

$$0 \leq F_{BE,\sigma} \leq D_{BE,\sigma}, \forall \sigma \in \Lambda$$

$$\sum_{l \in L_{in}(n)} Y_{\sigma,l} - \sum_{l \in L_{out}(n)} Y_{\sigma,l} = \left\{ \begin{array}{ll} F_{BE,\sigma} & \text{if } n = \sigma_t \\ -F_{BE,\sigma} & \text{if } n = \sigma_s \\ 0 & otherwise \end{array} \right\} \forall n, \forall \sigma \in \Lambda$$

$$Y_{\sigma,l} = M_{\sigma,l} \times (D_{BE,\sigma} \times g), \forall \sigma \in \Lambda, \forall l$$

$$\sum_{\sigma \in \Lambda} Y_{\sigma,l} \leq C_l, \forall l$$

Label limitation:

$$\sum_{\sigma \in \Lambda} \sum_{l \in L_{out}(n)} M_{\sigma,l} \leq L_{max}(n), \forall n$$

Included nodes:

$$F_{\lambda_\sigma} = F_{\lambda_i}, \forall \sigma, 1 \leq i \leq |N_i(\sigma)| \tag{2.27}$$

Excluded nodes:

$$\sum_{l \in L_{in}(n)} Y_{\sigma l} + \sum_{l \in L_{out}(n)} Y_{\sigma l} = 0, \forall n \in N_e(\sigma) \tag{2.28}$$

$$M_{\sigma,l} \in Z, where \;\; Z \;\; is \;\; the \;\; set \;\; of \;\; integers, \;\; 0 \leq M_{\sigma,l} \leq \lfloor 1/g \rfloor, \forall \sigma \in \Lambda, \forall l$$

In the objective function (2.26), we reduce the additional revenue that results from creating additional OD pairs to establish the explicit routes. It is reasonable to route through the included nodes if the series of OD pairs created by those nodes generates more revenues, and reject it if it doesn't. Note that if the network administrator intends to find a route through these nodes regardless of the revenue, he/she can simply raise the earning rate $e_{BE,\sigma}$ of that specific OD pair σ. In constraint (2.27), the carried traffic on all the newly added OD pairs should be equal. In constraint (2.28), the sum of outgoing and incoming flows to the excluded nodes should be zero.

The flow decomposition procedure for the included nodes also needs some modification. For example, we have to concatenate the partial routes that are generated by the set of included nodes to obtain an end-to-end path. In the flow decomposition procedure described before, when we trace a route, a minimum allocated bandwidth along this route is designated to be the bandwidth of this route. In the new algorithm, the bandwidth of the route is the minimum allocated bandwidth among all the newly added OD pairs. The flow decomposition procedure then uses this minimum bandwidth as the bandwidth of the end-to-end route.

2.3.4 Flow Decomposition with Widest Path

The flow decomposition process proposed in [4] belongs to a class of algorithms that are used to obtain routes from a link-based formulation. In the following we describe several other methods as well as a new method which reduces the number of resulting paths.

In [6], the authors suggest directed random walk which randomly traces a route to the destination as mentioned previously. In [4], the method of depth first search is used. In these two approaches, no distinction is made among the outgoing links of a node when the route is traced through the network. In [3], the authors use the widest shortest path algorithm to find the path. The widest shortest path algorithm selects the path with the maximum available bandwidth for the entire path from the set of shortest paths with equal number of hops. Here, we propose a method called the widest path that traces a route from the source to the destination node with the maximum bandwidth along the path. When the route is traced through the network at each node, the outgoing link with the maximum bandwidth is chosen.

In the following, we examine all the methods to trace the routes in Figure 2.2 and compare the number of routes that can be found. In Figure 2.2, an included node list with k nodes are required between source node σ_s and destination node σ_t. Assume the *MCF problem P7* found m parallel paths for each of the newly created OD pairs. One of the path between the additional OD pairs is has a bandwidth of n bits/sec, and the remaining $(m-1)$ paths have a bandwidth 1 bits/sec.

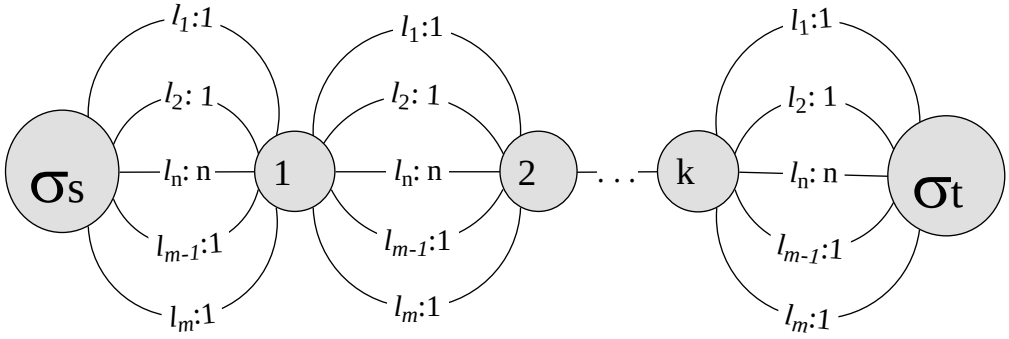


Figure 2.2: Worst case decomposition with node affinity

In the best case there will be m paths between the source node σ_s and destination σ_t. However, the directed random walk may generate up to $m(k+1)-k$ path (if n is large enough). Clearly this is undesirable. On the other hand, using the widest path algorithm the optimal number of m paths will be created.

2.3.5 QoS Approach

For the QoS traffic, we need to differentiate the services into several classes. Some classes may have a stringent bandwidth requirement and therefore must be allocated their requested bandwidth exactly. Some other classes may carry delay sensitive data like voice, and have more stringent constraint on the hop counts.

The approach presented for label constraints, explicit routes and loop elimination in the *MCF problem P7* can also be deployed in the case of QoS traffic. In order to be able to enforce a constraint on the path length (the path hop count), we need to set the granularity to one. As a result, the MCF problem becomes a non-bifurcation problem, and the maximum hop constraint can be easily imposed by introducing the following constraint [14].

$$\sum_{l} Y^{s}_{\sigma,l} \leq H_{max}(s) \times D^{s}_{\sigma} \tag{2.29}$$

In (2.29), $H_{max}(s)$ is the maximum number of hop for the traffic of class s. D^{s}_{σ} is the traffic demand for class s and OD pair σ. $Y^{s}_{\sigma,l}$ is the allocated bandwidth on link l, for class s and OD pair σ. For the non-bifurcation problem, the bandwidth allocated on link l for OD pair σ and class s, $Y^{s}_{\sigma,l}$ is either D^{s}_{σ} or zero. By ensuring that the allocated bandwidth for OD pair σ and class s on all the network links does not exceed $H_{max}(s) \times D^{s}_{\sigma}$, we ensure that the length of the path for this OD pair does not exceed $H_{max}(s)$.

The differentiation of services can be done by choosing different earning rates $e_{s,\sigma}$ for each class. The precedence of different OD pairs can also be distinguished by $e_{s,\sigma}$. However, the traffic of a lower class for an OD pair σ can not preempt the traffic of a higher class for an OD pair σ' (in terms of the allocated bandwidth). The formulation of the QoS traffic is as follows.

MCF problem P8:

Table 2.5: Notation of MCF problem for QoS traffic.

$e_{s,\sigma}$	The earning per unit carried traffic for class s and OD pair σ
F^{s}_{σ}	Total carried traffic for class s and OD pair σ
D^{s}_{σ}	Bandwidth demand for class s and OD pair σ
$Y^{s}_{\sigma,l}$	Allocated bandwidth for class s and OD pair σ on link l
$H_{max}(s)$	Allocated bandwidth for class s and OD pair σ on link l

$$maximize \;\; W_{QoS} = \sum_{s,\sigma} e_{s,\sigma} F^{s}_{\sigma} - \varepsilon \sum_{s,\sigma} \sum_{l} Y^{s}_{\sigma,l}$$

subject to

$$0 \leq F^{s}_{\sigma} \leq D^{s}_{\sigma}, \forall\sigma, \forall s$$

$$\sum_{l \in L_{in}(n)} Y^s_{\sigma,l} - \sum_{l \in L_{out}(n)} Y^s_{\sigma,l} = \left\{ \begin{array}{ll} F^s_\sigma & \text{if } n = \sigma_t \\ -F^s_\sigma & \text{if } n = \sigma_s \\ 0 & otherwise \end{array} \right\} \forall n, \forall \sigma, \forall s$$

$$\sum_l Y^s_{\sigma,l} \leq H_{max}(s) \times D^s_\sigma, \forall s, \sigma$$

$$Y^s_{\sigma,l} = M^s_{\sigma,l} \times D^s_\sigma, \forall \sigma, \forall l, \forall s$$

$$\sum_s \sum_\sigma Y^s_{\sigma,l} \leq C_l, \forall l$$

$$0 \leq M^s_{\sigma,l} \leq 1$$

2.4 Numerical Result

Our experimental results are presented in this section. In Section 2.4.1 we describe the networks for our experiment. We compare the effect of each of the parameters in our MCF problem formulations including ε, granularity g and $e_{s,\sigma}$. The performance under different constraints such as the label constraint and hop constraint, is also evaluated. The synthetic networks for our experiment is described in Section 2.4.1. Some comparisons of the parameters and constraints are shown in Section 2.4.2. A scheme that extracts a MIP solution from an LP solution is also presented in Section 2.4.2. Our QoS algorithm is specifically evaluated with its parameters and constraints in Section 2.4.3.

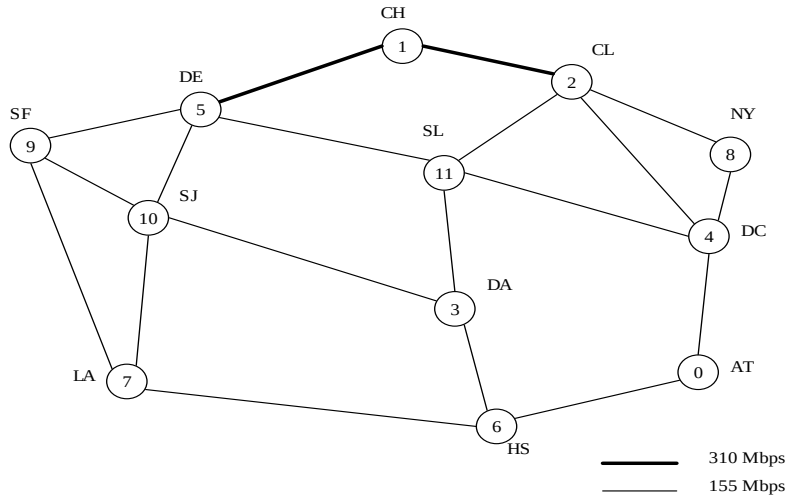


Figure 2.3: Abstract US network

2.4.1 Networks

In our experiment, two types of networks are considered. The first one is the abstract US backbone network with fixed topology and demand. The network has 12 nodes and 38 edges. We denote it as network 1. The network topology along with link capacities are shown in Figure 2.3. The traffic demand for this network can be found in [11].

The second type of networks are randomly generated by a graph generator GT-ITM [9] to produce a synthetic 2-level networks. The 2-level networks are used to simulate the hierarchical networks. The 2-level hierarchical model construct the topology as follows. In the first level, a connected graph is generated with the given node number inside a unit square as in Figure 2.4. The nodes are denoted as cluster in the first level, and is replaced by a smaller connected graph which is generated insider the smaller unit square with the given node number. The top level edges are typically longer than the second level edges. Inside the cluster, we have local access links with a capacity of 250M bits/sec. Between the clusters, we have long distance links with a capacity of 1000M bits/sec.

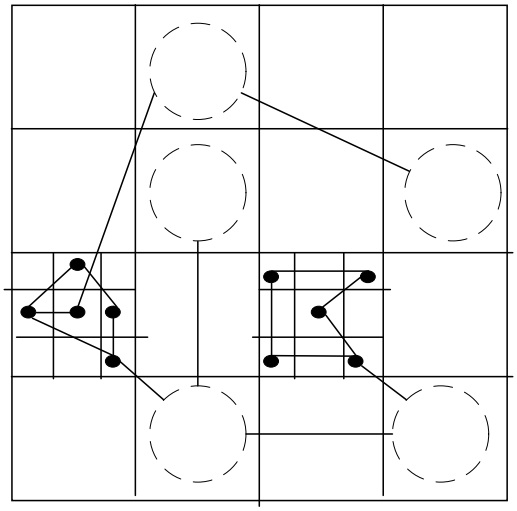


Figure 2.4: 2-level hierarchical topology

For the second type of networks, we randomly generate the demands between each pair of nodes as described in [3]. The generated demands are based on the Euclidean distance of the OD pair as follows. The graph generator randomly places the nodes in a unit square. The distance $\delta(x, y)$ between two nodes x and y can be obtained from their coordinates.

For node x, we generate two random numbers o_x, $d_x \in$ [0,1]. Similarly, for node y, we have o_y, $d_y \in$ [0,1].For each OD pair (x, y), we pick a random number $c_{(x,y)} \in$ [0,1]. Then the demand between x and y is given by

$$\alpha o_x d_y c_{(x,y)} e^{-\delta(x,y)/2\triangle}$$

Here α is the parameter that control the maximum demand. $\triangle$ is the largest Euclidean distance between any pair of nodes. Note that with this allocation, there will be more demand between pairs of nodes that are closer in distance than those that are further apart. The networks generated by GT-ITM are listed as follows.

Table 2.6: Networks generated.

Network	Number of Nodes	Number of Edges	Number of Clusters
network 2.a	40	116	5
network 2.b	40	128	4
network 3	30	108	5
network 4	20	54	2
network 5	10	28	2

2.4.2 Effect of Parameters

2.4.2.1 Choice of ε

The experiment is designed to show the effect and the proper range of ε in network 1, 2.a, 3, 4, 5. The objective function in *MCF problem P3* is replaced by (2.25). We compare the network resources, the total number of paths and loops, and the rate of carried traffic for different value of ε. The earning rate $e_{BE,\sigma}$ is set to one, so that the objective is to maximize the carried traffic in the network.

Setting the ε to 0 is the same as removing the consideration of the utilized network resources in the objective function, and the solution reduces to that of *MCF problem P3*. What is the proper ε in the networks? Since the sum of the bandwidth allocated in the network $\sum_\sigma \sum_l Y_{\sigma,l}$ is greater than the total carried traffic $\sum_\sigma F_{BE,\sigma}$ over all the OD pairs σ, ε should be small enough. Consider a single path establish between an OD pair σ. The bandwidth allocated in the network is the carried traffic $F_{BE,\sigma}$ $\times$ the hop number of this path. If ε is smaller than $\frac{1}{hop\ \ number}$, the choice of ε is fine. Since the demand might split, more than one path might exist between the OD pair σ with different hop number, but the choice of $\frac{1}{max\ \ hop\ \ number}$ is small enough. The ε we suggested is M, which is the number of links in the network.

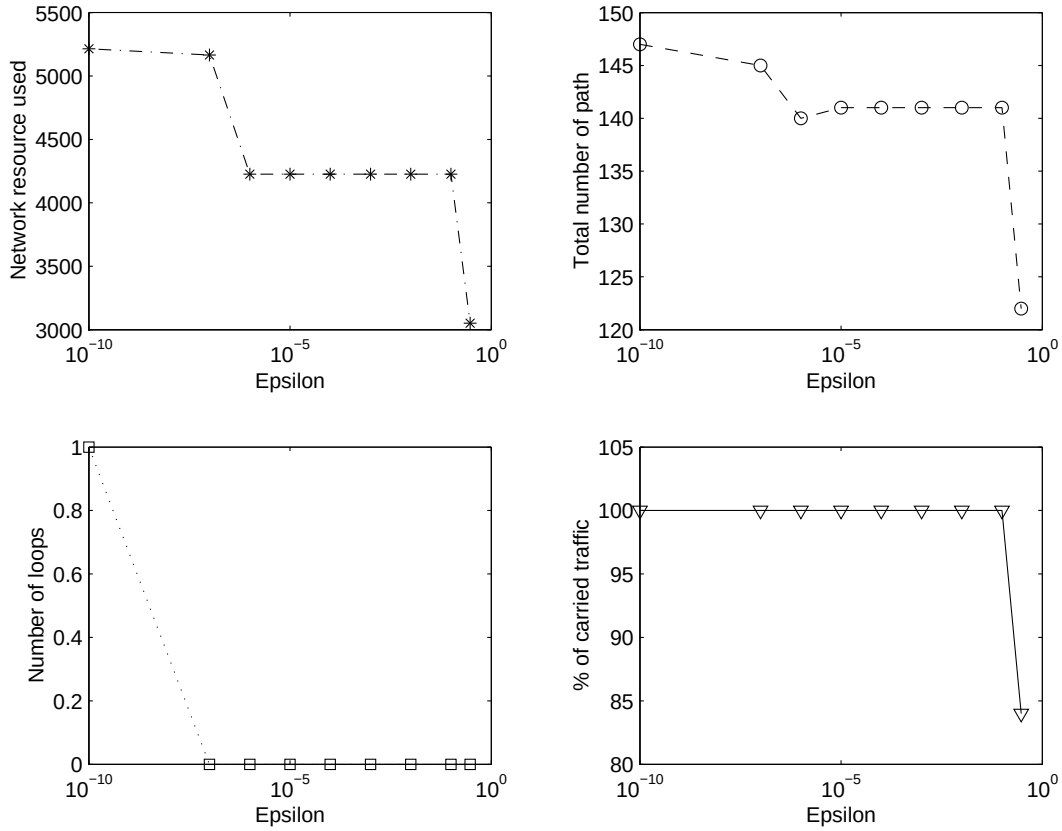


Figure 2.5: Effect of ε in network 1

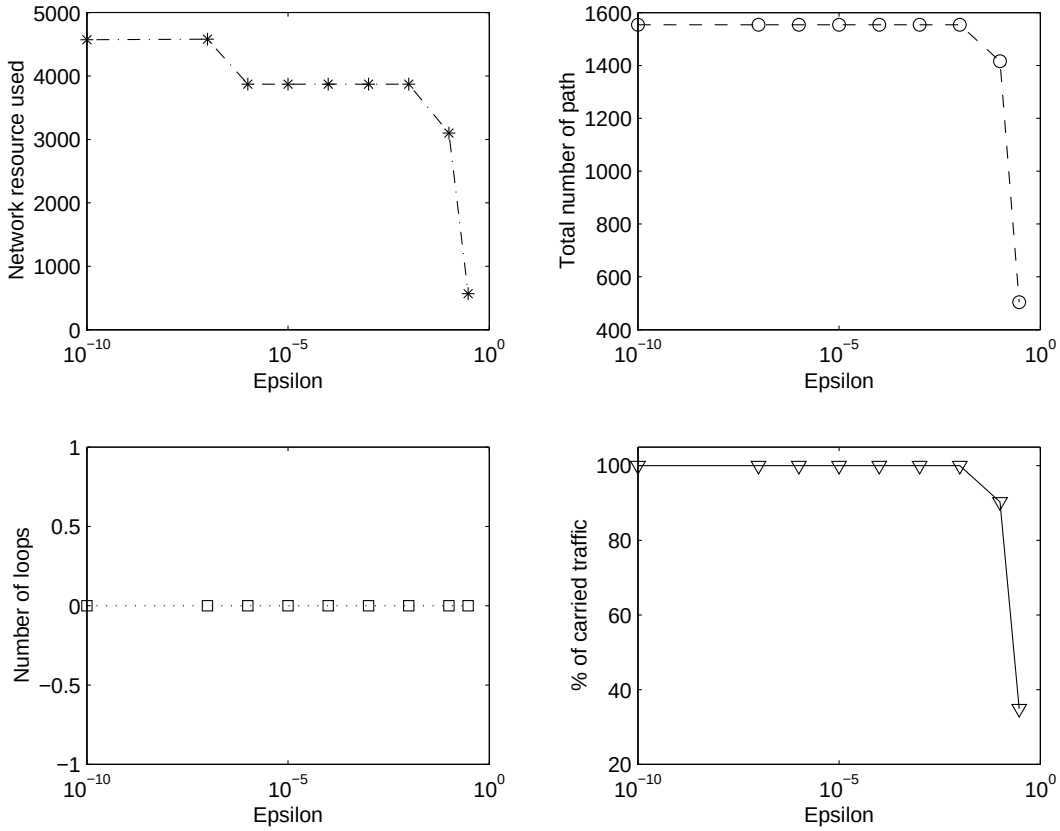


Figure 2.6: Effect of ε in network 2

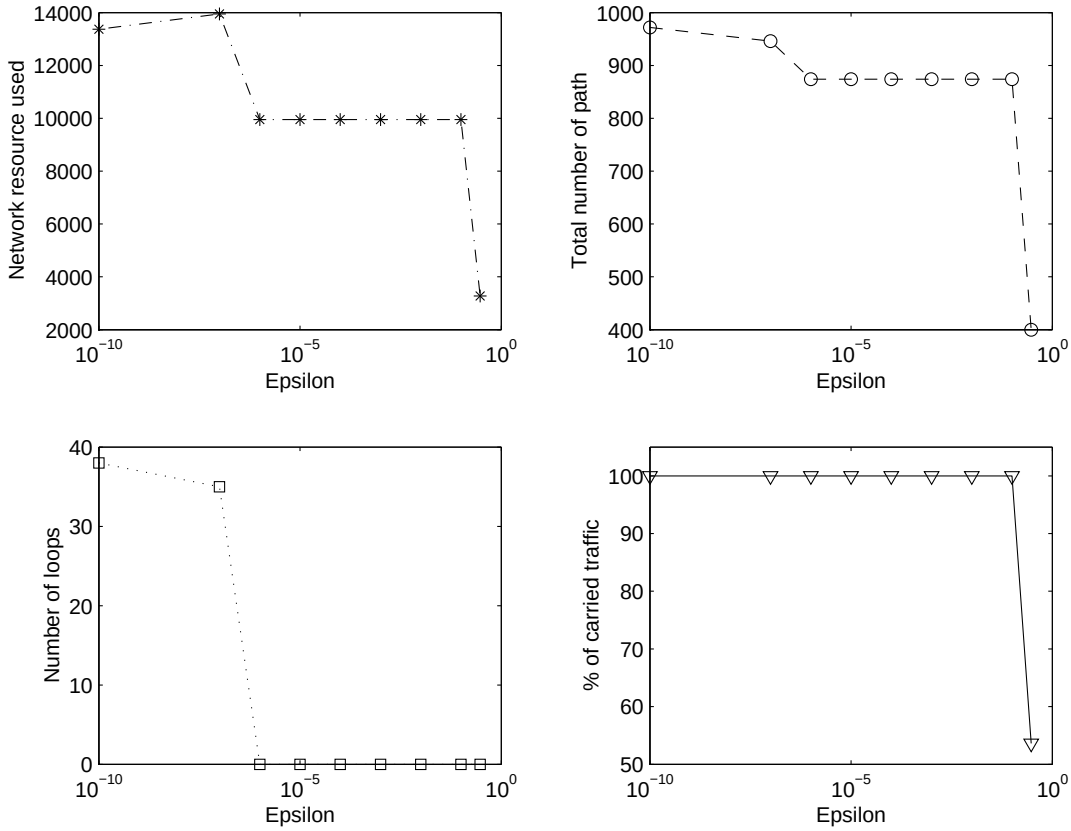


Figure 2.7: Effect of ε in network 3

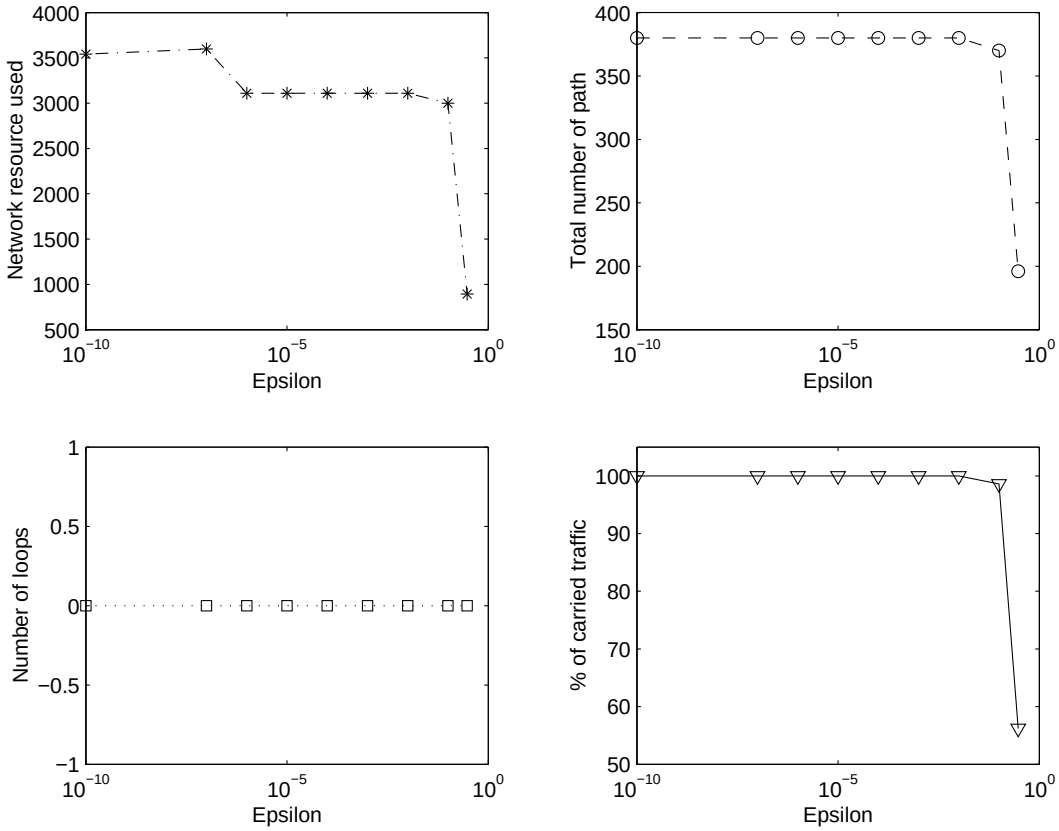


Figure 2.8: Effect of ε in network 4

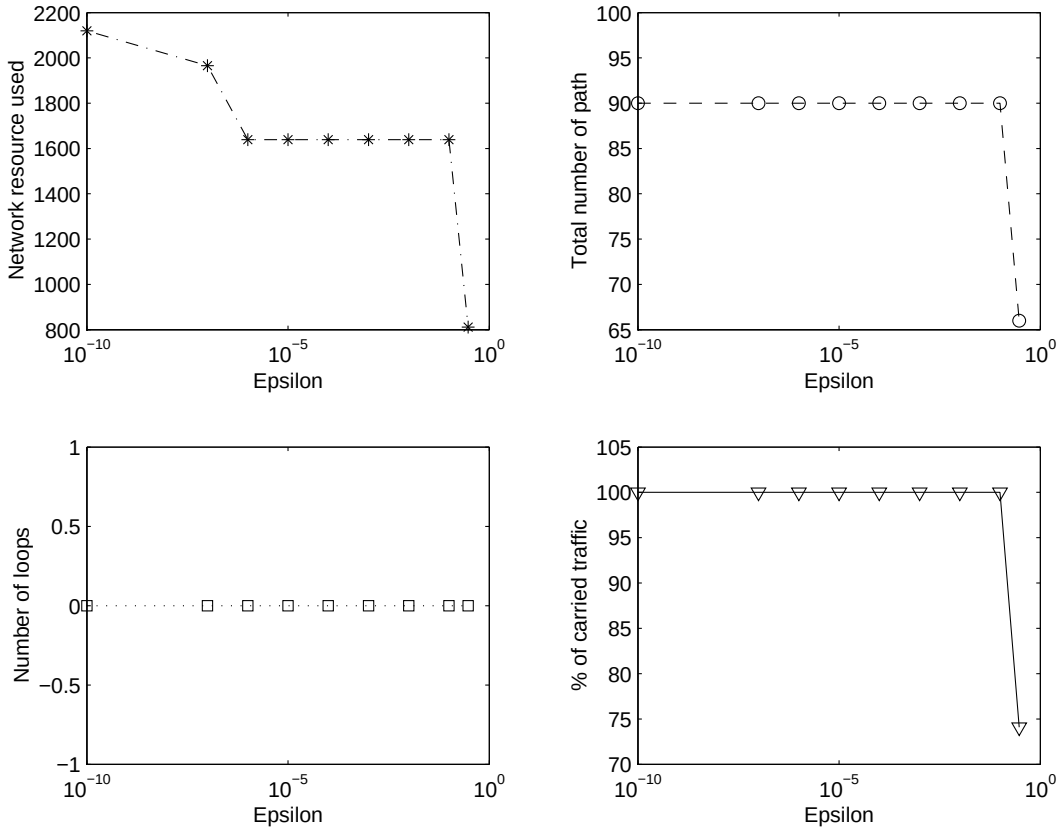


Figure 2.9: Effect of ε in network 5

In Figures 2.5-2.9 we plot the network resources, the total number of paths and the total number of loops in the network, as well as the percentage of carried traffic all vs the value of ε. The percentage of the carried traffic is define as the carried traffic over the traffic demand. It can be seen that in several cases for values of ε close to zero or equal to zero, there are several cycles in the network. As ε increases from zero, the number of cycles also decreases and reaches 0 for ε in the range of 10^{-6} to 10^{-2}. In addition, the utilized resources decrease while the percentage of carried traffic remains unchanged. For large values of ε, (e.g, $\varepsilon > 0.01$) the carried traffic, the number of paths , and the utilized resources all fall off rapidly. This is due to the fact that for large value of ε, the cost of carrying the traffic becomes significant and the revenue is maximized if the carried traffic is reduced. Clearly such values of ε should be avoided. However these figures also indicated that the performance of the algorithm is not very sensitive to the choice of ε. In addition, in cases where no loops exist for $\varepsilon = 0$, (e.g, Figures 2.6 and 2.8) choosing $\varepsilon > 0$ will reduce the utilized network resources, resulting in a more bandwidth efficient routing of traffic.

2.4.2.2 Granularity

In this experiment, we illustrate the effect of granularity on the total carried traffic in networks 1, 2.a, 4, 5. The constraints described by (2.22) and (2.24) are imposed on the *MCF problem P3*. With the integer constraint in (2.24), the problem becomes a MIP problem, and we use CPLEX [10] to solve it. The relative tolerance gap in CPLEX is set to 0.01.

The traffic demands are more than the networks can afford. In order to compare the effect of different granularity on the performance of the algorithm, we choose an offered traffic that is larger than the network can carry. As a result, the carried traffic plotted in Figure 2.10 is the largest amount that the network can carry for the given value of granularity g. Note that for a given granularity g, the demand for an OD pair can be split among $\lfloor 1/g \rfloor$ different paths. Consequently, larger values of g correspond to more stringent conditions on the routing problem, and may result in smaller carried traffic. However, as Figure 2.10 exhibits, this effect is small and the carried traffic does not vary significantly with the choice of granularity g.

2.4.2.3 Performance Comparison

In this experiment, we compare the result of our algorithm *MCF P7* with other results in [5] for both continuous case (solved as LP problem) and discrete case (solved as MIP problem) in network 1. In both cases, we illustrate the result of maximum link utilization rate, total network resources , and number of paths under different algorithms, including Shortest Path (SHP), Equal-Cost Multi-path (ECMP), Traffic Bifurcation (TB), Traffic Bifurcation with hop constraint and excluded nodes (HTB-NA(1)), Mitra's algorithm for Best Effort traffic (MCF3), our algorithm (MCF7) and the one with excluded nodes (MCF7-NA). The maximum link utilization rate is used to measure the maximum fraction of the link capacity

used in the network. We obtain the value of maximum link utilization rate by permitting certain fraction of link capacities to be used as to route the traffic. For the excluded nodes, the constraint in [5] is also used to compare the result.

In Figure 2.11, we measure the performance of MCF3 and MCF7 for the continuous cases by setting the maximum link utilization rate close to that of TB. In our experiment, if the maximum link utilization rate is set to 0.88, some traffic will be dropped. Thus, we evaluate the performance on the the rate 0.885 while all the traffic can be carried. The maximum link utilization rate of the excluded node is adjust to the value that the network resources is similar to that of the shortest path. We can see that the total network resources of MCF7 is close to that of TB, and much better than the result of MCF3. The difference between TB and MCF7 is that TB has to perform another optimization to reduce the network resources and MCF7 doesn't. Though the network resources of MCF7 is not as good as SHP or ECMP, it guarantee that the link capacity is not overutilized. For the number of paths, MCF7 has less paths than TB. The overall performance of MCF7 and MCF7-NA is close to TB and HTB-NA(1) in the continuous case.

In Figure 2.12, we illustrate the result of maximum link utilization rate, total network resources , and number of paths under different value of granularity for different algorithms for the discrete cases. The maximum link utilization rate of MCF7(g) is set to the same value as the TB(g)'s in order to compare. Also, the maximum link utilization rate of MCF7(g)-NA ,which contained the excluded nodes as in [5] is adjusted to the value such that the total network resources is the same as SHP's. We show the amount of total network resources of MCF7(g) and MCF7(g)-NA under different granularity. However, the network resources of TB(g) can't be obtained precisely in [5]. Since both algorithm of MCF7(g) and TB(g) minimize the network resources, and carried the traffic completely, we surmise they consume the same network resources. In the comparison of number of paths, we can see that both MCF7(g) and MCF7(g)-NA have less number of paths than that of TB(g).

2.4.2.4 LP to Integer Approach

As discussed previously, MIP problems are computationally extremely complex. Thus we seek another approach to find a solution which is computationally less complex, although the solution may not be optimal. Such a suboptimal solution will be acceptable if it's performance is close to that of the optimal solution. Based on the observation that most demand can be satisfied in one or two routes, we propose an approach that seeks an LP solution first and later modifies it to obtain a MIP solution. We solve the original MIP problem, but do not enforce the variables $M_{\sigma,l}$ to be integer. As a result,$M_{\sigma,l}$ may be a real number. We then round down the $M_{\sigma,l}$ to the nearest integer to obtain a MIP solution.

In this experiment, we compare the difference of the LP-to-Integer approach and the LP approach for various demand sets. The granularity g is set to one to see the worst performance. In this case, if the demand is split, the traffic demand will be dropped by the LP-to-Integer approach. The α control the amount of the demands as described in Section 2.4.1. In Figure 2.13, we plot the percentage of carried traffic vs the value of α. For higher

α, the demands will increase, but the rate of the carried traffic will drop. It can be seen that the LP-to-Integer approach doesn't carry as much traffic as LP approach in the worst case, but it stays quite close to the LP. The biggest gap between the LP-to-Integer and LP approach is around 4%, which is quite acceptable for saving lots of computational time comparing to the MIP problem.

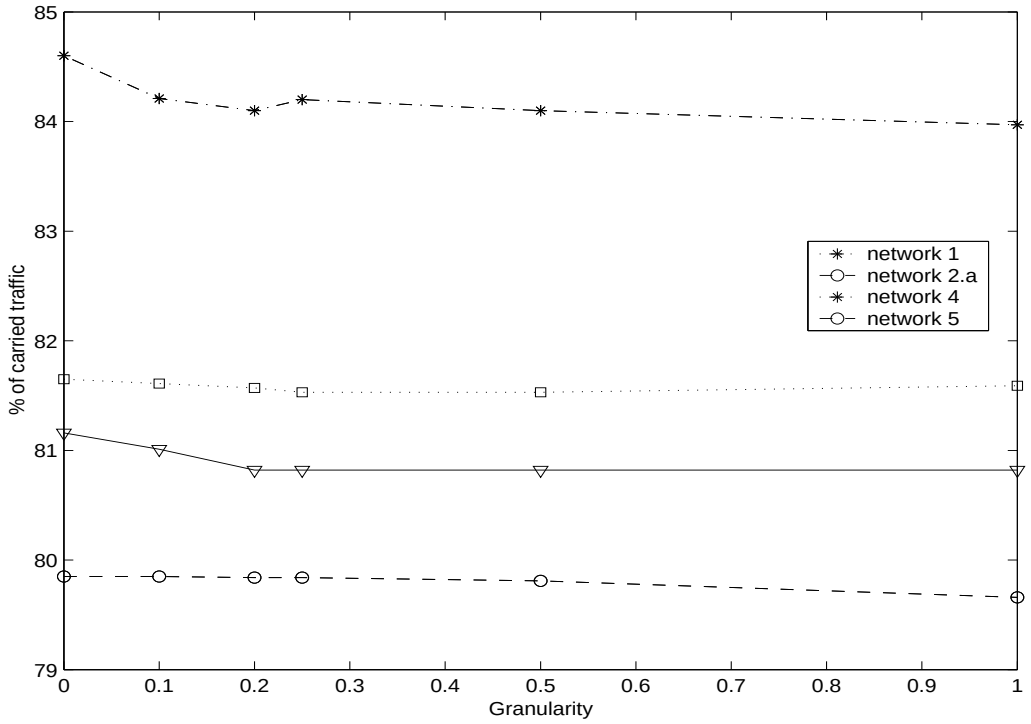
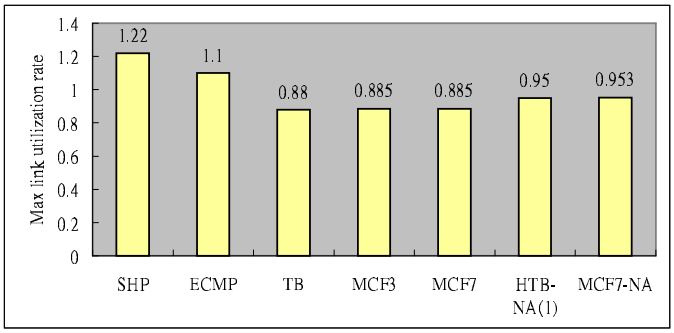
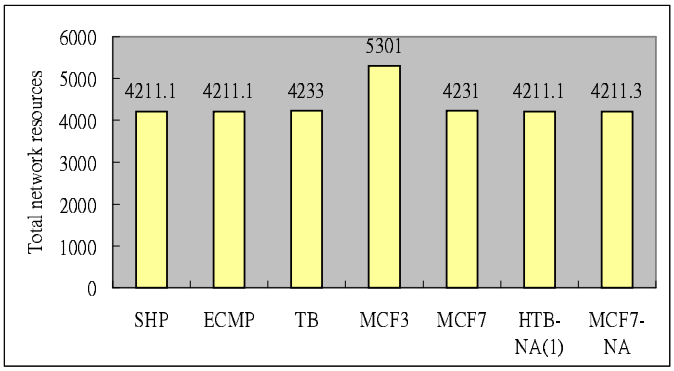


Figure 2.10: Effect of g





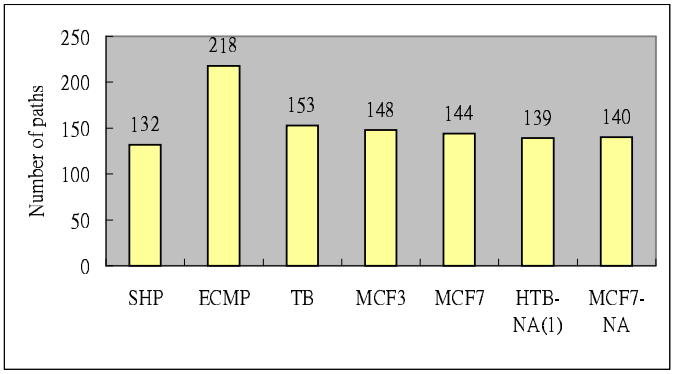
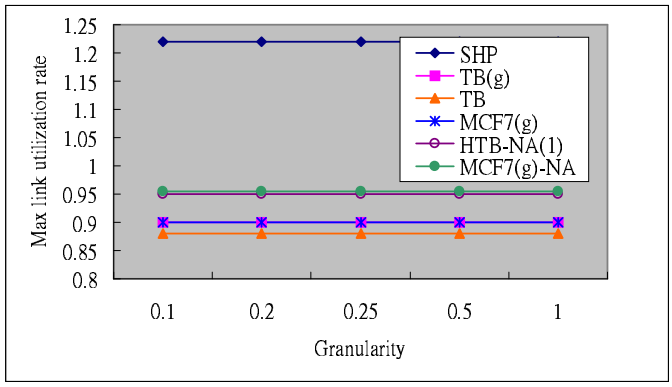
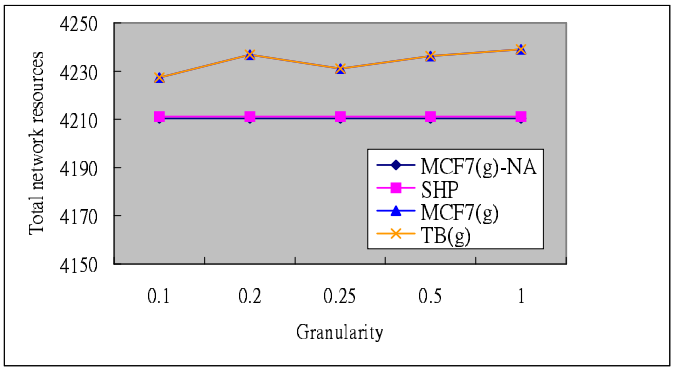


Figure 2.11: Comparison of LP problem





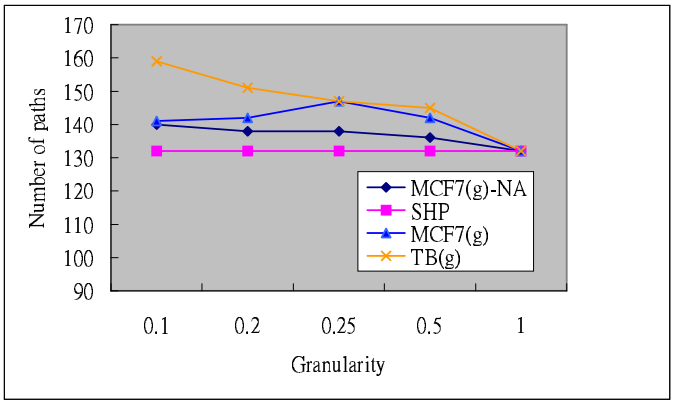


Figure 2.12: Comparison of MIP problem

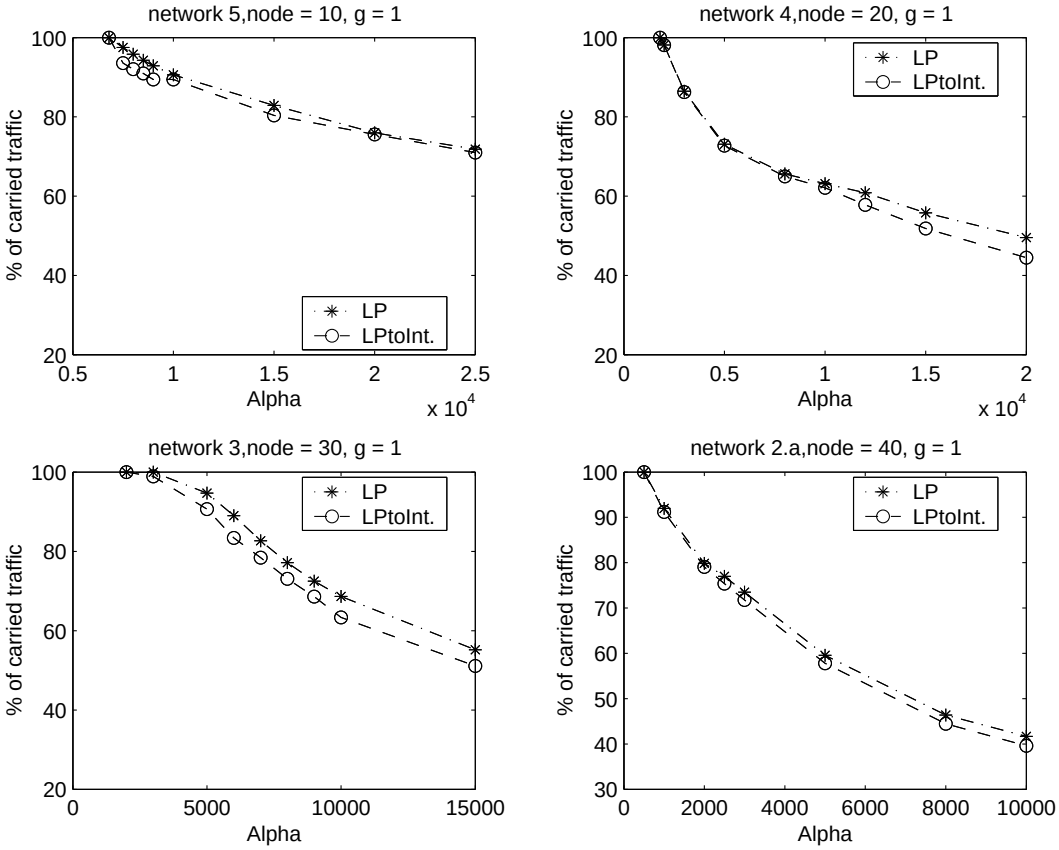


Figure 2.13: % of carried traffic vs alpha

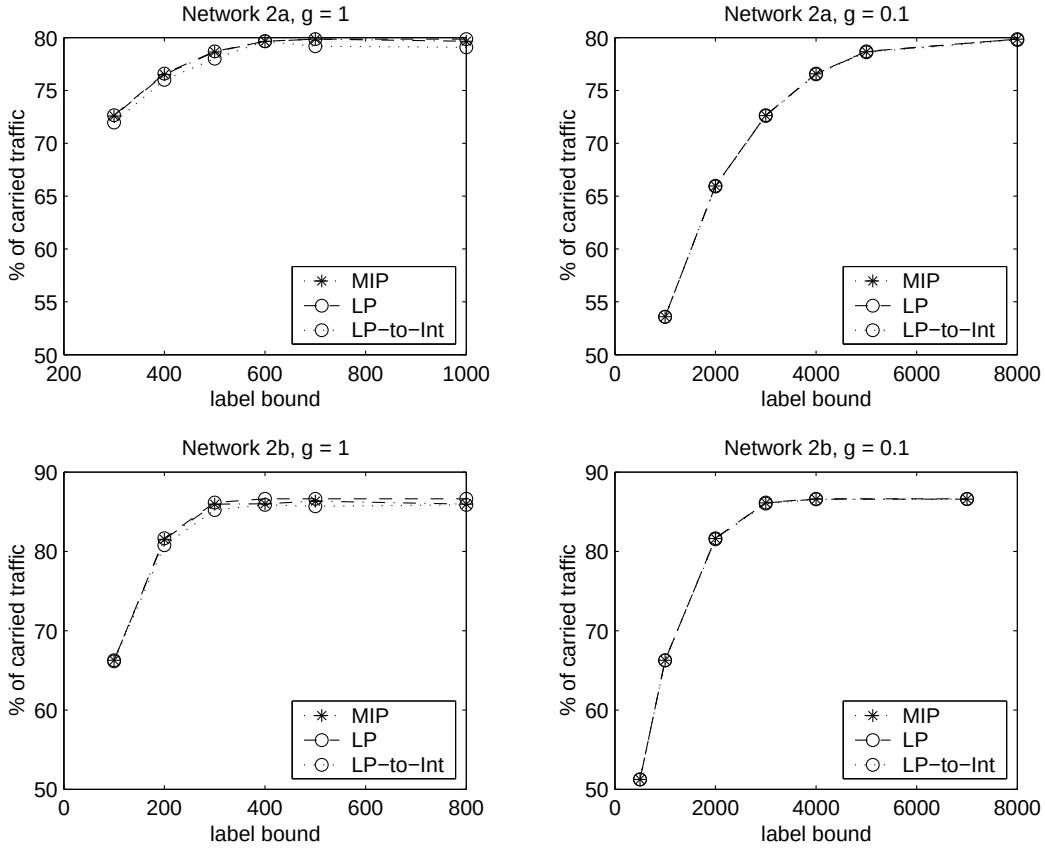


Figure 2.14: Effect of label constraint

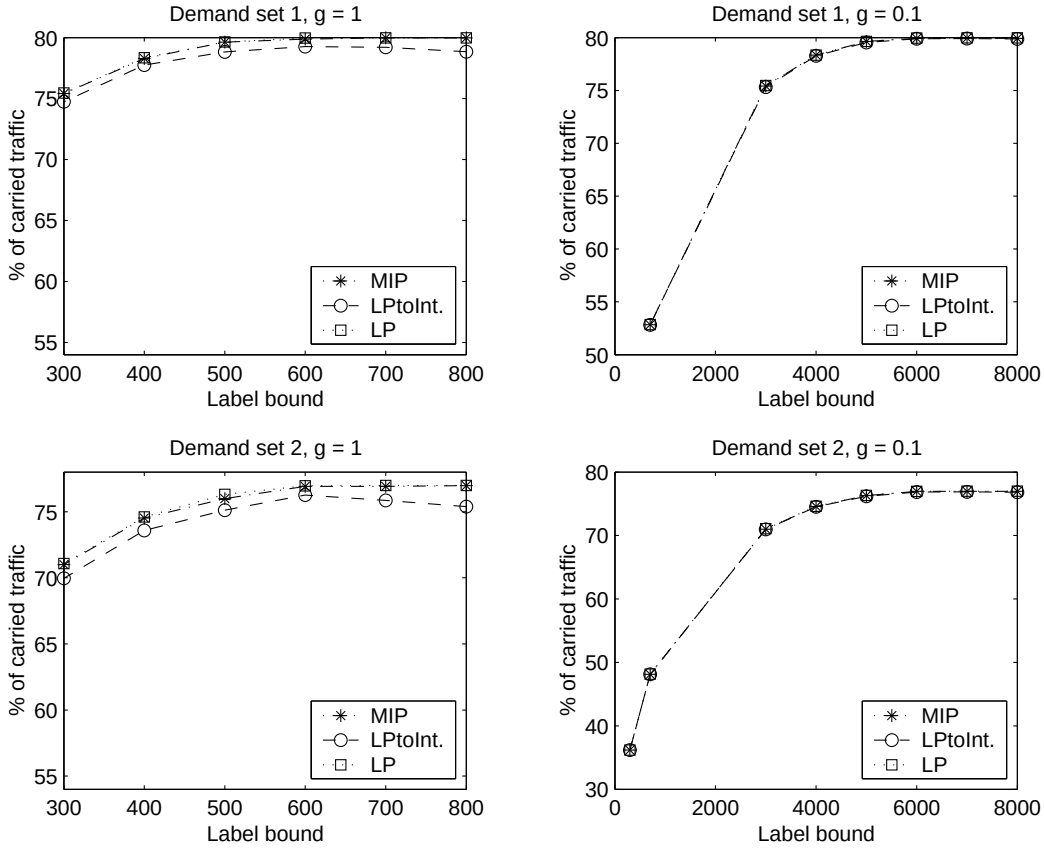


Figure 2.15: Effect of label constraint in network 2a

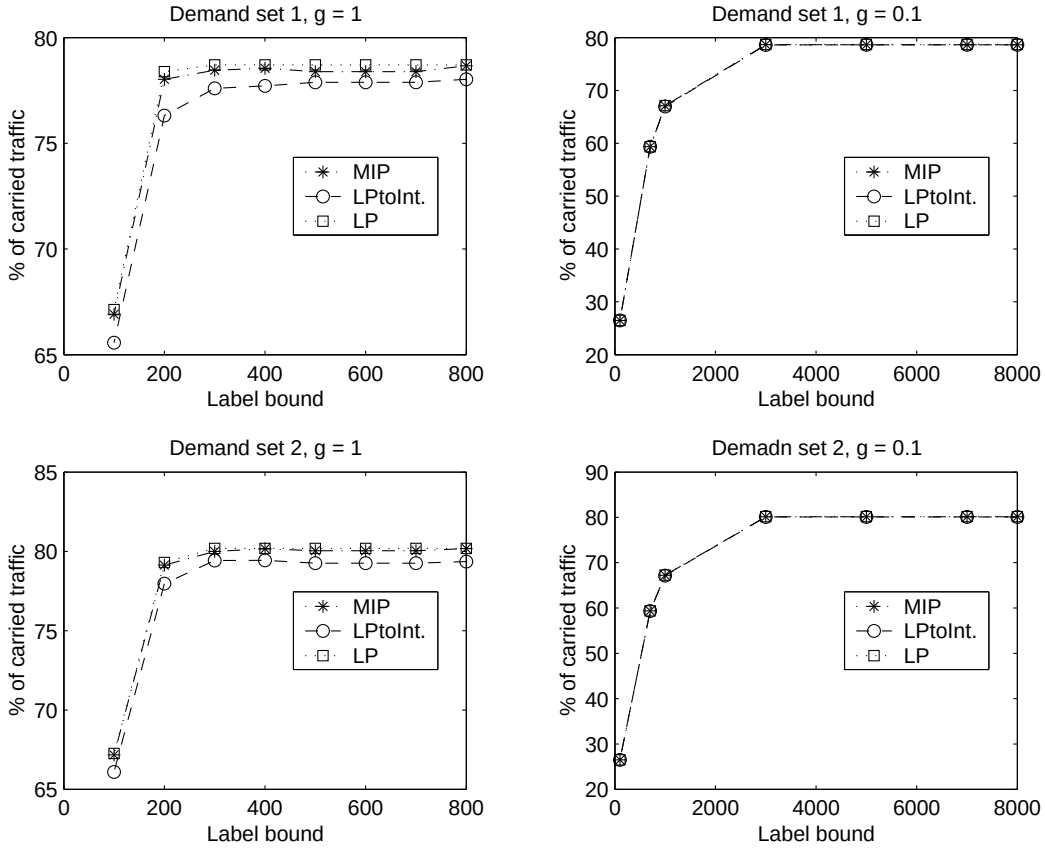


Figure 2.16: Effect of label constraint in network 2b

2.4.2.5 Label Constraints

In this section, we investigate the effect of label constraints on the performance of our algorithms. The results are presented in Figure 2.14 for the two networks 2.a and 2.b on two values of granularity $g = 0.1$ and $g = 1$. All of the nodes are set to have the same limit on the number of label.

In Figure 2.14, we plot the label constraint vs the percentage of the carried traffic. It can be seen that if the label constraints are much higher than the labels used on each node, there is not effect on the rate of the carried traffic. As the label constraints are below the number of labels used in some of the nodes, the rate of the carried traffic drops slightly. It'll drop rapidly if the label constraints are lower than the labels used on most of the node.

We compare the effect of the different granularity $g = 0.1$ and $g = 1$ in Figure 2.14, it can be seen that for $g = 0.1$, the rate of the carried traffic drops earlier in the higher label constraints. This is because in constrain (2.23), we limit the number of labels by measuring the $M_{\sigma,l}$ which is also the maximum number of flows that can be split for an OD pair σ. For $g = 0.1$ the flows between the OD pair σ is allowed to split up to 10 flows. As a result, the flows will reach the label bound earlier. Note that if the flows do not split, the flow decomposition process will aggregate the flows into one path, and thus reduce the amount of the actual routes or labels used. Thus the choice of granularity g also decides the efficiency of the label constraint.

In Figure 2.14, we also compare the performance of LP, MIP and LP-to-Int approach on the percentage of the carried traffic. For $g = 1$, it can be seen that the LP solution has the best performance in carrying the traffic. The MIP solution is next, and then LP-to-Int. But for $g = 0.1$, the performance of LP, MIP and LP-to-Int are very close.

In Figure 2.15 and 2.16, the experiment is repeated in network 2.a and 2.b each with different demand sets. We can see that the results are very similar.

2.4.3 QoS Result

2.4.3.1 Networks

The experiment illustrates the performance of the QoS traffic under different $e_{s,\sigma}$ and hop constraint $H_{max}(s)$. We run the algorithms for networks 1, 2.a, 3, 4, 5. Four service classes are considered: Classes A, B, C and BE. The QoS traffic is routed with the BE traffic. Different classes are differentiated by the value of $e_{s,\sigma}$. The original demand sets for each of the networks are divided equally between the four classes.

2.4.3.2 Effect of $e_{s,\sigma}$

We use different earning rate $e_{s,\sigma}$ to distinguish different class of service. In the experiment, we ignore the effect of the earning between OD pair σ. For class A, B, C and BE we set the $e_{s,\sigma}$ to 30, 20, 10 and 1 respectively. In the following result, we examine the difference in routing the QoS traffic together with the BE traffic and routing both of them in two different stages.

In Figures 2.17 - 2.21, the BE traffic is routed together with the QoS traffic. We plot the percentage of the carried traffic and the average hop number vs the percentage of the demand for network 1, 2a, 3, 4, 5. The performance of four different classes are measured. We also show the result for both LP and the LP-to-Int solution. It can be seen that the earning rate $e_{s,\sigma}$ imposed on different classes result to different drop precedent. The lower class BE is dropped first for all networks as we increase the traffic demands, and then followed by Class C, Class B. The result of the LP-to-Int solution is very close to the LP for the synthetic 2-level networks. However, there are some round off effect for the abstract US network. In some cases, the rate of carried traffic for Class A is lower than Class B, but the saving time for the computational complexity is warrantee for the LP-to-Int approach. The average hop number has the similar behavior with the rate of the carried traffic. As we increase the demand, the average hop number starts to drop. This is because the route with the large hop number utilized more network resources and is undesirable in the case of the deficient resources.

In Figures 2.22 - 2.26, the BE traffic is routed in another stage and use the residual capacity of the QoS traffic. Comparing the result with Figure 2.17 - 2.21, we found out that the rate of carried traffic for BE traffic is higher while the BE traffic is routed in another stage. In some cases, the rate is even higher than Class C.

The result of the LP-to-Int solution is very close to the LP for the synthetic 2-level networks,and result to some round off effect for the abstract US network. The average hop number has the similar behavior with the rate of the carried traffic.

2.4.3.3 Comparison of $H_{max}(s)$

This experiment intend to test the performance of the QoS traffic under the hop constraint in network 2.a, 3, 4, 5. We use the hop constraint for Class A and Class B traffic only. Two different demand set are chosen for each of the network to compare the result on different network loading. One of the demand set has less demands in that the network is sufficient to carry all the traffic demands. The other demand set is more than the network can affordable, and result to the dropping of demands for Class C and BE.

In the Figure 2.27 - 2.30, we plot the percentage of the carried traffic and the average hop number vs the hop bound with less load. The result of the LP-to-Int approach is compared with the LP solution. It can be seen that if the hop bound is beyond the maximum number of hop in the network, the result is as if no hop constraint imposed. If we decrease the hop bound , the carried traffic of Class A and Class B will start to drop. The rate of the carried traffic of the LP-to-Integer solution drop around 20 to 30 percentage than the LP solution. This implies that more demands will split if the path has to route to the destination node in certain hops.

In the Figure 2.31 - 2.34, we show the performance of the demands set with more load. It can be seen that as the demands of Class A and Class B dropped, the networks have more resources to route Class C and Class BE traffic, so the rate of carried traffic will raise for Class C and Class BE.

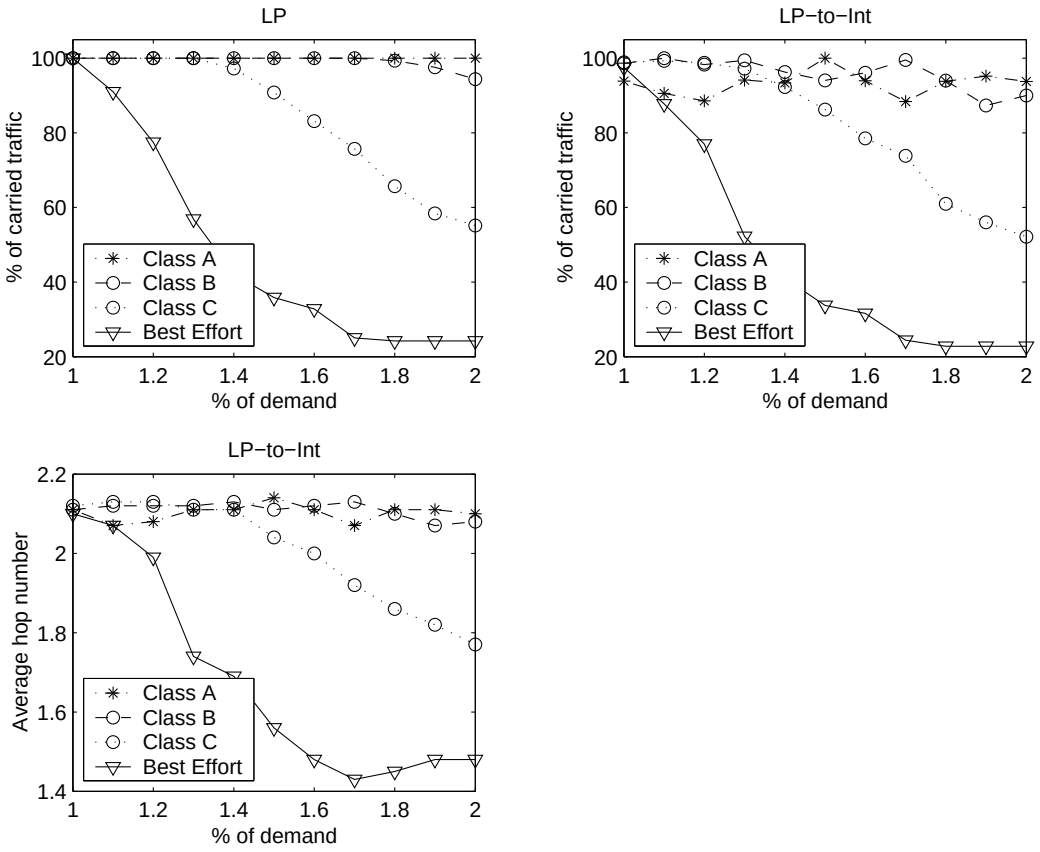


Figure 2.17: Effect of $e_{s,\sigma}$ in network 1 : one stage routing

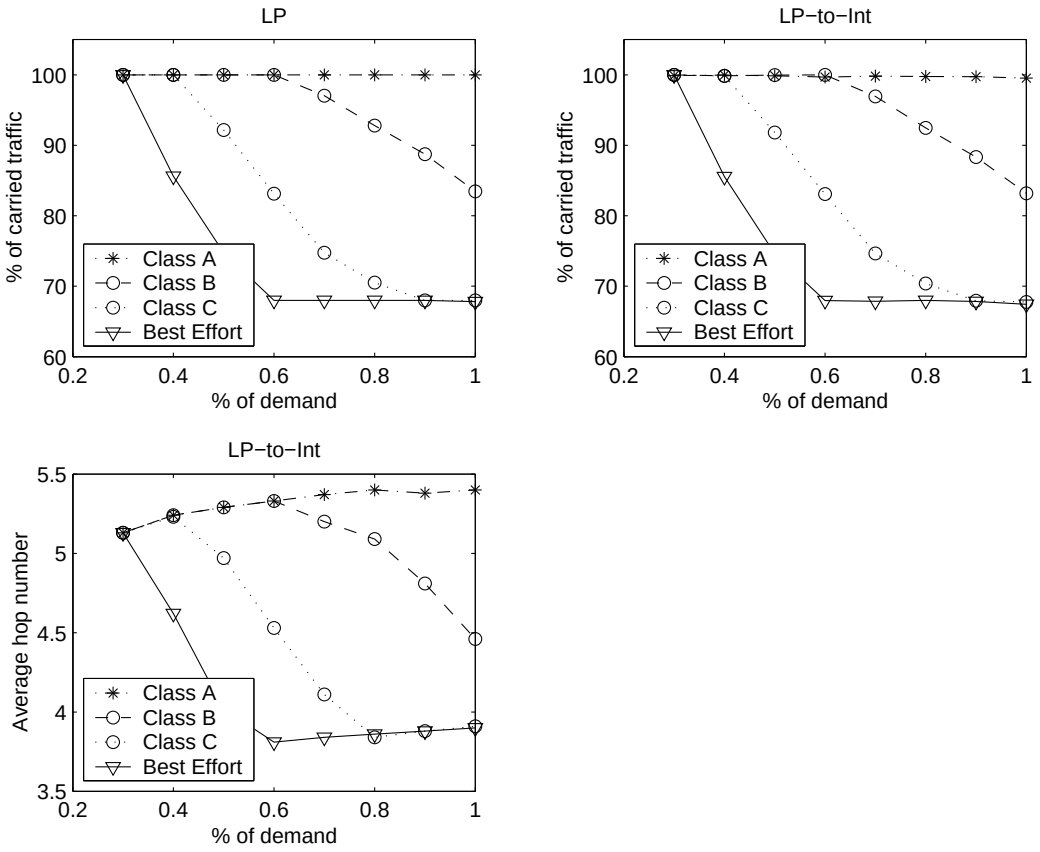


Figure 2.18: Effect of $e_{s,\sigma}$ in network 2 : one stage routing

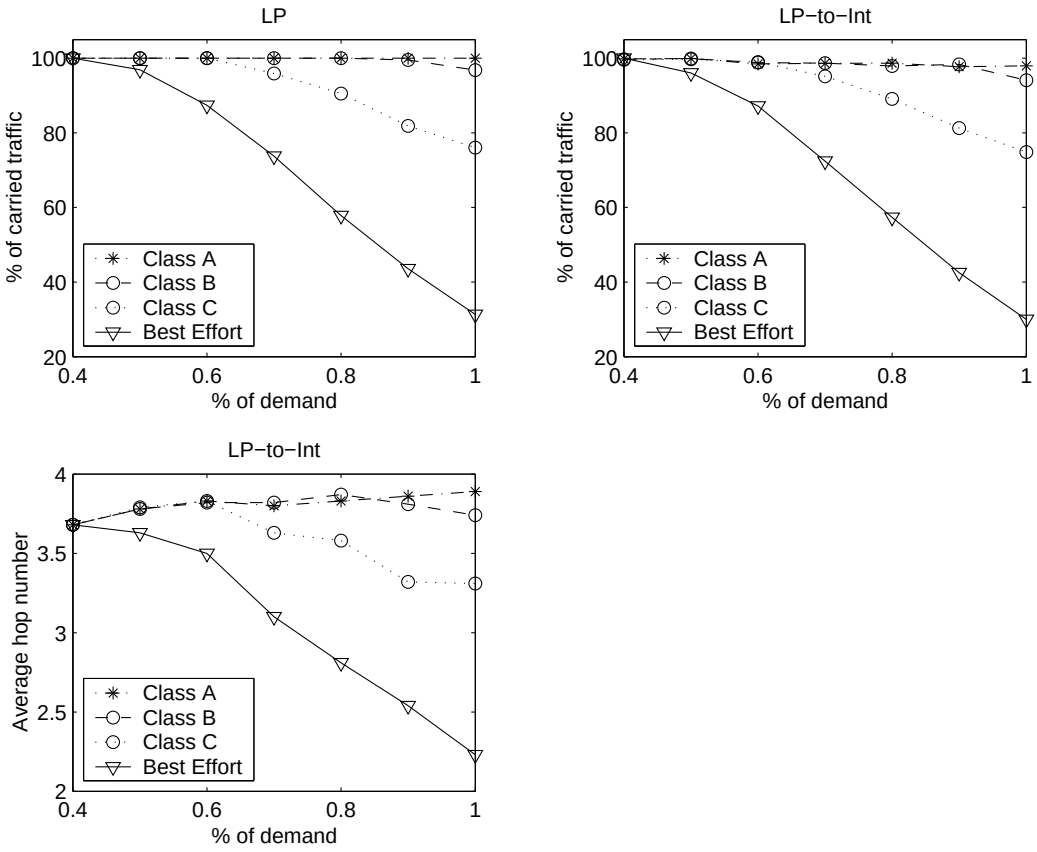


Figure 2.19: Effect of $e_{s,\sigma}$ in network 3 : one stage routing

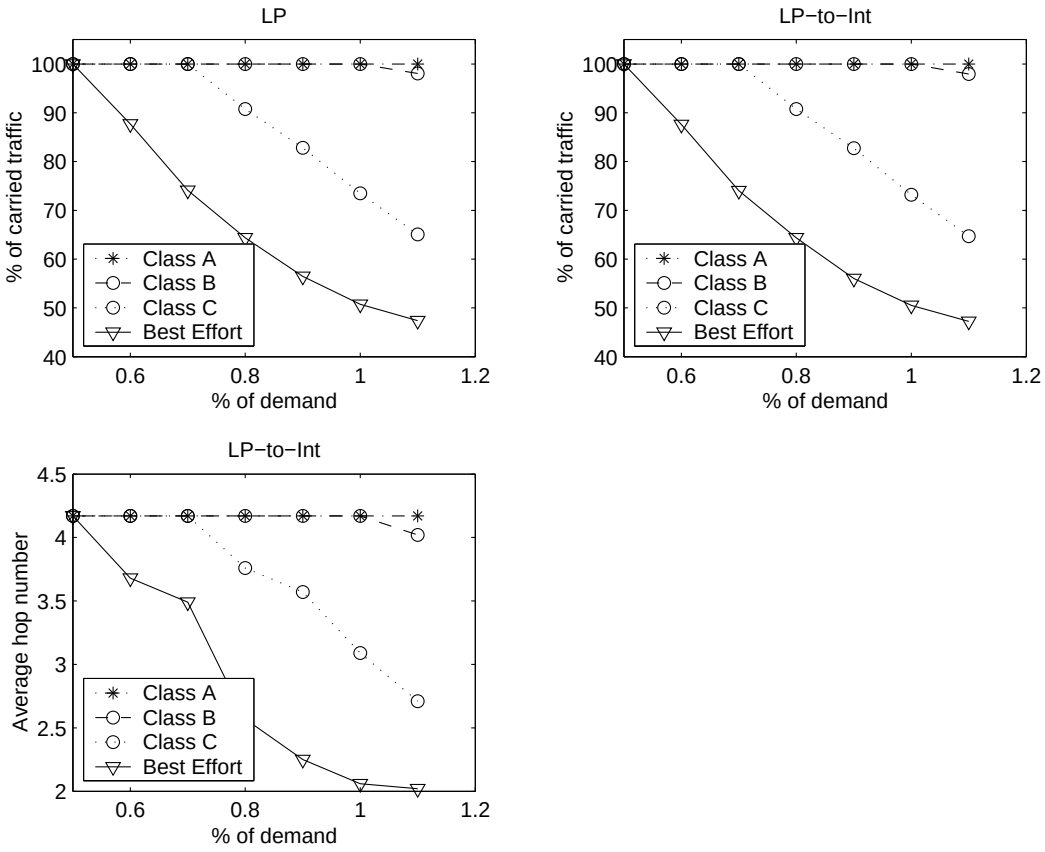


Figure 2.20: Effect of $e_{s,\sigma}$ in network 4 : one stage routing

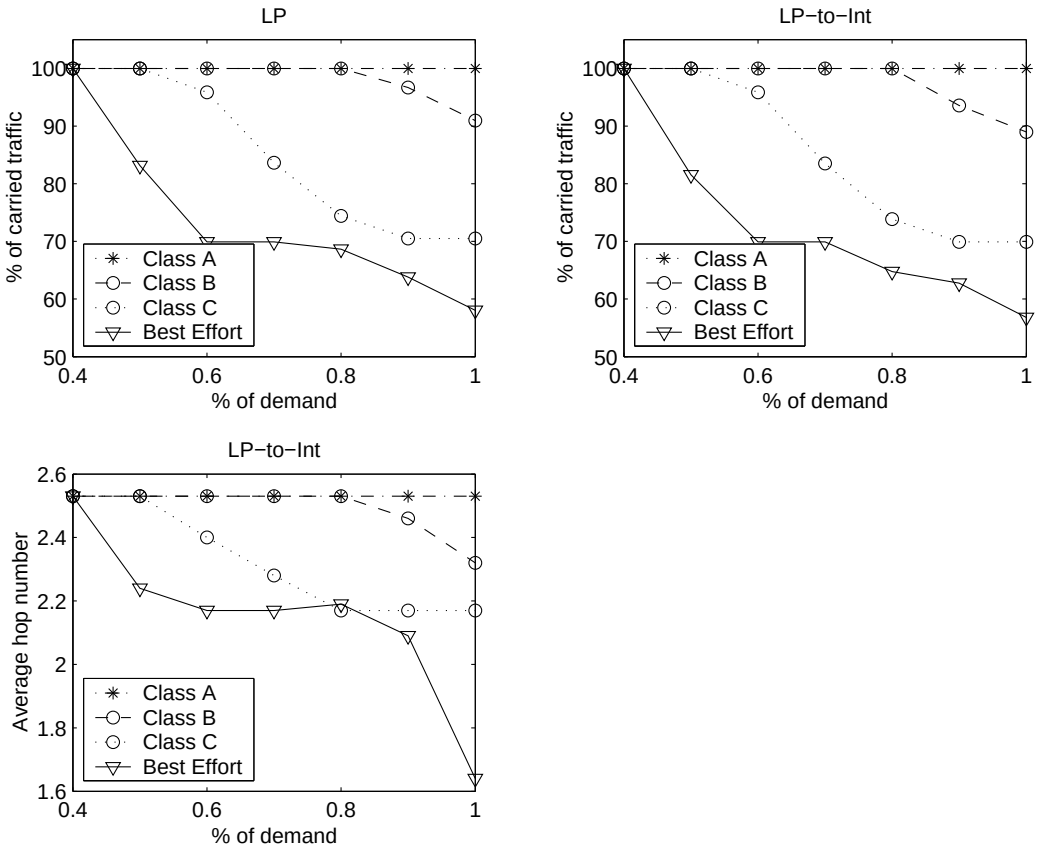


Figure 2.21: Effect of $e_{s,\sigma}$ in network 5 : one stage routing

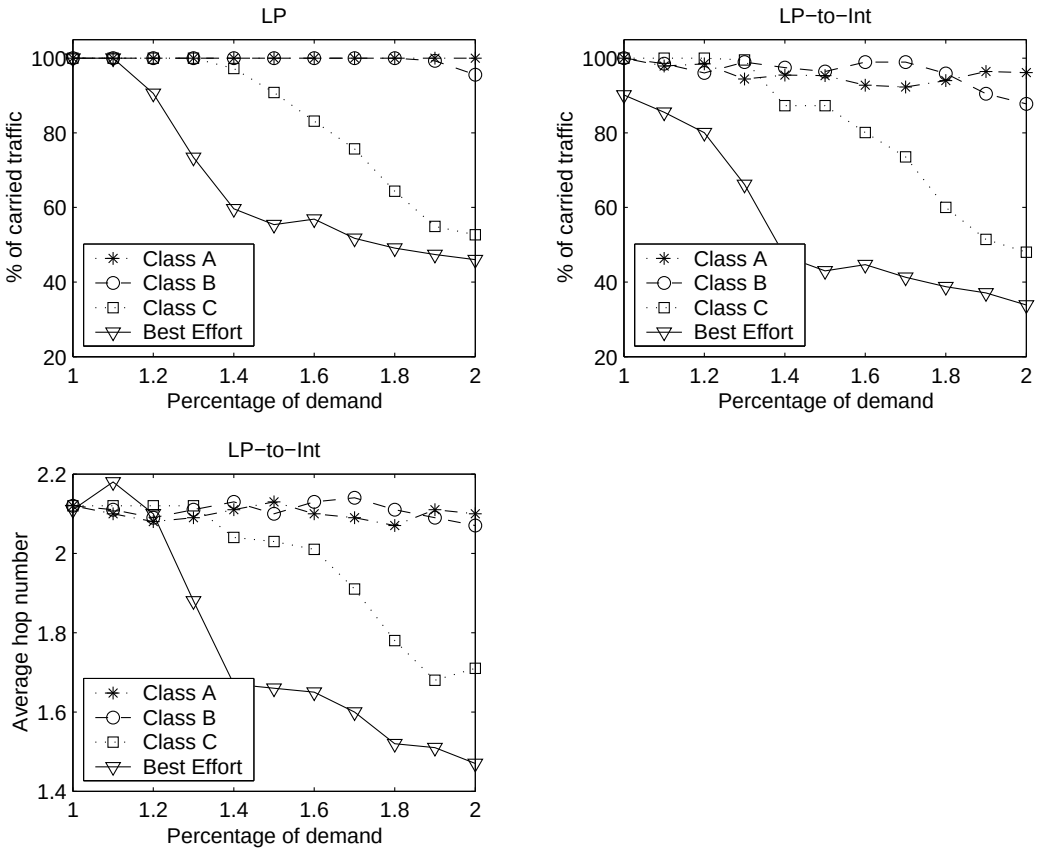


Figure 2.22: Effect of $e_{s,\sigma}$ in network 1 : two stage routing

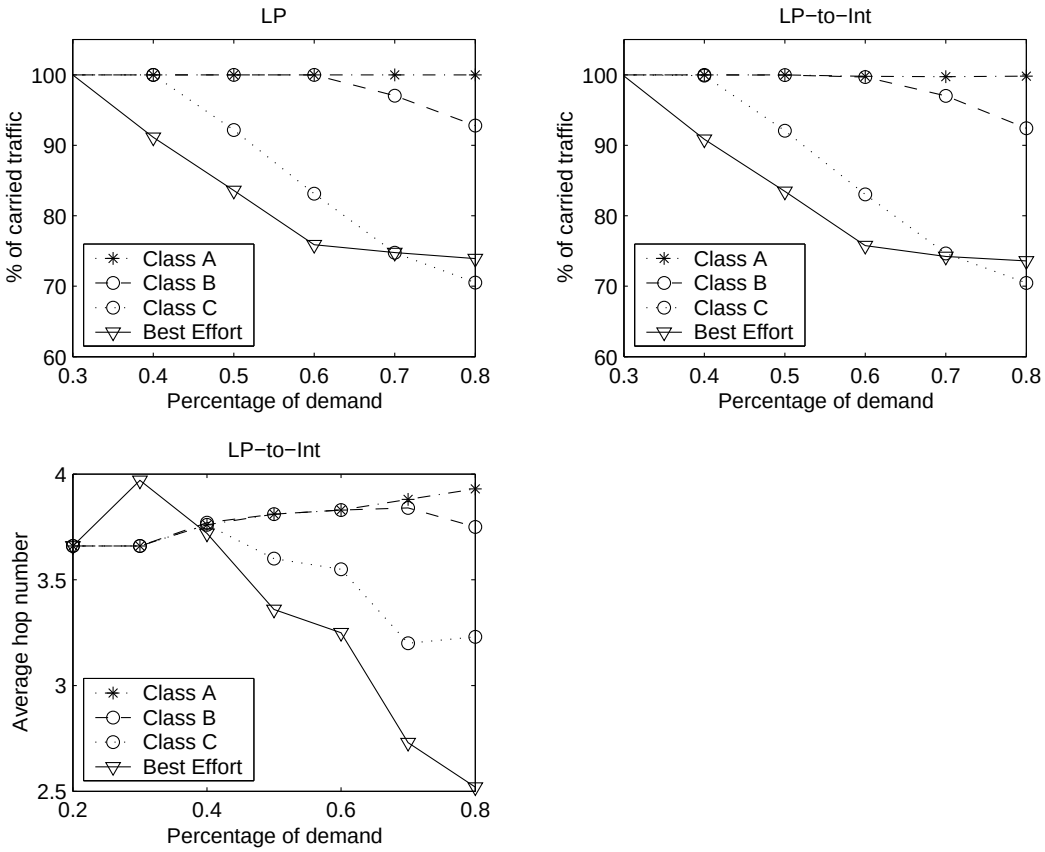


Figure 2.23: Effect of $e_{s,\sigma}$ in network 2 : two stage routing

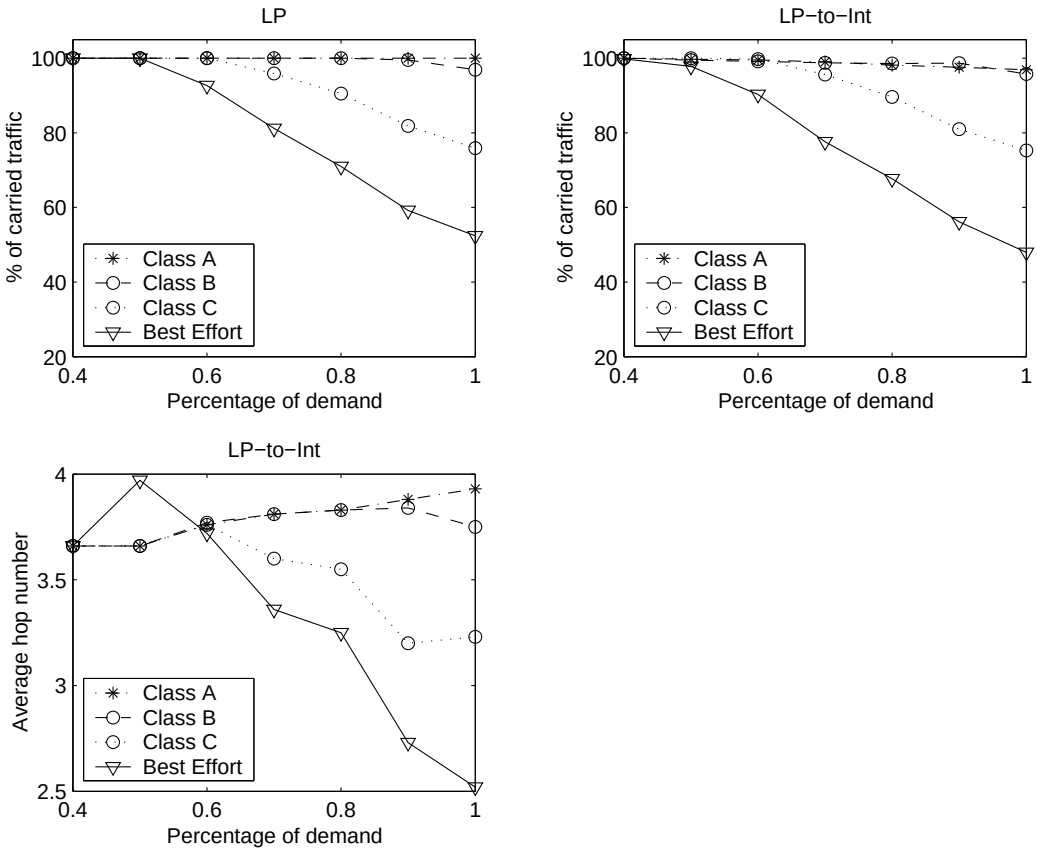


Figure 2.24: Effect of $e_{s,\sigma}$ in network 3 : two stage routing

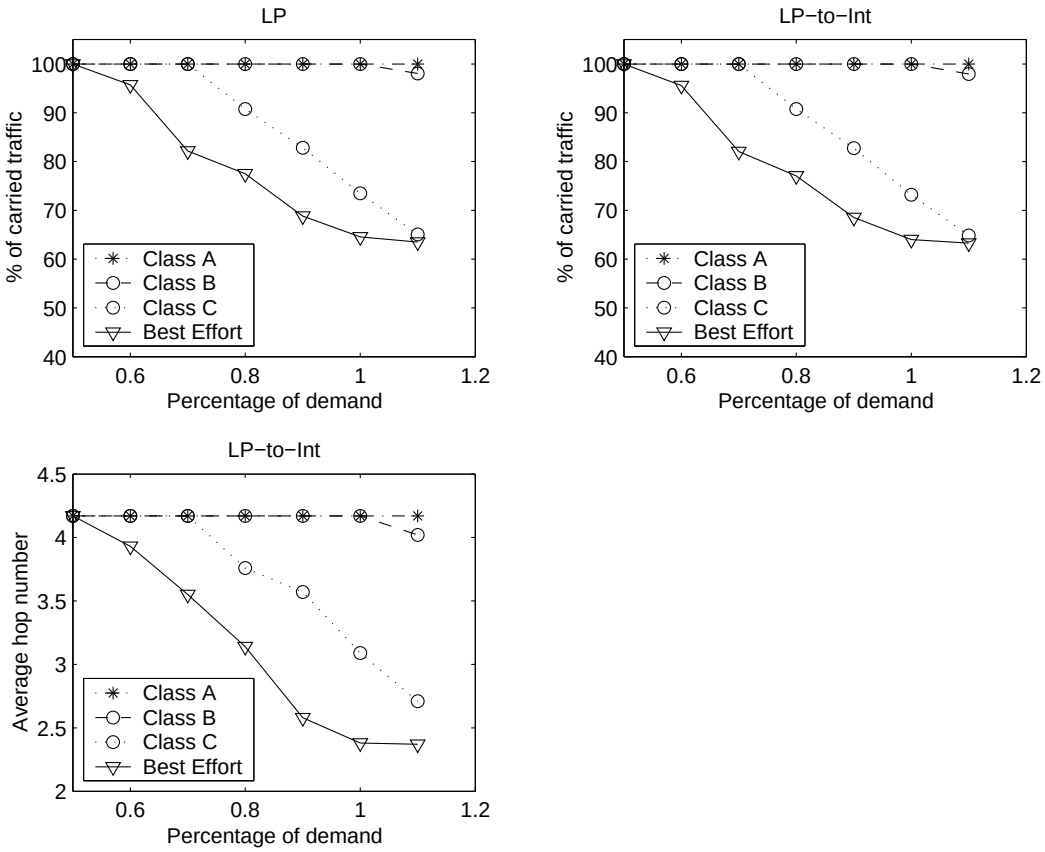


Figure 2.25: Effect of $e_{s,\sigma}$ in network 4 : two stage routing

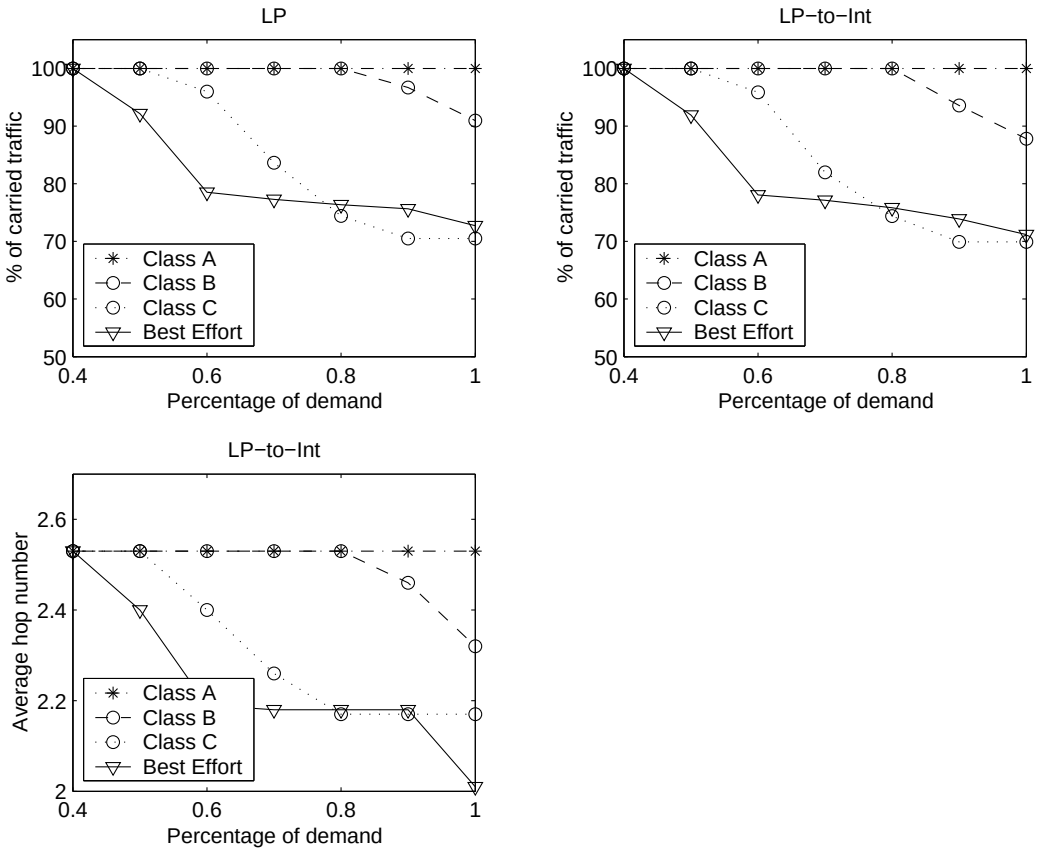


Figure 2.26: Effect of $e_{s,\sigma}$ in network 5 : two stage routing

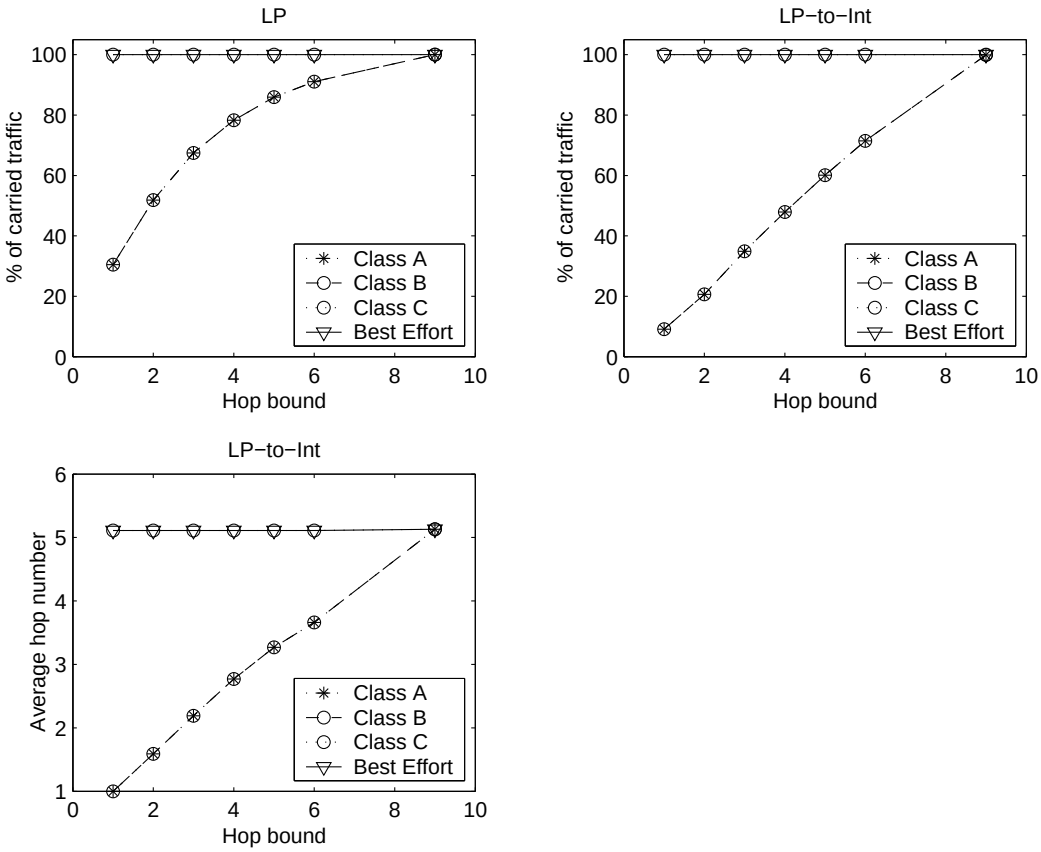


Figure 2.27: Effect of $H_{max}(s)$ in network 2 with light load

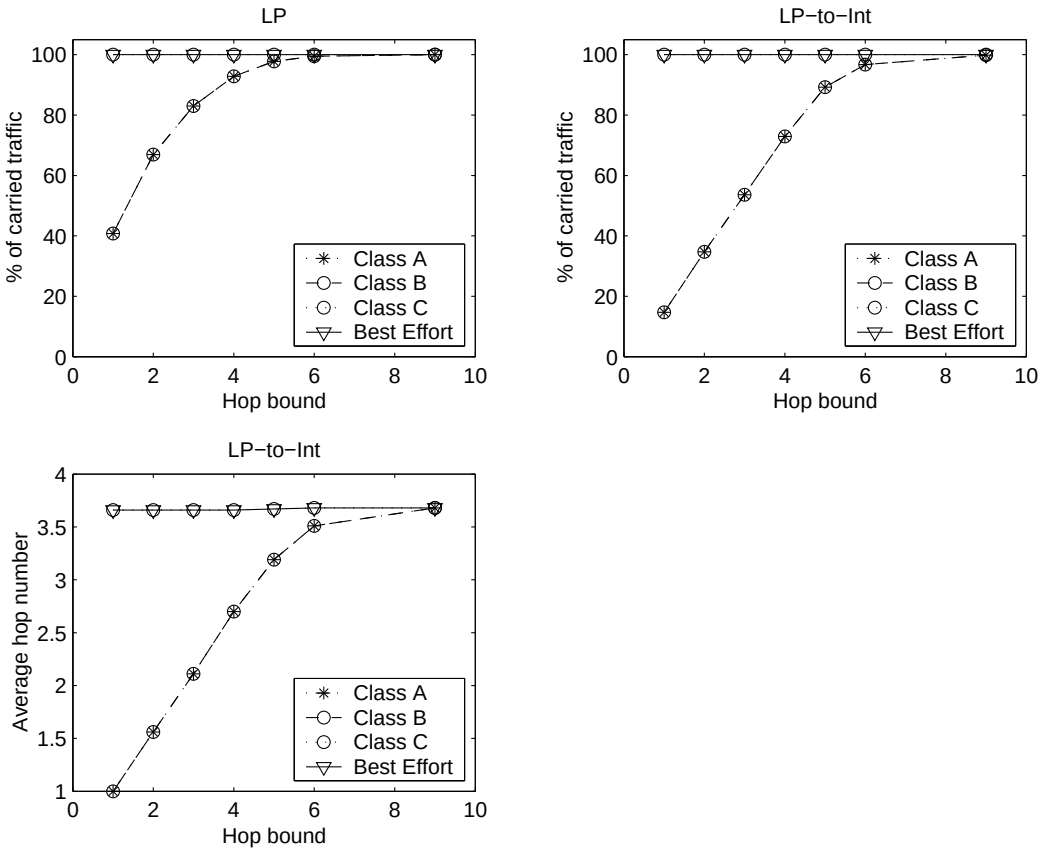


Figure 2.28: Effect of $H_{max}(s)$ in network 3 with light load

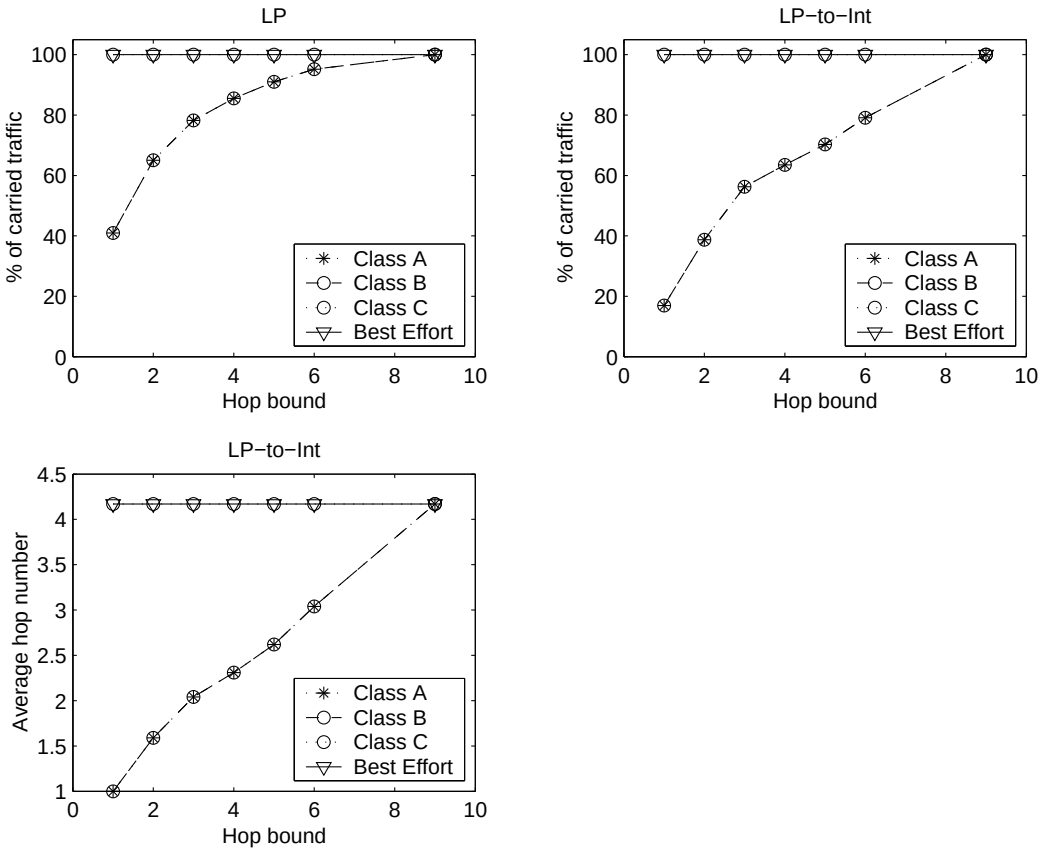


Figure 2.29: Effect of $H_{max}(s)$ in network 4 with light load

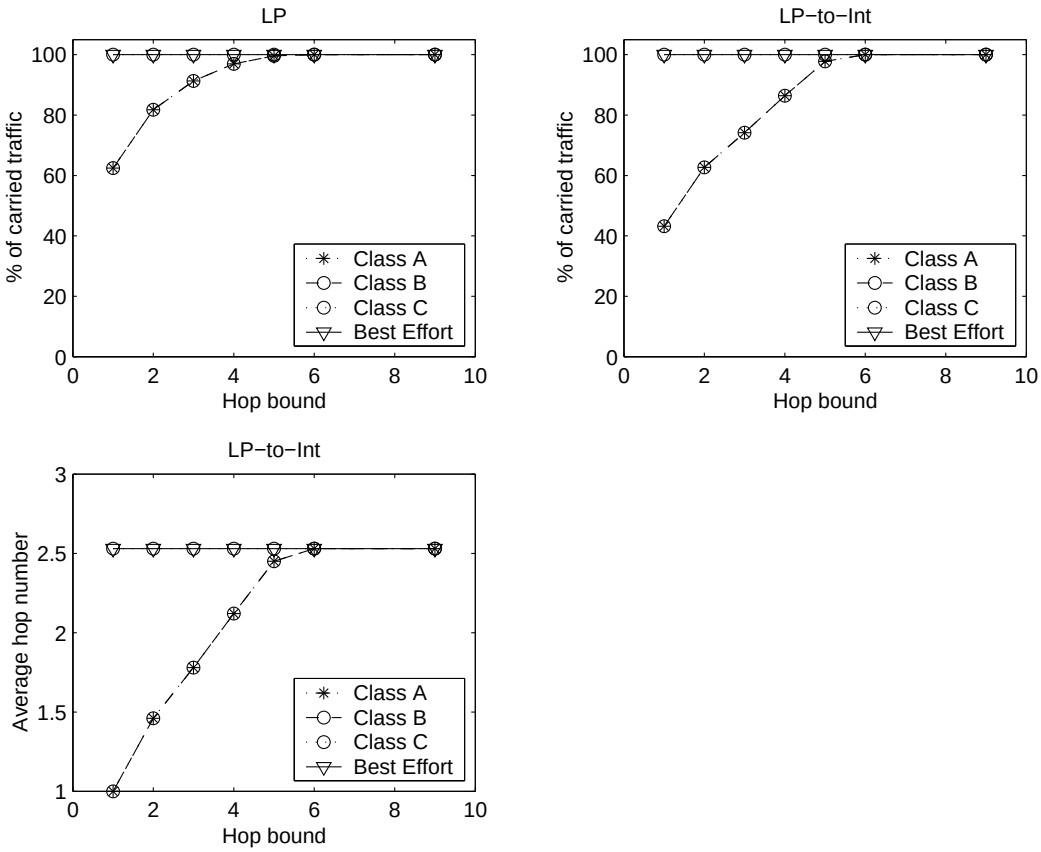


Figure 2.30: Effect of $H_{max}(s)$ in network 5 with light load

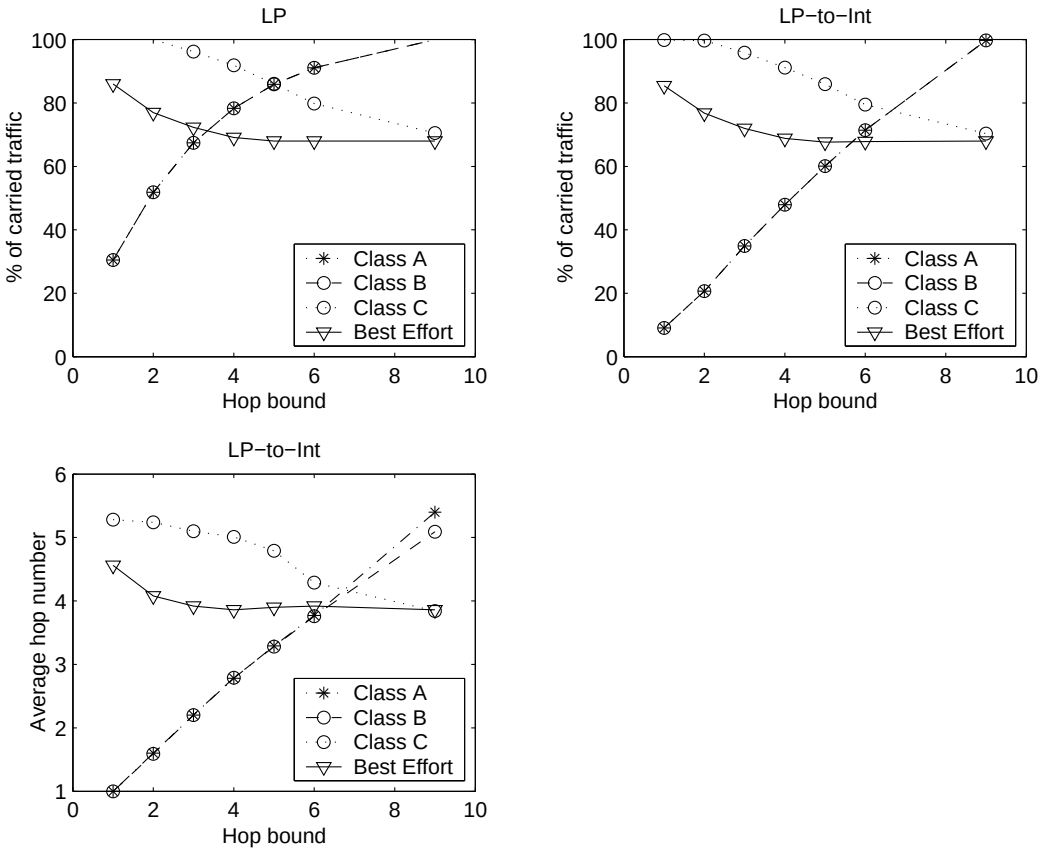


Figure 2.31: Effect of $H_{max}(s)$ in network 2 with heavy load

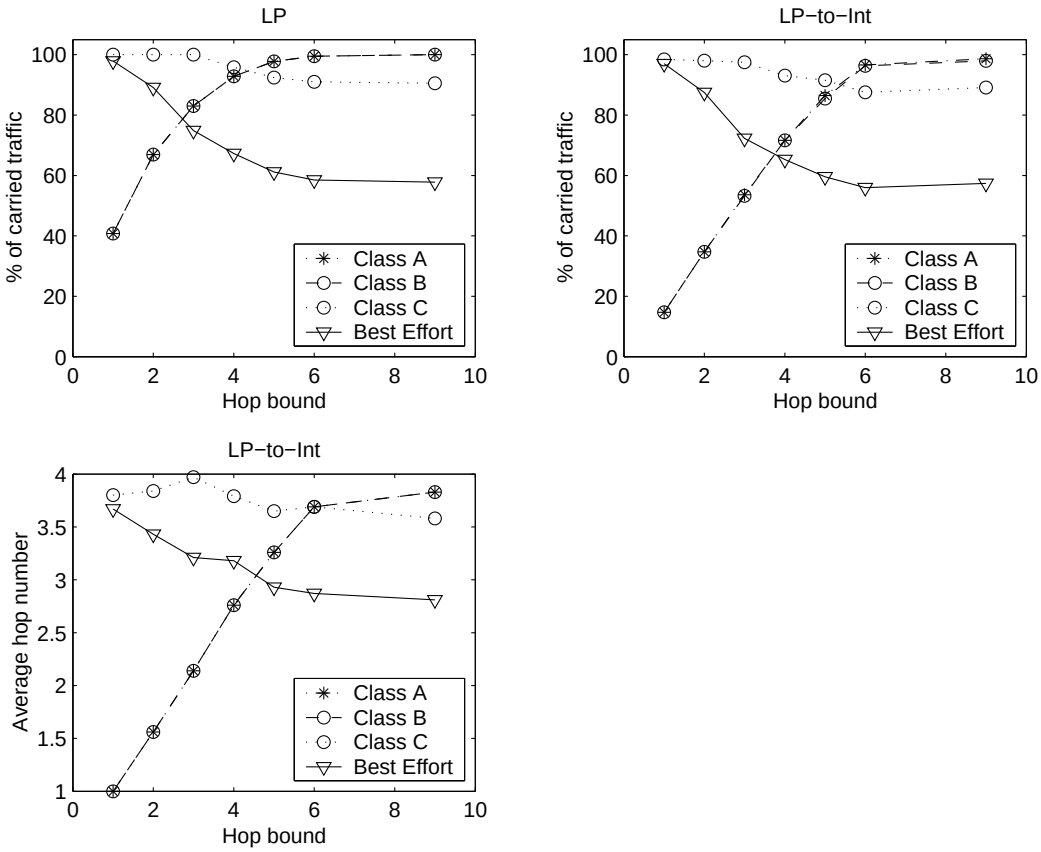


Figure 2.32: Effect of $H_{max}(s)$ in network 3 with heavy load

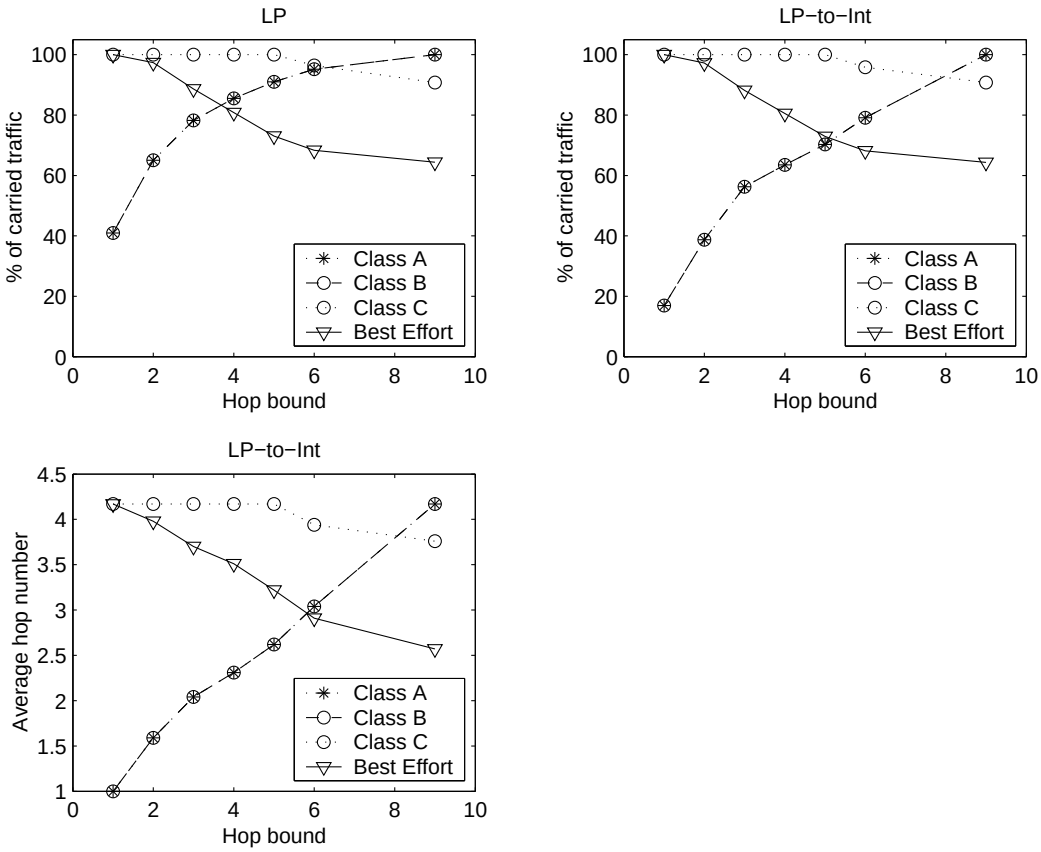


Figure 2.33: Effect of $H_{max}(s)$ in network 4 with heavy load

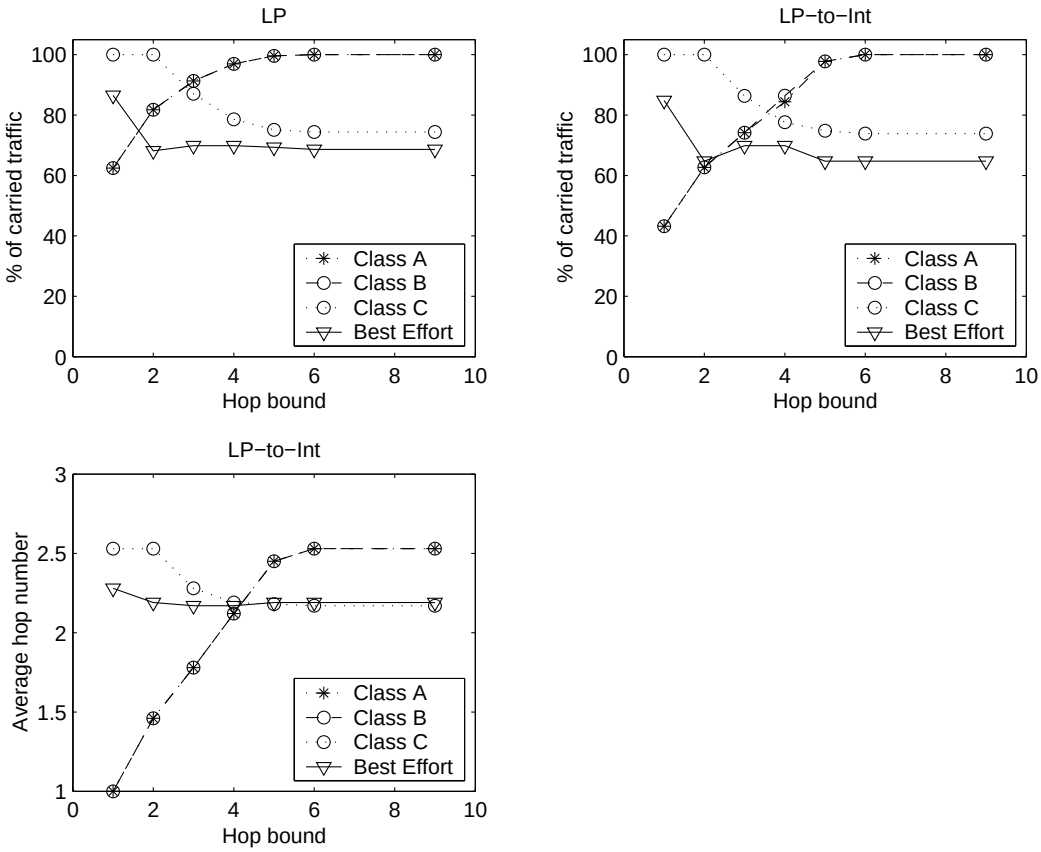


Figure 2.34: Effect of $H_{max}(s)$ in network 5 with heavy load

Chapter 3

Path Protection

3.1 Introduction

In this chapter, we present our path protection algorithms restorable against multiple nodes and links failure in the network to carry the QoS traffic. Following the classification in Section 1.4, our path protection approach specifically for one-plus-one $(1 + 1)$ and one-to-one $(1 : 1)$ path protection is presented in Section 3.2. The numerical results of the performance are shown in Section 3.3.

3.2 Our Approach

In this section, we present our approach to support both $1 : 1$ path protection and $1 + 1$ path protection based on our approach to carry the QoS traffic in Section 2.3.5. A backup path can be set up to be link-disjoint or node-disjoint from the working path. In Section 3.2.1, the scheme of setting up the backup path is shown with the link-disjoint and node-disjoint constraints. In Section 3.2.2, we discuss a method to choose the working path so as to have less hops than the backup path.

3.2.1 Establish $1 : 1$ and $1 + 1$ Backup Path

Path protection is only desirable for QoS traffic. We establish the backup paths based on *P8* for different classes of traffic and different OD pairs depending on the request of the service. The $1 + 1$ path protection requires two link-disjoint or node-disjoint LSP's from source to destination. We proposed the link-disjoint and node-disjoint constraint to select the backup path properly. The $1 : 1$ path protection mechanism require that the resource on the backup paths are available to preemptible low-priority traffic. This can be done by routing the QoS traffic and the low priority BE traffic at different stage. In our design, the QoS traffic and the corresponding backup paths are routed first and can use all the link capacities . The BE traffic is routed next, and can use the capacity utilized by the backup path and the residual capacity left by the QoS traffic.

Following the same technique used in Section 2.3.3, a backup path can be easily created as a new OD pair. Thus we have an additional OD pair from the same source to the same destination. Let σ_w be an OD pair of the QoS traffic and suppose an OD pair σ_b is created for the backup path. The backup path must have the same traffic demand as working path, so we set $D_{\sigma_w} = D_{\sigma_b}$. We also let $F_{\sigma_b} = F_{\sigma_w}$, so that both the working path and the backup path will be allocated the same amount of bandwidth. As a result, the traffic demand will be accepted or rejected simultaneously.

The backup path may be link or node disjoint with the working path. We support the selection of the backup path for both criteria. In *P8*, a non-bifurcation problem is set up to carry the QoS traffic, and this results in a single path for each traffic demand. The link-disjoint constraint can easily imposed in the non-bifurcation problem as follows.

$$Y_{\sigma_w,l} + Y_{\sigma_b,l} \leq D_{\sigma_w}, \forall l. \tag{3.1}$$

For the MIP solution, since only one path will be established for each traffic demand, the bandwidth allocated on each link $Y_{\sigma,l}$ will be the same as the traffic demand D_{σ_w} or 0. In the case of LP-to-Integer solution, although the above is not true for the LP solution, if will hold when integer solution is generated, since in this case if the allocated bandwidth is less than the demand it is rejected. In the constraint (3.1), it ensures that the link l is only utilized by either the working path or the backup path.

The node disjoint constraint for the working path and backup path is as follows.

$$\sum_{l \in L_i(n)} Y_{\sigma_w,l} + \sum_{l \in L_i(n)} Y_{\sigma_b,l} \leq D_{\sigma_w}, \forall n \notin T, \tag{3.2}$$

or

$$\sum_{l \in L_o(n)} Y_{\sigma_w,l} + \sum_{l \in L_o(n)} Y_{\sigma_b,l} \leq D_{\sigma_w}, \forall n \notin S. \tag{3.3}$$

In the constraint (3.2), it ensures that either the OD pair σ_w or the OD pair σ_b has the traffic entering node n. Similarity in (3.3), either the OD pair σ_w or the OD pair σ_b has the traffic leaving node n. It should be emphasized that, these constraints only work in the case of the non-bifurcation, or when granularity $g = 1$.

3.2.2 Length of the Backup Path

In Section 3.2.1, we establish the backup paths by creating new OD pairs. By using the link-disjoint constraint and the node-disjoint constraint, we have a choice in selecting the level of reliability of the backup path. However, we don't distinguish the working path and backup path at the time of design. As a result, the working path may be longer than the backup path. This is undesirable in the case of $1:1$ path protection because the traffic is normally travelling on the working path, and is switched to the backup path only when the working path fails. The longer working path will use more network resources and cause more delay for the traffic while there is a shorter alternative path.

With the hop constraint in Section 2.3.5, the working path and the backup path will have a limited hop count, but the choice of the working path might still longer than the backup path. Thus we need to differentiate the length between both of them. We can choose the shorter path as the working path in the process of flow decomposition. Another approach is to resolve the problem in the MCF problem. Since the longer path use more network resources, to distinguish the utilized resources between the working path and the backup path, we introduce the parameter B_σ as follows.

$$maximize \quad \sum_{\sigma} e_{s,\sigma} F_\sigma^s - \varepsilon \sum_{\sigma} B_\sigma \sum_{l} Y_{\sigma,l}. \tag{3.4}$$

We define B_σ to be the parameter used to differentiate the working path and backup path. For those working path OD pairs, the B_σ is relatively larger than the B_σ of the backup path OD pairs. Thus, in (3.4), if a working path utilize more resources than backup path, we will obtain less revenue. Since our objective is to maximize the revenue, this situation will not happen, and we can obtain a shorter working path.

3.3 Numerical Result

The experiments we have conducted verify the efficiency our path protection scheme for different cases including the sharing of the backup resources, the link-disjoint constraint, the node-disjoint constraint, and the use of B_σ. In Section 3.3.1, the networks and the experimental configuration is described. In Section 3.3.2, the result of the experiments are illustrated and discussed.

3.3.1 Simulation Configuration

The environment is similar to that of QoS traffic in Section 2.4.3. We have three QoS classes, Class A, Class B, Class C and one BE class. The demand set for each network is divided evenly among the four service classes. For Class A, Class B and Class C each OD pair requires a backup path. Note that the bandwidth is assigned for the traffic demand only when both the working path and backup path can be accommodated. The difference from the QoS experiment is that we routed the QoS traffic and BE traffic at different stage. Consequently, the resources of the backup path can be assigned to the BE traffic.

The experiment compares the performance of $1+1$ and $1:1$ path protection scheme. In $1+1$ path protection, the backup resources of the QoS traffic are not shared by the BE traffic, the resources of the backup paths are fully occupied by another copy of the carried traffic, but only one copy is selected in PML. In the $1:1$ path protection, the resources of the backup path are shared, the resource of backup paths are counted as the residual capacities available for the BE traffic.

The choice of B_σ has to be cautious in order not to effect the use of ε. B_σ is intended to reduce the importance of resources reserved for backup path comparing to the resources allocated for the working path. B_σ should be smaller in backup path than in working path. By the conclusion from the experiment of ε in Section 2.4.2, we know that the choice of ε is not critical. We set $B_\sigma = 1$ for the working path,and $B_\sigma = 0.1$ for the backup paths.

3.3.2 Experiment Result

The results of the experiments done for network 1 are illustrated in two parts. In the first part, the result of $1+1$ path protection is presented. In the second part, we illustrated the performance of $1:1$ path protection. In each path protection mechanism, different backup path selection schemes are used to compare the results. And we also show the effect of the average number of hops when B_σ. is used.

3.3.2.1 $1+1$ Path Protection

In Figures 3.1 - 3.4, we run the $1+1$ path protection algorithm for network 1. We plot the percentage of carried traffic in both LP and LP-to-Int cases, the average hop number and the backup average hop number all vs the percentage of the demand. In Figure 3.1, the experiment shows the performance of the link-disjoint backup path selection scheme with the differentiation of the working path and the backup path. As the result from the QoS traffic in Section 2.4.3, the traffic starts to drop as the demand increases. The drop sequence follows their priority of classes, from BE, Class C, Class B. The difference from the QoS traffic experiment using two stage in Section 2.4.3 is that all classes of traffic drops early because we account the demand of backup. We also double the demand of the BE for comparison. The rate of carried traffic for BE is getting closer to that of Class C as the demand increases. This is due to the constraint that the bandwidth is assigned for the traffic demand only when both the working path and backup path can be accommodated. Since BE traffic does not have the constraint, the rate of carried traffic will perform better than the other classes as the demands continue to increase. Comparing the LP solution to the LP-to-Int solution, we see that as the demand increases, the round off effect of the LP-to-Int solution also increases. We can use the smaller granularity g for those demands who don't need the backup path to get closer to the LP solution. Another observation is that as the demands increase, the average hop number will also drop. This holds because the large hop number consumes more resources, and is undesirable while the network resources are deficit. In Figure 3.2, we examine the link-disjoint backup path selection scheme without differentiation between the working path and backup path. It's evident that the average backup hop number has smaller hop number than the average hop number of working path. The hop number of BE traffic is even shorter than the QoS traffic. Comparing to 3.1, we can see that the B_σ can distinguish the working path with the backup path properly. The performance of the LP solution in 3.2 is the same as that in 3.1.

In Figure 3.3, we examine the node-disjoint backup path selection scheme with the use of B_σ. Each QoS traffic drops earlier comparing to 3.1. This is because with the node-disjoint constraint, the backup path is more difficult to be found. Other difference is that the rates of the carried traffic for the BE and Class C are closer comparing to the link-disjoint scheme. This is because the rate of Class C traffic drops and the BE traffic raises. In Figure 3.4, we can also see that the average backup path hop number is shorter than the working path. This also shows the effect of B_σ.

3.3.2.2 $1:1$ Path Protection

In Figures 3.5 - 3.8, we run the $1:1$ path protection algorithm for network 1. We plot the percentage of carried traffic in both LP and LP-to-Int cases, the average hop number and the backup average hop number all vs the percentage of the demand. In Figure 3.5, the experiment shows the performance of the link-disjoint backup path selection scheme with the use of B_σ. The traffic starts to drop as the demand increases. The drop sequence also follows their priority of classes Class C, Class B. The difference from the $1+1$ path protection is that the BE traffic has high percentage of carried traffic. This is because the resources of the backup path left by the QoS traffic can be utilized by the BE traffic. Though the BE traffic can be preemptible, the resource utilization dramatically increased. Comparing the LP solution to the LP-to-Int solution, we still see that as the demand increases, the round off effect of the LP-to-Int solution also increases. In Figure 3.6, we use the link-disjoint scheme without B_σ. It's also clear that the average backup hop number has smaller hop number than the average hop number of working path. Since the BE traffic is not effect by the backup path, it has the same hop number as in Figure 3.5. The performance of the LP solution in 3.6 is the same as that in 3.5.

In Figure 3.7, we examine the node-disjoint scheme with the use of B_σ. Each QoS traffic also drops earlier comparing to 3.5. In Figure 3.8, we can also see that the average backup path hop number is shorter than the working path. This also shows the effect of B_σ.

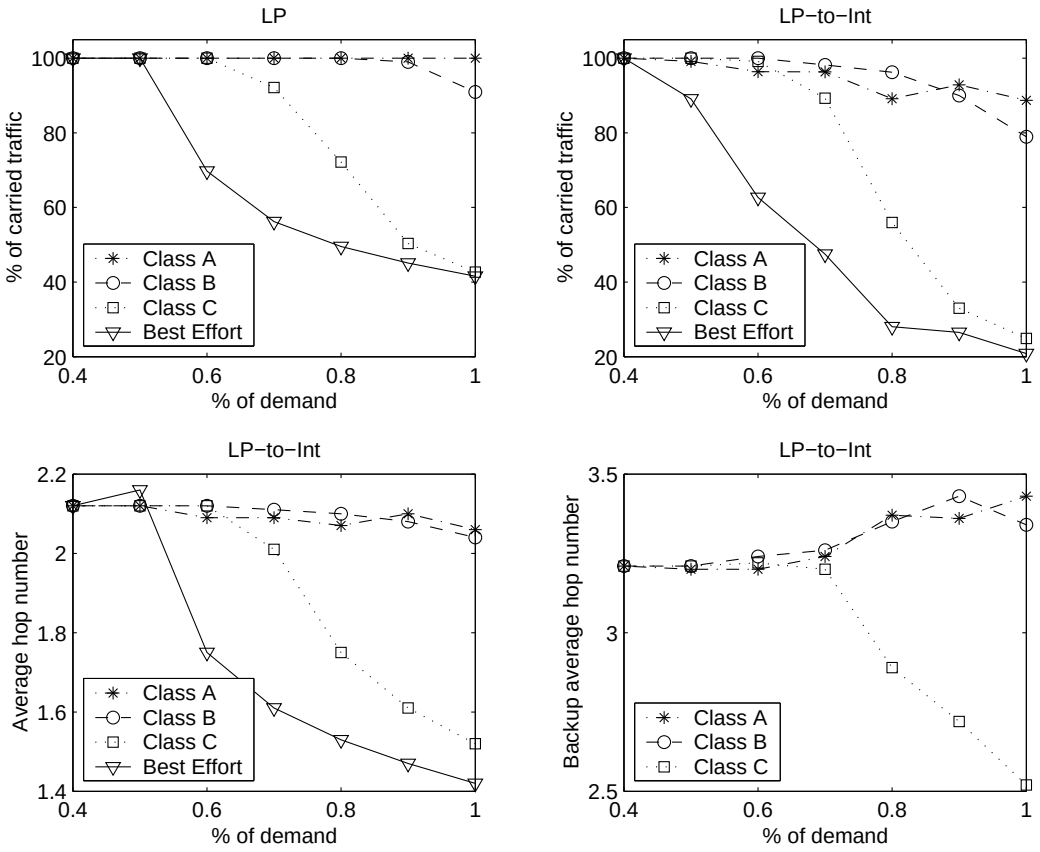


Figure 3.1: $1+1$: link-disjoint backup path, with B_σ

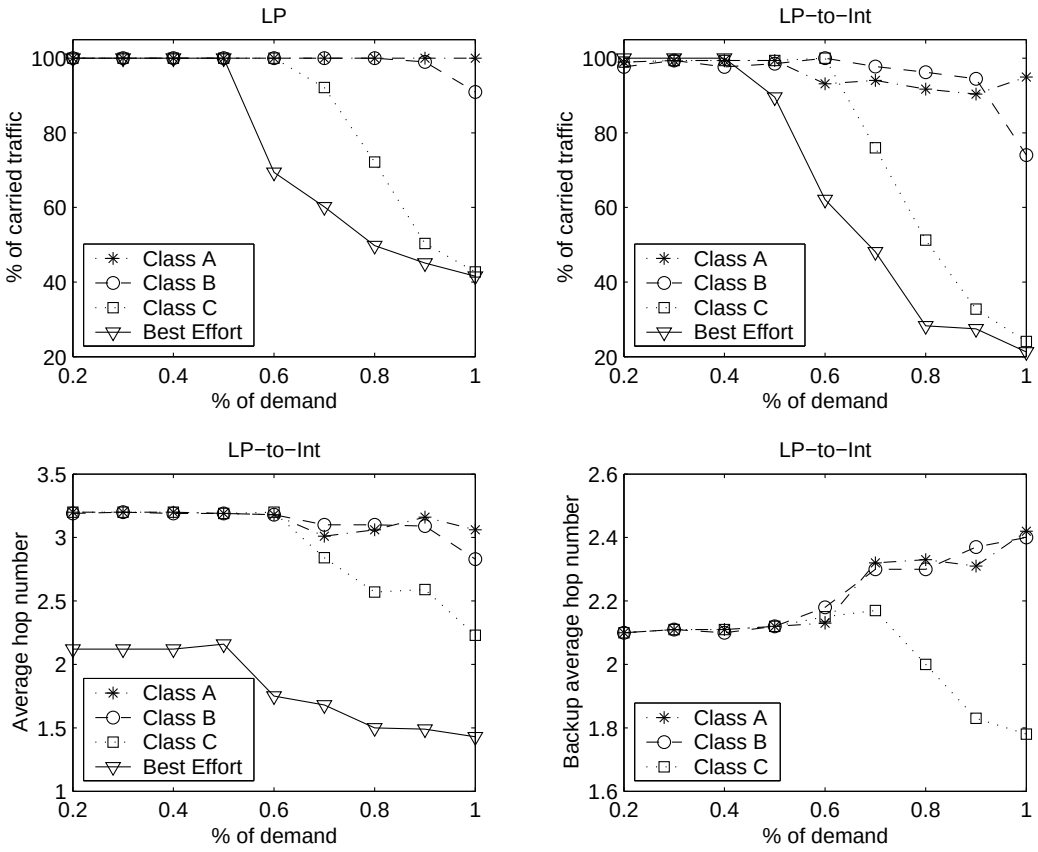


Figure 3.2: $1 + 1$: link-disjoint backup path, without B_σ

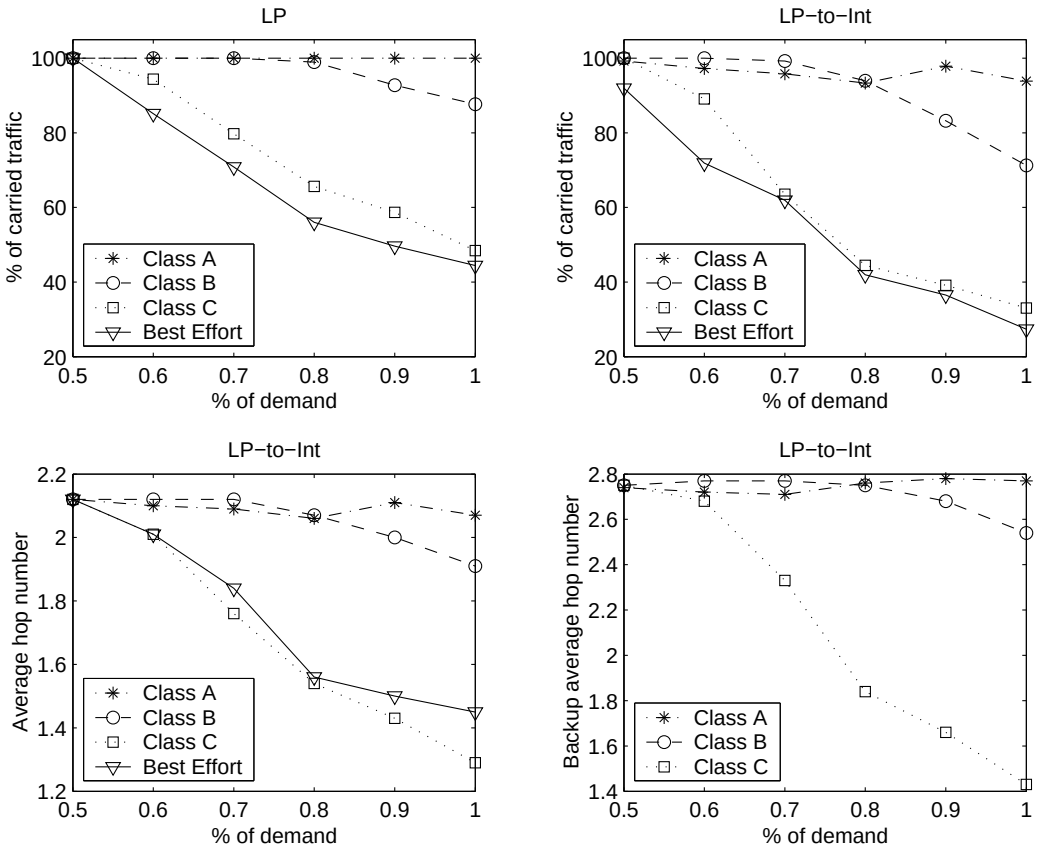


Figure 3.3: $1 + 1$: node-disjoint backup path, with B_σ

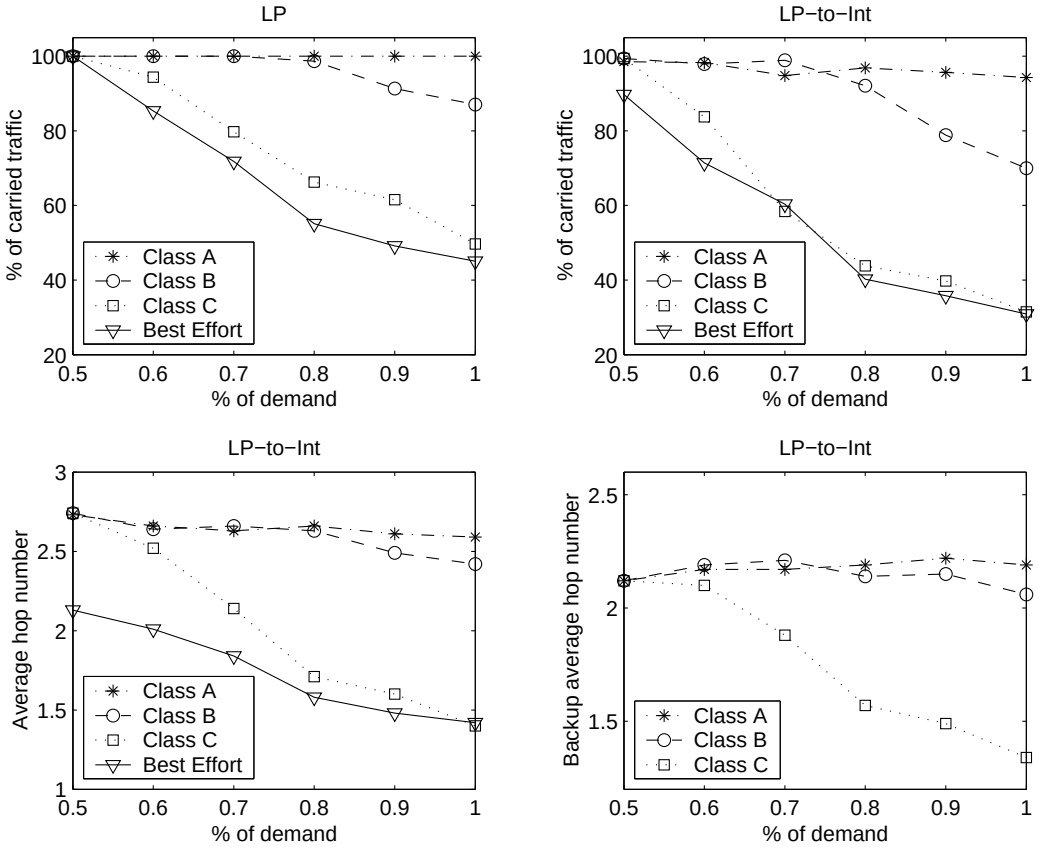


Figure 3.4: $1+1$: node-disjoint backup path, without B_σ

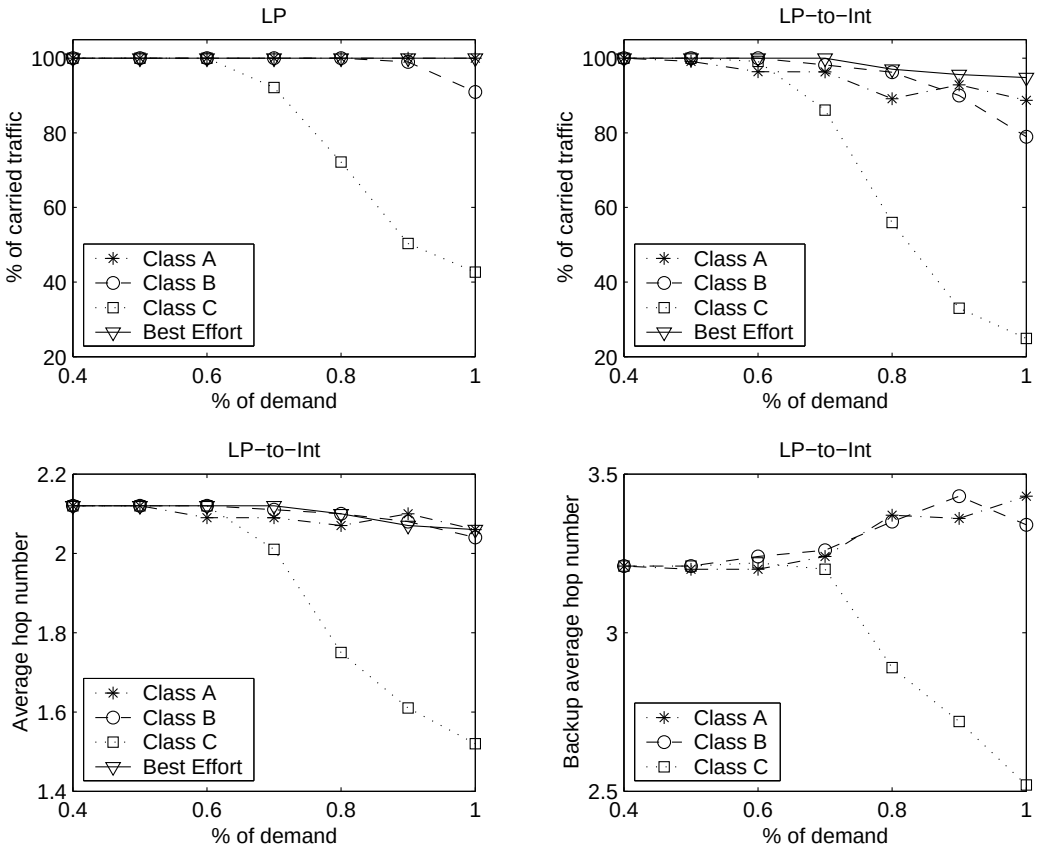


Figure 3.5: $1:1$: link-disjoint backup path, with B_σ

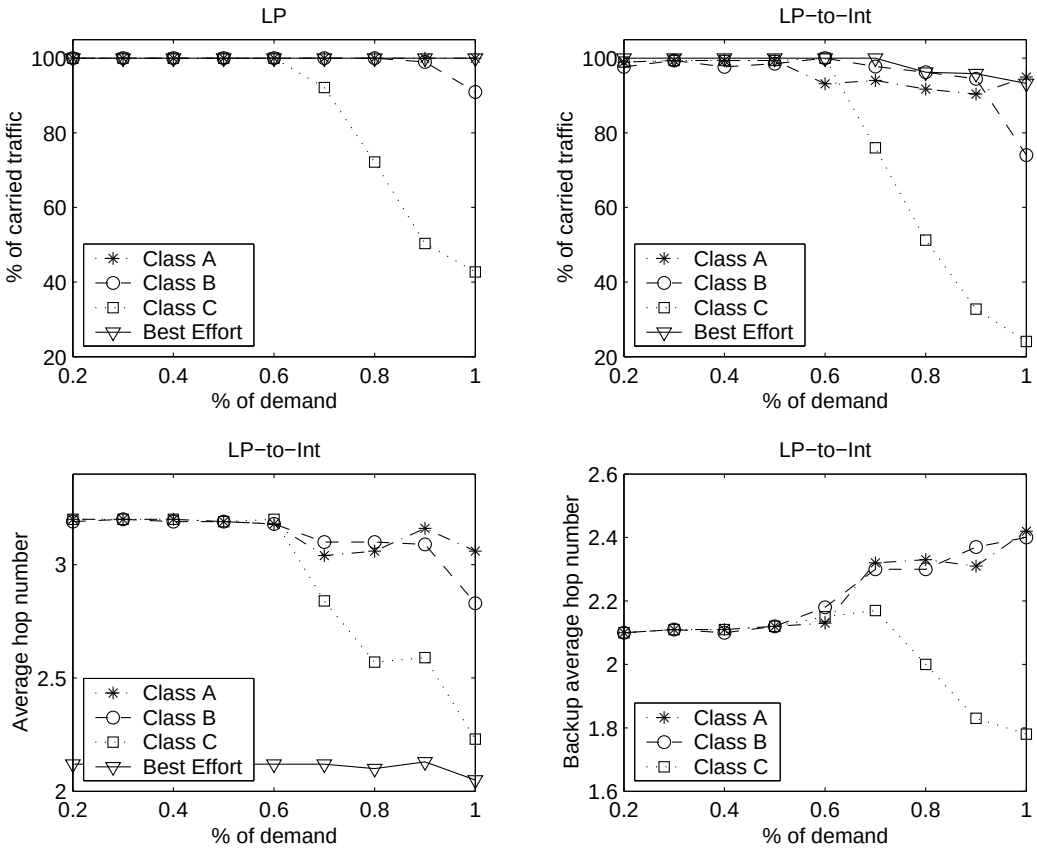


Figure 3.6: $1:1$: link-disjoint backup path, without B_σ

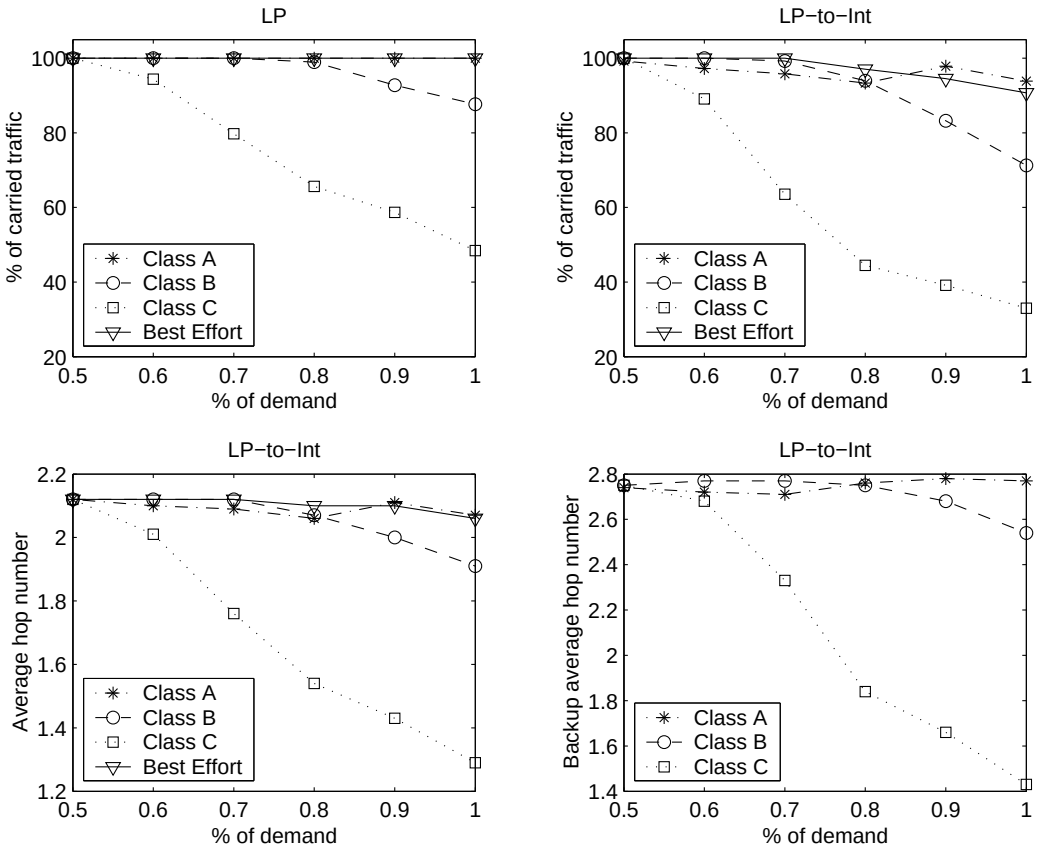


Figure 3.7: $1:1$: node-disjoint backup path, with B_σ

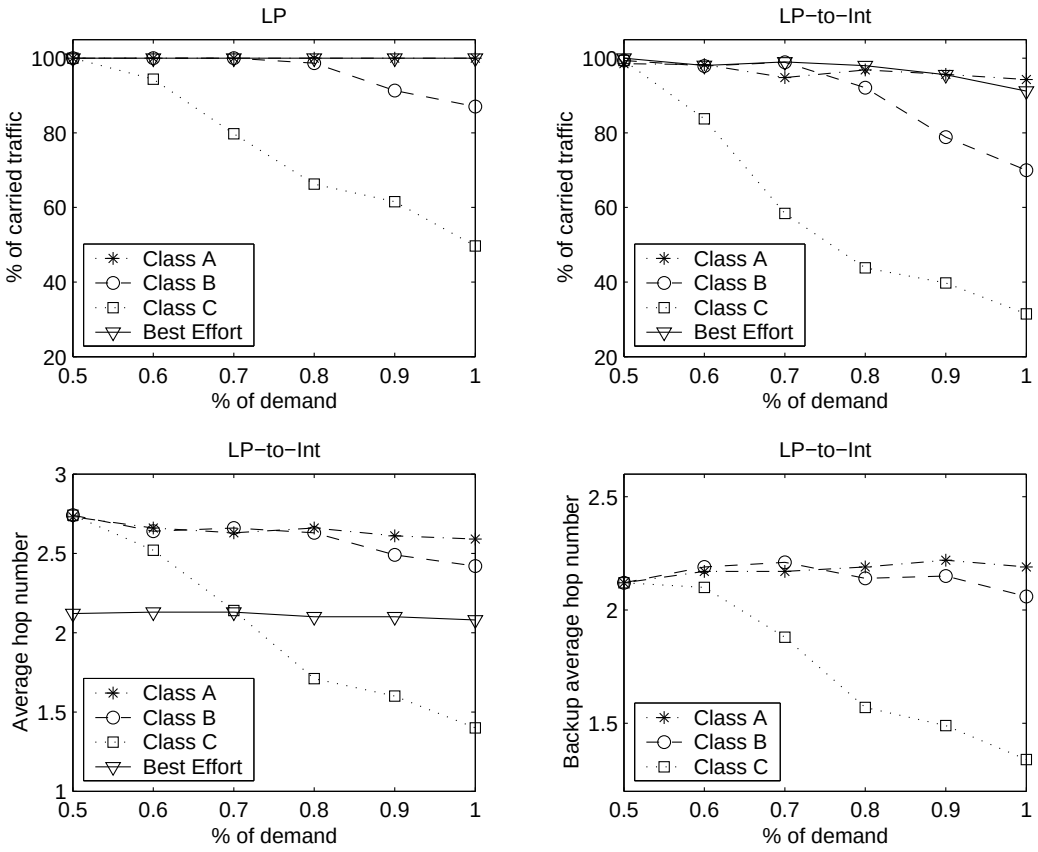


Figure 3.8: $1:1$: node-disjoint backup path, without B_σ

Chapter 4

Conclusion

In this thesis, we review several approaches to the constraint-based routing problems from the literature which intend to utilize the explicit routing of MPLS to achieve the goals of traffic engineering. The route-based MCF problem for the QoS traffic in [4] chooses the final routes from the pre-selected route sets. This limits the network performance while the route set is small, and it's difficult to choose the route set prudently when the route set is large (especially in the large densely connected networks). In addition, the link-based MCF problem implemented for the BE traffic in [4] may result in loops and can lead to a large label space in a node. This in turn limits the scalability of this approach if implemented in large scale networks. In [3], the authors proposed algorithms using to-trees to reduce the number of labels used. However, their result is an overkill that creates additional burdens and constraints in the network due to the deployment of to-trees. In [5], the authors proposed an approach for node affinity which can control the LSP's to include or exclude specific groups of links or nodes. As a result, we can specify the desired path partially or completely into an included node list just as the loosely explicit route and the strictly explicit route. However, their approach can't achieve the intend goal of included nodes, and may also create loops in the solution.

We first introduce the integer variables and consider the split ratio as in [5] for the link-based MCF problem and propose the label constraints to limit the number of labels. The constrained optimization problem becomes mixed integer programming problem. The computational complexity is increased because of the MIP problem, so we propose an LP-to-Integer approach which finds a solution close to the optimal solution but is computationally less complex. An efficient way to eliminate the loops is implemented by considering the utilized resources in the objective function. In this approach, we eliminate the loops and reduce the network resources successfully. Also, an algorithm to implement the node affinity is proposed by replacing the original demand into several segments of demands between the desired included nodes. As in [14], we limit the number of hops for some classes of traffic which may have stringent hop constraints due to the delay requirements. The $1 + 1$ and $1 : 1$ path protection is also implemented in our link-based MCF problem, and can be easily extended to $n : m$ protection to allow the sharing of the backup resources. We also consider both the link-disjoint and node-disjoint backup path selection algorithms which increase the flexibility of selection of the backup paths.

Bibliography

[1] Xipeng Xiao, Alan Hannan, Brook Bailey and Lionel Ni, "Traffic engineering with MPLS in the Internet", *IEEE Network Magazien*, pp.28-33,March 2000.

[2] D.Awduche, J.Malcolm, J.Agogbus, M.O'Dell and J.McManus, "Requirements for Traffic Engineering over MPLS" , *RFC 2701* , Sept. 1999.

[3] David Applegate, Mikkel Thorup, "Load optimal MPLS routing with N + M labels" , *IEEE INFOCOM*, 2002.

[4] D.Mitra and K.G. Ramakrishnan, "A case study of multiervice multipriority traffic engineering design for data networks" , *IEEE GLOBECOM*, 1999, pp.1077-1083.

[5] Youngseok Lee, Yongho Seok, Yanghee Choi and Changhoon Kim, "A Constrained Multipath Traffic Engineering Scheme for MPLS Networks" , *IEEE* , 2002, pp.2431-2436.

[6] Teunis Ott, Tony Bogovic, Tami Carpenter, K.R. Krishnan, and David Shallcross, "Algorithm for Flow Allocation for Multi Protocol Label Switching" , *Telcordia Technologies, Inc* , Auguest. 2000.

[7] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture" , *RFC 3031* , 2001.

[8] Y. Wang, and Z. Wang, "Explicit Routing Algorithms for Internet Traffic Engineering" , *ICCCN* , 1999.

[9] E. W. Zegura, "GT-ITM: Georgia tech internetwork topology models (software)" , *http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm/gt-itm.tar.gz* , 1996.

[10] ILOG CPLEX, *http://www.ilog.com/products/cplex/*

[11] Optimized Multipath, *http://www.fictitious.org/omp*

[12] Changcheng Huang, Vishal Sharma, Ken Owens and Srnivas, "Building Reliable MPLS Networks Using a Path Protection Mechanism" , *IEEE Communications Magazine* , March 2002.

[13] Mort Naraghi-Pour and Manju V. Hedge, "Path Protection in MPLS Networks" , *Celox Networks, Inc* , November, 2001.

[14] M.K. Girish, Z. Bei, and J. Hu, "Formulation of the Traffic Engineering Problems in MPLS based IP Networks", *INFOCOM* , 2000.

[15] Manju Hegde, Mort Naraghi-Pour, "Engineering traffic in MPLS networks" , *http://www.eetimes.com/story/OEG20011121S0066*

[16] Chuck Semeria, "Traffic Engineering for the New Public Network" , *Juniper Networks, Inc* , September, 2000.

Vita

Chung-yu Wei received his Bachelor of Science degree in Information Engineering from I-Shou University, Taiwan, in 1997. He was awarded the Outstanding College Student Scholarships every year. After finishing his mandatorial military service as a Second Lieutenant in the Army, he worked as a system engineer for D-Link Corporation in Taiwan for two years. He enrolled in the Department of Electrical and Computer Engineering at Louisiana State University in January 2002, and will receive his Master of Science in Electrical Engineering degree in December 2003.