in "Performance Evaluations and Planning Methods for the Next Generation Internet", edited by A. Girard, B. Sanso, and F. Vazquez-Abad, Springer Verlag, 2005

Chapter 1

NETWORK TRAFFIC ENGINEERING WITH VARIED LEVELS OF PROTECTION IN THE NEXT GENERATION INTERNET

S. Srivastava
University of Missouri–Kansas City, USA
shekhar@conrel.sice.umkc.edu

S. R. Thirumalasetty
now with Ciena Corporation, USA
sthiruma@ciena.com

D. Medhi
University of Missouri–Kansas City, USA
Corresponding author
dmedhi@umkc.edu

Abstract

In this paper, we consider the network traffic engineering problem for provisioning tunnels in a backbone network where services with varied levels of protection are offered. Network protection to address for a failure continues to be a critical issue for the Next Generation Internet. Our modeling framework allows protection at various level to be considered in a unified manner through the notion of cycles that are made up of a disjoint pair of paths where the network may be capacitated by both bandwidth as well as tunnel constraint. We also consider a variety of network goals including the ability to provide as much bandwidth as possible for best-effort services along with guaranteed protection services and develop a composite objective function. We then present two heuristic for solving the models presented. Through our studies of different network topologies, we show the convergence as well as the effectiveness of our approach in considering multiple goals in a unified manner. For example, we have shown the tradeoff between accepting new requests of protection service classes and providing residual bandwidth for best-effort services. Finally, our results also show that capacity and tunnels can have equally important roles in ensuring effective traffic engineering of a network.

Keywords: Network protection, integer linear optimization formulation, algorithm, tunnels, MPLS.

1. Introduction

Next generation Internet (NGI) is expected to carry a wide variety of services with differing service requirements. Often, the work on this area falls primarily into the category of addressing different quality-of-service (QoS) requirements. While QoS is important, providing network protection to address for a failure is a paramount issue as well. Our vision for NGI is the ability to offer services with different levels of protection for different customers (or service classes) while still providing best-effort services. For example, consider the following service classifications: (i) guarantee the service quality only under normal network operating conditions, (ii) full guarantee of bandwidth/service quality under normal situations plus a reduced level of service in the event of a major link failure, and (iii) fully guaranteed bandwidth/service quality both under normal as well as under any major link failure situation. A customer may sign up for any of these service classes ahead of time. For brevity, we refer to them collectively as book-ahead guaranteed (BAG) services.

Our goal is to consider the network traffic engineering problem of provisioning demands and routes for BAG service requests, along with varied protection requirements for survivability. A way to provide provisioned routes for BAG is through the notion of tunnels. For example, multi-protocol label switching (MPLS) for the NGI is well-suited for this capability. MPLS technology [Davie and Rekhter, 2000] provides the ability to allocate bandwidth for different service classes through label-switched paths (LSP). Recently, MPLS has been enhanced with the fast reroute capability which allows the possibility to provide protection for a tunnel through provisioned back-up tunnel using LSPs [Aubin and Nasrallah, 2003]. There is however an important limitation imposed by MPLS. For example, each LSP setup requires a label on each intermediate node which is used for switching the input traffic to the destined output port. Thus, setting up each new LSP introduces additional labels to each intermediate node. To route each packet, a label switched router (LSR) would need to search through the Label Swapping table to find the matching label and the port to determine the output label and the port. It then appends the output label to the packet and sends the packet to the output port. Consequently, each activated LSP leads to more labels at LSRs, thereby requiring more processing to forward each packet. Thus, our interest here is to consider the network traffic engineering problem for varied levels of protection taking into account any restriction on number of tunnels imposed by label-switched routers while allowing enough bandwidth available for best-effort services at the same time.

Network protection and survivability have been addressed for a variety of communication networks over the years; for example, see [Fumagalli et al., 1999], [Kajiyama et al., 1994], [Kawamura et al., 1994], [Medhi, 1994], [Medhi and Khurana, 1995], [Ramamurthy and Mukherjee, 1999], [Xiong and Mason, 1999], [Wu, 1992]. Using MPLS for traffic engineering has received considerable attention in recent years [Awduche et. al., 1999], [Le Faucheur and Wai, 2003], [Aubin and Nasrallah, 2003], The work that is closest to ours is by Kodialam and Lakshman [Kodialam and Lakshman, 2000] where they have presented optimization models and algorithms for guaranteed tunnels with restoration. However, there are several differences. For example, we focus on protection rather than restoration, especially considering book-ahead guaranteed protection services with varied levels of protection.

The rest of the paper is organized as follows. In section 1.2, we provide an optimization formulation of the BAG traffic engineering problem with tunneling constraints for varied levels of protection. The basic problem can be considered for a variety of different objectives; in section 1.3, we show how different objectives can be a unified into a single objective. In section 1.4, we present two heuristic algorithms to solve the problem formulated. In section 1.5, we present numerical results to show effectiveness of our formulation.

2. Problem Formulation

For each origin-destination node pair in the network, we assume that we need to satisfy the required level of protection for a set of customers where each customer has a different protection grade-of-service requirement. Recall that we consider three protection grade-of-service classes; for simplicity, we refer to them as zero-, fractional-, full-protection BAG services. Here zero-protection means that the service is guaranteed under normal operating conditions but not under a failure; fractional-protection means providing a reduced level of services under a major link failure in addition to guaranteed service under normal operating

$\mathcal{N}$	Set of nodes in the network
$\mathcal{L}$	Set of links in the network
$\mathcal{K}$	Set of demand pairs with traffic demand in the network
k	Demand pair identifier
$i(k), j(k)$	originating node $i(k)$ and destination node $j(k)$ for demand pair identifier k
$\mathcal{S}_k$	Set of BAG service requests for demand $k \in \mathcal{K}$
d_k^s	Demand volume of service request s for demand pair k
$\mathcal{P}_k^s$	Set of candidate cycles for service request $s \in \mathcal{S}_k$ for demand $k \in \mathcal{K}$
C_ℓ	Capacity of link $\ell \in \mathcal{L}$
T_ℓ	Maximum number of tunnels allowed on link $\ell \in \mathcal{L}$
α_k^s	Protection level of service request $s \in \mathcal{S}_k$ of $k \in \mathcal{K}$
$c_{km}^{sp}(c_{km}^{sb})$	Cost of primary path p (back-up path b) associated with cycle m, request s, demand k
$\delta_{km}^{s\ell}$	1, if candidate cycle $m \in \mathcal{P}_k^s$ for service $s \in \mathcal{S}_k$ of demand pair $k \in \mathcal{K}$ uses link $\ell \in \mathcal{L}$ in its primary path; 0, Otherwise
$\beta_{km}^{s\ell}$	1, if candidate cycle $m \in \mathcal{P}_k^s$ for service $s \in \mathcal{S}_k$ of demand pair $k \in \mathcal{K}$ uses link $\ell \in \mathcal{L}$ in its backup/secondary path; 0, Otherwise
$U_{\{\alpha_k^s>0\}}$	1, if $\alpha_k^s > 0$; 0, otherwise
Variables:	
x_{km}^s	0/1 decision variable for choosing cycle m for s, k
w_k^s	0/1 artificial (slack) variable for s, k
Parameters:	
γ	Weighing factor for routing cost on the back-up path
η_k^s	Penalty cost of artificial path w_k^s
θ_k^s	Cost normalization factor for s, k
u_k^s	Utility of s, k
R	Utility weighing factor

Table 1.1. Summary of Notations

conditions; finally, full-protection means providing guarantee under both normal as well as failure conditions.

Consider a demand pair k between originating node $i(k)$ and destination node $j(k)$ (refer to Table 1.1 for a summary of all notations) where we have a set of service requests $\mathcal{S}_k$ from customers requiring BAG services at differing protection grade-of-service. We denote the bandwidth demand of customer $s \in \mathcal{S}_k$ for demand pair k by d_k^s.

First, consider zero-protection BAGS level demand request. In this case, only a path with bandwidth d_k^s needs to be provisioned ahead of time (or allocated ahead of time). Considering only the shortest (e.g., in terms of hops) path may not address the overall traffic engineering goal. Thus, we need to consider a set of candidate paths for each demand k.

For full-protection BAGS service class, a backup path needs to be available and bandwidth d_k^s needs to be reserved on the backup path. We require that the backup path survive if the primary path is affected due to any critical failure situation of interest to a provider, e.g., for a single link failure at a time. Although the primary and backup path could be independently modeled, we use a pairing idea, i.e., consider a pair of paths consisting of primary and backup paths. Similar to the case of zero-protection services, the selection of the shortest pair of primary and backup paths for a demand d_k^s may not be in the best interest of a traffic engineering objective. Thus, we consider a candidate set of primary/backup paths for a flow demand d_k^s, for full-protection.

Finally, in the case of fractional-protection BAGS services, the backup path needs to be allocated bandwidth sufficient to carry a fraction of d_k^s in order to address partial survivability. Thus, the fractional BAGS service class also requires a pair of disjoint paths. The difference is that, on the backup path, only a fraction of the demand is required to be reserved. If we denote the fraction by α_k^s (where $0 \leq \alpha_k^s \leq 1$), then the primary path would reserve d_k^s while the backup path would reserve $\alpha_k^s d_k^s$.

When we consider all the three cases, it is easy to see that by appropriately setting α_k^s we can consider each of the protection levels, i.e., $\alpha_k^s = 0$ refers to zero-protection, $\alpha_k^s = 1$ refers to full-protection, while α_k^s refers to fractional-protection if $0 < \alpha_k^s < 1$.

There are two benefits to the way we have introduced α_k^s: (i) fractional-protection need not be of a specific pre-defined value; each customer can request a different level, (ii) for zero-protection, we can also consider a pair of disjoint paths as well where on the backup path, we assign $\alpha_k^s = 0$; this means we can still consider a backup path but it is not used. Consequently, the three service classes can be considered in a unified manner from a modeling framework—all we need to do is to consider a set of candidate pairs of disjoint paths (see the work presented in [Suurballe, 1974], [Suurballe and Tarjan, 1986] on how to compute a pair of disjoint paths). For simplicity, we refer to a pair of disjoint paths as *cycles*. For example, a provider may be interested in providing protection service for a single link failure at a time. Thus, we need to consider a cycle that

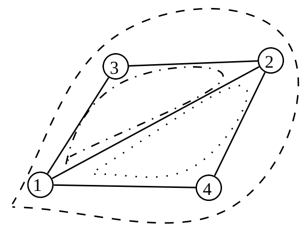


Figure 1.1. Illustration of Cycles

consist of a pair of link-disjoint paths; in fact, we need to consider a set of candidate cycles as input for the traffic engineering problem. As an illustration, consider Figure 1.1 where for the demand pair connecting nodes 1 and 2, we have three different candidate cycles 1-2-3-1, 1-2-4-1, and 1-3-2-4-1 which are link-disjoint; they all are considered as candidate cycles in our formulation while the traffic engineering objective decides which cycle to pick based on a set of requirements.

In general, we denote the set of candidate cycles for service s for demand pair k by $\mathcal{P}_k^s$. If we associate x_{km}^s as the decision variable with candidate cycle m, then for each $s \in \mathcal{S}_k$, $k \in \mathcal{K}$, the decision to select only one cycle is governed by the following constraint:

$$\sum_{m \in \mathcal{P}_k^s} x_{km}^s \ \leq \ 1.0, \quad s \in \mathcal{S}_k, k \in \mathcal{K}. \tag{1.1}$$

The inequality allows for the case when a demand request may not be satisfied. By introducing the artificial (slack) variable w_k^s, the above constraint can be re-written as

$$\sum_{m \in \mathcal{P}_k^s} x_{km}^s + w_k^s \ = \ 1.0, \quad s \in \mathcal{S}_k, k \in \mathcal{K}. \tag{1.2}$$

Note that for each cycle (because of the way each of them are generated) we have a 'primary' path and the backup path. Using link-cycle indicators (see Table 1.1), flow on link ℓ (denoted by $\hat{r}_\ell$) to carry demand volume under both normal and failure situations (for primary and backup path for different BAG demand request) can be captured by the amount

$$\hat{r}_\ell = \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} \sum_{m \in \mathcal{P}_k^s} [\delta_{km}^{s\ell} + \alpha_k^s \beta_{km}^{s\ell}] d_k^s x_{km}^s.$$

Note the inclusion of parameter α_k^s with the second term to dictate the level of protection on the backup path; also since we are considering a cycle consisting of disjoint paths, for a specific link ℓ, if $\delta_{km}^{s\ell}$ takes the value 1, then the corresponding $\beta_{km}^{s\ell}$ must be zero. Given capacity C_ℓ

for link ℓ, we thus have the following capacity constraint:

$$\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{S}_k}\sum_{m\in\mathcal{P}_k^s}[\delta_{km}^{s\ell}+\beta_{km}^{s\ell}\alpha_k^s]d_k^s x_{km}^s \le C_\ell, \quad \ell\in\mathcal{L}. \tag{1.3}$$

Finally, we have an additional constraint due to restriction on the number of tunnels outgoing from any label-switched router. If we restrict the number of active tunnels on link ℓ to T_ℓ, then we can write

$$\sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{S}_k}\sum_{m\in\mathcal{P}_k^s}[\delta_{km}^{s\ell}+U_{\{\alpha_k^s>0\}}\beta_{km}^{s\ell}]\, x_{km}^s \le T_\ell, \quad \ell\in\mathcal{L} \tag{1.4}$$

where $U_{\{\alpha_k^s>0\}}$ is to indicate that back-up paths are not to be counted as tunnels in the case of zero-protection request $\alpha_k^s=0$.

The network traffic engineering problem (**P**) is to minimize a suitable objective $f(\boldsymbol{x},\boldsymbol{w})$ (to be discussed in detail in the next section) in the presence of both capacity and tunneling constraints for book-ahead guaranteed services while incorporating varied levels of protection requirements; this can be written as

$$F^* = \min_{\{\boldsymbol{x},\boldsymbol{w}\}} f(\boldsymbol{x},\boldsymbol{w})$$

subject to

$$\text{constraints (1.2), (1.3), (1.4)}$$
$$x_{km}^s, w_k^s \in \{0,1\},\ m\in\mathcal{P}_k^s, s\in\mathcal{S}_k, k\in\mathcal{K}.$$

The above formulation does not include how to handle best-effort services; this will be discussed in the next section. Problem (**P**) is a multi-commodity flow model using link-path representation where "path" is replaced by cycles to allow for pairs of disjoint paths. The notion of using cycles in a link-path formulation setting was originally presented in [Medhi, 1991]. Another advantage of the above formulation is that we can incorporate situation-disjoint path to the primary path without changing the overall formulation since this only involves generation of cycles. The notion of situation-disjointness has recently been presented in [Krithikaivasan et al., 2003], [Pióro and Medhi, 2004]; in essence, construction of situation-disjoint paths generalizes the idea of link-disjoint or node-disjoint paths.

3. Objective Function

There are several possible goals that can be considered: i) provide bandwidth for best-effort services, ii) to minimize routing cost for different service requests, iii) to minimize the penalty for not meeting some service requests, iv) to maximize the revenue for service requests that are carried. In the following, we show how to combine these four goals in a single objective function.

We start with how to handle best-effort services. Besides allocation for BAG services, our goal is to provide maximum residual bandwidth to best-effort services so that this class can provide as good of a service as possible. We incorporate this requirement in the objective function by considering the maximization of residual capacity which can be written as

$$\max_{\{\boldsymbol{x},\boldsymbol{w}\}} \sum_{\ell\in\mathcal{L}} (C_\ell - r_\ell)$$

where r_ℓ is the bandwidth consumed by different BAG services on link ℓ. This is equivalent to:

$$\min_{\{\boldsymbol{x},\boldsymbol{w}\}} \sum_{\ell\in\mathcal{L}} r_\ell.$$

That is, to provide as good of a service as possible to best-effort services, we minimize the total bandwidth allocated to links in the network. Thus, our first objective is

$$f_1 = \sum_{\ell\in\mathcal{L}} r_\ell. \tag{1.5}$$

In a network with BAG services, there are however two ways to determine r_ℓ. In the first case, r_ℓ is computed assuming that the primary and backup paths, which were chosen for each s, k, are allocated bandwidth d_k^s and $\alpha_k^s d_k^s$, respectively, at the time of network provisioning; this is referred to as the *hard requirement*. In the second case, while the backup path is assigned for every primary path, the actual bandwidth is not necessarily reserved on the backup path during normal operating conditions; rather, signalling mechanism is used to actually allocate the bandwidth once the failure occurs. In other words, since the bandwidth which would be needed on the backup path is not reserved during normal operation, best-effort services can use this bandwidth under normal operating conditions; this is referred to as the *soft requirement*. In the following, we discuss the construction of the objective function for various goals, separately for the hard and soft requirements.

3.1 With Hard Requirement

For the hard requirement, (1.5) can be written as

$$f_1 = \sum_{\ell\in\mathcal{L}} \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} \sum_{m\in\mathcal{P}_k^s} (\delta_{km}^{s\ell} + \beta_{km}^{s\ell}\alpha_k^s) d_k^s x_{km}^s \tag{1.6}$$

for the first goal. Consider next the second goal: minimization of the routing cost of demand request. If c_{km}^{sp} is assumed to be the routing cost on the primary path, and c_{km}^{sb} is assumed to be the routing cost on the backup path of the candidate cycle $m \in \mathcal{P}_k^s$ of service request $s \in \mathcal{S}_k$ and demand pair $k \in \mathcal{K}$, then the objective of minimizing the total routing cost can be written as

$$f_2 = \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} \sum_{m\in\mathcal{P}_k^s} (c_{km}^{sp} + \gamma c_{km}^{sb}) x_{km}^s \tag{1.7}$$

where γ $(0 \le \gamma \le 1)$ is a weighing factor for the routing cost of the backup path. For example, this parameter allows less weight to be given on the backup path for routing. If we use (1.7) in Problem (**P**) as it is, then the optimal solution is not to route any bandwidth at all due to constraint (1.2); certainly, this is not desirable. Thus, for the third goal, if we introduce penalty cost η_k^s with artificial variable w_k^s to indicate the cost incurred for not routing demand, then we can re-write the objective function (1.7) as

$$f_{2'} = \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} \big(\sum_{m\in\mathcal{P}_k^s} (c_{km}^{sp} + \gamma c_{km}^{sb}) x_{km}^s + \eta_k^s w_k^s \big). \tag{1.8}$$

In order to combine the cost due to first goal, f_1, with the cost due to the second and the third goals, $f_{2'}$, we need to incorporate a new factor so that these two cost components can be normalized. That is, by introducing the normalizing factor $\theta_k^s (\ge 0)$ and only for routing variable part in (1.8), i.e., for (1.7) (and not double counting with penalty cost for artificial variables in (1.8), we can write the combined objective as

$$\begin{aligned} f_{1+\theta 2'} \quad &= \sum_{\ell\in\mathcal{L}} \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} \sum_{m\in\mathcal{P}_k^s} (\delta_{km}^{s\ell} + \beta_{km}^{s\ell} \alpha_k^s) d_k^s x_{km}^s \\ &+ \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} \big(\theta_k^s \sum_{m\in\mathcal{P}_k^s} (c_{km}^{sp} + \gamma c_{km}^{sb}) x_{km}^s + \eta_k^s w_k^s \big). \end{aligned} \tag{1.9}$$

In general, the unit routing cost on primary and backup, c_{km}^{sp} and c_{km}^{sb}, can be written in several ways. We show here three different ways. If it is based on hop count of a tunnel, then we can write

$$c_{km}^{sp} = \sum_{\ell\in\mathcal{L}} \delta_{km}^{s\ell} \quad \text{and} \quad c_{km}^{sb} = \sum_{\ell\in\mathcal{L}} \beta_{km}^{s\ell} \alpha_k^s.$$

If the cost of tunnel is based on flow forwarding cost associated with the tunnel, then we can use instead

$$c_{km}^{sp} = d_k^s \qquad \text{and} \qquad c_{km}^{sb} = \alpha_k^s d_k^s.$$

Finally, if the unit cost is based on both flow and hop-count, then we can write

$$c_{km}^{sp} = \sum_{\ell\in\mathcal{L}} \delta_{km}^{s\ell} d_k^s \quad \text{and} \quad c_{km}^{sb} = \sum_{\ell\in\mathcal{L}} \beta_{km}^{s\ell} \alpha_k^s d_k^s. \tag{1.10}$$

Suppose we incorporate the last one in the objective (1.9); then, by re-arrangement, we get

$$f_{1+\theta 2'} = \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} \Big(\sum_{m\in\mathcal{P}_k^s} [(1+\theta_k^s)c_{km}^{sp} + (\gamma+\theta_k^s)c_{km}^{sb}]x_{km}^s + \eta_k^s w_k^s \Big). \tag{1.11}$$

Finally, we consider the fourth goal: revenue maximization for accepting a demand. We address this factor in the form of a utilization parameter. The utility may vary from one request to the other, i.e., a request may be required to be allocated to a cycle even at a higher cost (that is, it consumes more bandwidth), if the utility of that request is higher. Let u_k^s $(0 < u_k^s < 1)$ be the (normalized) utility of service class $s \in \mathcal{S}_k$ of request $k \in \mathcal{K}$. By incorporating utility, the objective function finally becomes

$$f_h = \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} \Big(\sum_{m\in\mathcal{P}_k^s} [(1+\theta_k^s)c_{km}^{sp} + (\gamma+\theta_k^s)c_{km}^{sb} - Ru_k^s]x_{km}^s + \eta_k^s w_k^s \Big). \tag{1.12}$$

where R is the utility weighing factor. The value of the utility weighing factor R dictates the importance of utilities of requests over the costs. The special case of Problem $(\mathbf{P})$ of minimizing function f_h will be referred to as $(\mathbf{P}_h)$. To summarize, in the above function, we have incorporated four different goals—this is done through a combination of parameters $\theta_k^s, \gamma, R, u_k^s, \eta_k^s$. However, we need to distinguish that u_k^s is an input choice due to customer requirement for a network provider much like α_k^s. Thus, we are left with the key parameters: θ_k^s, γ, R, and η_k^s.

3.2 With Soft Requirement

Soft requirement pertains to the scenario where the backup paths are indicated at the time of provisioning, but not reserved with bandwidth, i.e., the capacity needed on the backup path is "allowed" for use by best-effort services *as long as* there is no failure; certainly, backup paths are immediately assigned the required protection bandwidth depending on the BAG service class as soon as a failure occurs through a signalling message, thus, bumping out best-effort services. Note that such a benefit may come at the expense of increase in time required to restore from a failure. Regardless, load on link due to bandwidth only on the primary paths can be captured as $r_\ell = \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} \sum_{m\in\mathcal{P}_k^s} \delta_{km}^{s\ell} d_k^s x_{km}^s$ for the first goal, leading to the network load

$$f_1 = \sum_{\ell\in\mathcal{L}} \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} \sum_{m\in\mathcal{P}_k^s} \delta_{km}^{s\ell} d_k^s x_{km}^s. \tag{1.13}$$

With soft requirement, the routing cost is only due to primary paths; thus, incorporating penalty cost, we have

$$f_2 = \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} \big(\sum_{m\in\mathcal{P}_k^s} c_{km}^{sp} x_{km}^s + \eta_k^s w_k^s \big). \tag{1.14}$$

for the second and the third goals. Similar to the hard requirement scenario, we can construct the combined allocated capacity and routing cost function $f_{1+\theta 2}$ as

$$\sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} \big(\sum_{m\in\mathcal{P}_k^s} [x_{km}^s c_{km}^{sp} + \theta_k^s \sum_{\ell\in\mathcal{L}} \delta_{km}^{s\ell} d_k^s x_{km}^s] + \eta_k^s w_k^s \big).$$

Incorporating the routing cost in the same fashion as in (1.10), and rearranging the terms, we get

$$f_{1+\theta 2} = \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} \big(\sum_{m\in\mathcal{P}_k^s} x_{km}^s (1+\theta_k^s) c_{km}^{sp} + \eta_k^s w_k^s \big).$$

Incorporating the utility for revenue for carried demands (the fourth goal), we have the final objective function as

$$f_s = \sum_{k\in\mathcal{K}} \sum_{s\in\mathcal{S}_k} [\sum_{m\in\mathcal{P}_k^s} ((1+\theta_k^s) c_{km}^{sp} - u_k^s R) x_{km}^s + \eta_k^s w_k^s]. \tag{1.15}$$

Note that (1.15) looks the same as (1.12); however, there is a subtle difference due to hard and soft requirement which is captured through the cost components. Problem (**P**) with the above objective function will be referred to as problem ($\mathbf{P}_s$).

From either the hard or the soft requirement, we can see that four goals are combined in a single objective function. In the process, we find that there are four key parameters (drivers) that impacts the final allocation of tunnels and bandwidths: R, θ_k^s, η_k^s in case of the hard and the soft requirements, and γ in the case of the hard requirement. In Section 1.5, we will discuss the implication of these parameters on the effectiveness of our formulation through appropriate network measures.

4. Solution Approach

The network traffic engineering problem presented in this work is for book-ahead guaranteed services; thus, the tunnel selection can be determined off-line ahead of time so that tunnels can be provisioned for different request and level of protection. First note that Problem (**P**) is an integer linear programming (ILP) problem. We describe here two solution approaches. Our first solution approach, $gIP()$, is based on successive approximation by continuous relaxation of Problem (**P**), while our second approach, $gSA()$, is based on the simulated allocation method [Pióro and Gajowniczek, 1997].

4.1 Heuristic I: *gIP*

The first approach is based on successive approximations by continuous relaxations of the integer linear programming problem **P** which is shown in Algorithm 1. We define two sets X_f and X_v containing variables $\boldsymbol{x}$ which have been fixed (their values are already decided) and the ones that are still kept as variables, respectively. The LP relaxation of Problem (**P**), given these two sets, will be denoted by $Relaxed\_gIP(X_v, X_f)$. Initially, X_f is empty and X_v contains all $\boldsymbol{x}$ variables. Note that $Relaxed\_gIP(X_v, X_f)$ is a continuous linear programming problem which can be solved by the simplex method. Upon solving the very first relaxation, we obtain the lower bound on the objective to the original problem. We then inspect values of variable $x^s_{km} \in X_v$ at optimality. These values can be categorized into the following ranges: (a) $x^s_{km} \geq 1.0 - \epsilon$, (b) $x^s_{km} \leq \epsilon$, (c) $\epsilon < x^s_{km} < 1.0 - \epsilon$, where $\epsilon > 0$ is the acceptable error margin. The values of x^s_{km}'s that fall in the range described in (a) are set to 1, while for range described in (b) are set to 0. With this set up, sets X_v and X_f are updated as, $X_v \leftarrow X_v \backslash \{x^s_{km}\}$, $X_f \leftarrow X_f \cup \{x^s_{km}\}$. The variables which has the range that fall under type (c) are left as variables, that is, they remain in set X_v. We adjust capacities and tunnels on each link based on the values of x^s_{km} which are now in X_f. We then solve the reduced problem $Relaxed\_gIP(X_v, X_f)$. We repeat the procedure till we find variables that fall only in range (b); when this occurs, we solve the problem as an ILP problem. In our experience, we have found that the original ILP problem is reduced considerably through this procedure, and the final, reduced ILP can be manageably solved by a standard branch-and-bound or branch-and-cut method.

It is possible that when the value of x^s_{km} that falls in the range of type (a) is set to 1 and the problem is rerun, the solution might be infeasible because of the capacity constraint (1.3) being violated. To avoid this problem, we keep track of current state of allocation on link capacities and tunnels. When the obtained value of x^p_{km} is of type (a), we assign it to be 1 only if none of the links violated capacity and tunnel constraints; accordingly, we update sets X_v and X_f. Otherwise, we do not update these sets and let x^p_{km} to remain as a variable. Such an issue does not arise when deciding for types (b) and (c).

4.2 Heuristic II: *gSA*

The second heuristic is based on simulated allocation [Pióro and Gajowniczek, 1997]. It works with partial allocation sequences $\boldsymbol{x} = (x^s_{km}, m \in \mathcal{P}^s_k, s \in \mathcal{S}_k, k \in \mathcal{K})$. During the execution of the algorithm, values of $\boldsymbol{w}$

Algorithm 1 gIP: Successive Approximation Approach

set $X_v = \{x^s_{km}, m \in \mathcal{P}^s_k, s \in \mathcal{S}_k, k \in \mathcal{K}\} \cup \{w^s_k, s \in \mathcal{S}_k, k \in \mathcal{K}\}$
set $X_f = \emptyset$, done $\leftarrow$ 0, change $\leftarrow$ 1
while done = 0 AND change = 1 **do**
  $\boldsymbol{x} \leftarrow$ solve $Relaxed\_gIP(X_v, X_f)$
  done $\leftarrow$ 1, change $\leftarrow$ 0
  for all $x \in X_v$ **do**
    if $x \le \epsilon$ **then**
      $X_f = X_f \cup \{x\}$
      $X_v = X_v \backslash \{x\}$
      $x = 0$, change = 1
    else if $x \ge 1.0 - \epsilon$ **then**
      $X_f = X_f \cup \{x\}$
      $X_v = X_v \backslash \{x\}$
      $x = 1$, change = 1
    else
      done = 0
    end if
  end for
end while
if done = 0 OR change = 1 **then**
  $\boldsymbol{x} \leftarrow$ solve $Integer\_gIP(X_v)$
end if
return **x**

are chosen in such a way that constraint (1.2) is always satisfied, i.e.,

$$\sum_{m \in \mathcal{P}^s_k} x^s_{km} = 0 \quad \Rightarrow w^s_k = 1, \tag{1.16}$$

otherwise $w^s_k = 0$; for all $s \in \mathcal{S}_k, k \in \mathcal{K}$. Additionally, we define

$$c(\boldsymbol{x}, \ell) = \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} d^s_k \sum_{m \in \mathcal{P}^s_k} [\delta^{s\ell}_{km} + \alpha^s_k \beta^{s\ell}_{km}]\, x^s_{km} \tag{1.17a}$$

and

$$t(\boldsymbol{x}, \ell) = \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} \sum_{m \in \mathcal{P}^s_k} [\delta^{s\ell}_{km} + U_{\{\alpha^s_k > 0\}} \beta^{s\ell}_{km}]\, x^s_{km}. \tag{1.17b}$$

Note that $c(\boldsymbol{x}, \ell)$ and $t(\boldsymbol{x}, \ell)$ determine the present state of the constrained resources (allocated capacities and tunnels on link ℓ) for a given allocation sequence $\boldsymbol{x}$. A path m' of set $\mathcal{P}^s_k$ is said to be an *accessible*

Algorithm 2 *gSA*: Simulated Allocation Approach

```
step ← 0
count ← 0
F* ← ∞
(x, w) ← (0, 1)
while (step < step_max AND F* > F*_min) do
  step = step + 1
  if x ∈ M then
    allocate(w)
  else
    if random ≤ q(x) then
      deallocate_1(w)
    else
      deallocate_2(w)
    end if
    if f(x, w) < F* then
      F* = f(x, w)
      (x, w)_min ← (x, w)
    end if
  end if
end while
return (x, w)_min
```

path from present allocation sequence $\boldsymbol{x}$ if

$$c(\boldsymbol{x}, \ell) + [\delta^{s\ell}_{km'} + \alpha^s_k \beta^{s\ell}_{km'}] d^s_k \le C_\ell \;\; \ell \in \mathcal{P}^s_k \tag{1.18a}$$

and

$$t(\boldsymbol{x}, \ell) + [\delta^{s\ell}_{km'} + U_{\{\alpha^s_k > 0\}} \beta^{s\ell}_{km'}] \le T_\ell \;\; \ell \in \mathcal{P}^s_k. \tag{1.18b}$$

Thus, setting the chosen $x^s_{km'} = 1$ does not violate constraints (1.3) and (1.4). We define set $\mathcal{M}$ as the set of maximum allocation sequence, such that $\boldsymbol{x} \in \mathcal{M}$ means that for an allocation $\boldsymbol{x}$ there exists no unallocated demand ($w^{s'}_{k'} = 1$) with an accessible path m'.

The algorithm starts with $\boldsymbol{x} = 0$ and $\boldsymbol{w} = 1$. At each step, we either choose to allocate (*allocate*($\boldsymbol{w}$)) or to deallocate (*deallocate*($\boldsymbol{w}$)) based on the current state of allocation ($\boldsymbol{x}$). For $x \notin \mathcal{M}$, we execute procedure *allocate*($\boldsymbol{x}$), otherwise procedure *deallocate*($\boldsymbol{x}$). The routine *allocate*($\boldsymbol{x}$) collects all the unallocated demands ($w^s_k = 1$) and amongst them randomly chooses a $s \in \mathcal{S}_k$ of $k \in \mathcal{K}$. All the paths in the set of candidate paths $\mathcal{P}^s_k$ are chosen in the order of increasing cost (ξ^s_{km}) and checked for accessibility. The first accessible path m' that satisfies capacity and tunnel constrains is chosen, and corresponding variable $x^s_{km'}$ is set to 1.

Procedure, $deallocate(\boldsymbol{w})$, invokes another procedure, $deallocate\_1(\boldsymbol{w})$, with probability $q(\boldsymbol{x})$; otherwise, it invokes procedure, $deallocate\_2(\boldsymbol{w})$. Procedure, $deallocate\_1(\boldsymbol{w})$, randomly chooses a $s \in \mathcal{S}_k$ of $k \in \mathcal{K}$ with $w^s_k = 0$ and sets it to 1 and finds the path m' with $x^s_{km'} = 1$ and frees the resources (capacity and tunnels) used by the path and sets $x^s_{km'}$ to 0. On the other hand, $deallocate\_2(\boldsymbol{w})$ evaluates current values of $c(\boldsymbol{x}, \ell)$ and $t(\boldsymbol{x}, \ell)$ and locates critically loaded links. These links ($\ell' \in \mathcal{L}$) are either critically loaded in capacity (i.e., $c(\boldsymbol{x}, \ell') = C_{\ell'}$) or in tunnel requirement (i.e., $t(\boldsymbol{x}, \ell') = T_{\ell'}$). Next, it locates a $s \in \mathcal{S}_k$ and $k \in \mathcal{K}$ with a path m' such that $\delta^{s\ell'}_{km'} x^s_{km'} = 1$ or $(U_{\{\alpha^s_k > 0\}})\beta^{s\ell'}_{km'} x^s_{km'} = 1$. For the chosen path m' of service class $s \in \mathcal{S}_k$ and demand $k \in \mathcal{K}$, it relinquishes the capacity and tunnels used by $x^s_{km'}$ and sets $w^s_k = 1$ and $x^s_{km'} = 0$.

Probability $q(\boldsymbol{x})$ in Algorithm 2 plays an important role in the convergence and solution quality of the algorithm. This depends on the current value of the objective function compared to a lower bound on the optimal objective, F^*_{min}, which is obtained from the LP relaxation of the entire problem. By considering the quantity,

$$\overline{F}(\boldsymbol{x}) = \frac{f(\boldsymbol{x}, \boldsymbol{w}) - F^*_{min}}{F^*_{min}}.$$

we set $q(x)$ as follows:

$$q(x) = \begin{cases} q_1, & \text{if } \overline{F}(\boldsymbol{x}) < \overline{F}^* \\ q_2, & \text{otherwise.} \end{cases}$$

In practice, we have found the following values to work well: $\overline{F}^* = 0.1, q_1 = 0.96, q_2 = 0.8$. We have set $\text{step}_{max} = 10,000$.

5. Results and Discussion

The primary goal of this section is to show the effectiveness of our formulation in solving the network traffic engineering problem at varied levels of protection; in particular, we show the role of the key parameters $\gamma, R, \theta^s_k, \eta^s_k$ in capturing the different goals, for both the hard and the soft requirements. In addition, we are interested in the convergence behavior of the heuristics, and the interplay between the capacity and tunnel constraints.

For our study, we have considered four example networks EN-I, EN-II, EN-III, EN-IV (Figures 1.2– 1.5). EN-I has 12 nodes, 18 links and average nodal degree (ratio of number of edges to number of nodes) of 1.5; EN-II has 6 nodes, 12 links and an average nodal degree of 2.0; EN-III has 12 nodes, 25 links and an average nodal degree of 2.08; EN-IV has 10 nodes, 26 links and an average nodal degree of 2.6. The baseline link capacity is set to 622 Mbps for each link. Three BAG demand classes are considered for each demand pair in the network at 100 Mbps, one for each of the three levels of service protection: zero-, fractional-, and full–protection; we have then associated an utility cost appropriate for each of these classes. In our case, for each $k \in \mathcal{K}$, we have set $u_k^1 = 1.0$ with $\alpha_k^1 = 0.0$ for zero-protection (service class: 's1'), while $u_k^2 = 3.0$ with $\alpha_k^2 = 0.5$ for partial-protection (service class: 's2'), and finally, $u_k^3 = 5.0$ with $\alpha_k^3 = 1.0$ for full-protection (service class: 's3'). The penalty cost for each demand and service class, η_k^s, is then computed as $\eta^*(1+\alpha_k^s)d_k^s$, where η^* is a weighing constant. For simplicity, we will use a single value for normalization factor among different requests and demand pairs, i.e., $\theta_k^s = \theta$ for $s \in \mathcal{S}_k, k \in \mathcal{K}$. Thus, our key parameters reduces to γ, R, θ, and η^*.

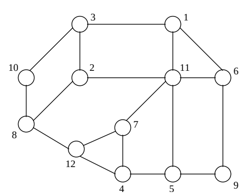


Figure 1.2. Experimental Network I

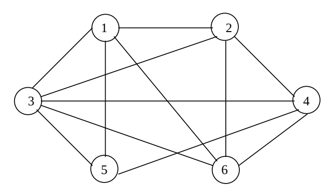


Figure 1.3. Experimental Network II

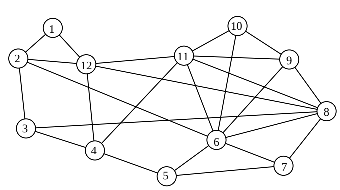


Figure 1.4. Experimental Network III

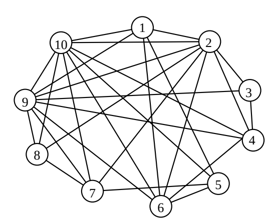


Figure 1.5. Experimental Network IV

We have implemented Algorithm 1 (gIP) in C^{++} using CPLEX callable libraries to solve the LP relaxations. We have implemented Algorithm 2 (gSA) using C^{++}. A component that feeds both the heuristics is the generation of candidate cycle paths. It may be noted that Suurballe and Tarjan have developed an algorithm for generating shortest pair of disjoint paths [Suurballe, 1974], [Suurballe and Tarjan, 1986]; this, however, helps in generating only the shortest cycle, not a set of candidate cycles. In our case, we have implemented a simple procedure by extending the k-shortest path algorithm ([Lawler, 1976]) where the k paths generated by the k-shortest path algorithm are compared to each other to filter out common links to generate a set of candidate cycles containing only disjoint pair paths. For the test networks we considered in our study, the determination of the candidate cycles through this procedure took only a few seconds of computing time. Note that the candidate set is *only* a feeder to the optimization model, and certainly, the eventual solution can depend on how many candidate cycles are included at the beginning. Based on our preliminary study, we have found that there was no difference in solution quality when the number of candidate cycles is five or more for EN-II, while some minor differences were found in solution quality for the other test networks until about 15 candidate cycles; thus, in all our studies reported below, we have set the number of candidate cycles to 5 for EN-II, and 15 for the others.

5.1 Comparison of gIP and gSA

For comparative study of heuristics gIP and gSA, we use Problem ($\mathbf{P}$) with the hard requirement, i.e., Problem ($\mathbf{P}_h$). In our study, we have set parameter values as $\theta = 0.5$, $\eta^* = 5$, $R = 100$, and $\gamma = 0.5$. We have run cases with various tunnel limits (T_ℓ); here, we will report when tunnels limit on each link was set to 50, 15, 25, 20 for EN-I, EN-II, EN-III, EN-IV, respectively. We have started the capacity at the baseline value (622Mbps) and increased it up to 400%. The combination of tunnel values and the change in capacity allows us to understand the relation between them; we will be exploring the variation in number of tunnels later in Section 1.5.3. Besides gIP and gSA, we also use a hybrid heuristic gIP+gSA where we first solve Problem ($\mathbf{P}_h$) using gIP and then use the final solution of gIP as the starting point in gSA, for further improvement using gSA. Note that the continuous relaxation of the binary variables of Problem ($\mathbf{P}_h$) serves as the lower bound on the integer solution obtained; this is marked as "LP" in results reported.

We report our results in Figures 1.6 to 1.9. From the results, we can see that Heuristic gIP is a very good method in practice and often gives results close to the LP lower bound; on the other hand, Heuristic gSA was not as effective unless the network has plenty of capacity. Nevertheless, in several instances, we have found the hybrid method, gIP+gSA, to give a better objective value than just gIP. Thus, in

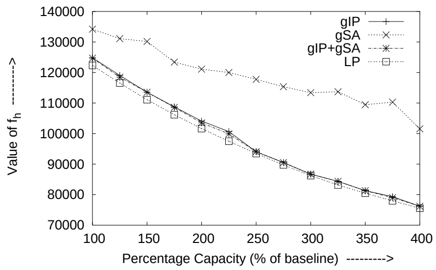


Figure 1.6. Value of f_h for EN-I with increasing capacity and $T_\ell = 50$

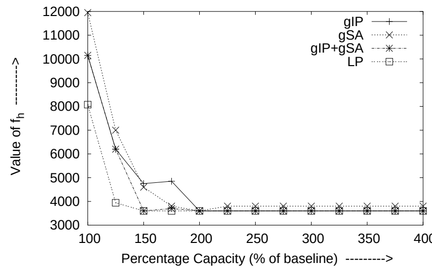


Figure 1.7. Value of f_h for EN-II with increasing capacity and $T_\ell = 15$

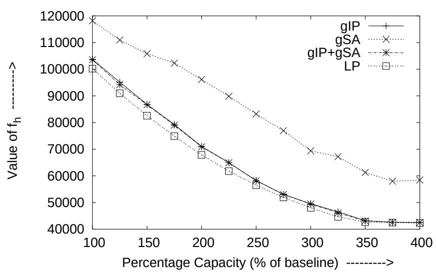


Figure 1.8. Value of f_h for EN-III with increasing capacity and $T_\ell = 25$

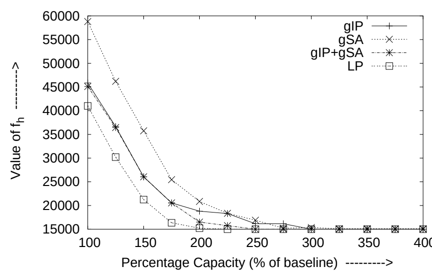


Figure 1.9. Value of f_h for EN-IV with increasing capacity and $T_\ell = 20$

the rest of the paper, we will report results obtained using the hybrid method, $gIP{+}gSA$.

5.2 Effectiveness of Formulation ($\mathbf{P}_h$)

We now discuss detailed results for Problem ($\mathbf{P}_h$), especially to understand its effectiveness through key parameters γ, R, θ, and η^*. For this purpose, it is helpful to consider a set of measures or indicators, instead of considering the objective function value. We define the following measures for this purpose:

- APR — Average Path Ratio in terms of length of the primary path to the backup path
- MRC — Minimal Residual (normalized) Capacity ($= \min_{\ell \in \mathcal{L}} \frac{\hat{r}_\ell}{C_\ell}$)
- FAD — Fraction of Accepted Demands (out of total demand requests)

The above measures are common measures of importance to network providers in regard to provisioning of services in a network; they will

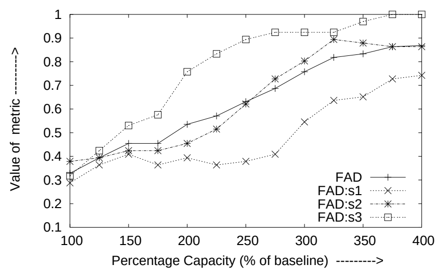


Figure 1.10. FAD for EN-III when capacity is increased

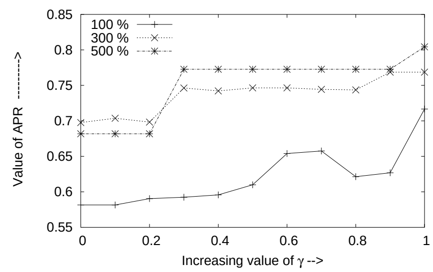
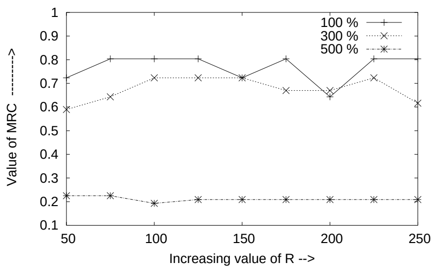


Figure 1.11. Value of APR for EN-III with increasing γ

be discussed in relation to dependency on the key parameters which a network provider can tune depending on the importance of one or more over other goals within their network. For brevity, we will discuss our results for example network EN-III where we have set $T_\ell = 50$ (impact due to different values of tunnel constraint will be discussed later in Section 1.5.3). The default values of key parameters are: $\gamma = 0.5$, $R = 100$, $\theta_k^s = 0.5$, and $\eta^* = 5$ except for change of values when one of these parameters is varied independently. First, we start with impact on relative prioritization of service classes: zero-protection, partial-protection, and full-protection.

Relative Prioritization of Service Classes: In Figure 1.10, we show the measure FAD for network as a whole and also for each protection service classes. We look at this measure as the capacity of the network is increased. For baseline capacity, some demands are accepted; as capacity is increased, more demands are accepted into the network which is expected. It is important to note that in all cases, most demands are accepted for the service class with full-protection ('s3'). It may however be surprising why not all demand requests with full-protection are accepted; this again depends on the value of the key parameters, in particular, η^* (which can be increased to let more demands be carried, rather than penalized) and R with u (revenue of a particular service class is increased depending on the importance of the service class).

Dependence on γ: We have also evaluated the role of γ; recall that γ is the routing cost weighing factor for the backup path. When γ is set to zero, there is no routing cost for backup path where as when it is set to one, backup path has full routing cost based on number of hops and allocated flow. For experiments with increasing value of γ, we compute APR for all service classes and demands with protection requirement



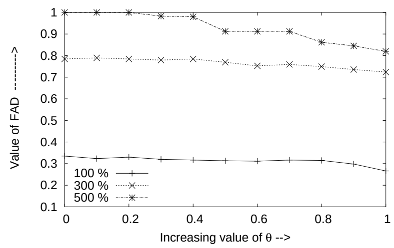


Figure 1.12. Value of MRC for EN-III with increasing R

Figure 1.13. Value of FAD for EN-III with increasing θ

($\alpha > 0$) for three different values of the network capacity. We present results in Figure 1.11 for the EN-III.

We can see that as γ increases, the value of the ratio APR increases as well. Since the routing cost of primary path is always more than the backup path, in the presence of additional capacity our model, on average, results in provisioning that prefers shorter primary path.

Dependence on *R*: We study the impact of the utility weighing parameter R on the allocation of demands; it determines the overall utility as compared to the routing costs and the bandwidth/tunnel requirement costs. Thus, we consider the measure MRC for increasing values of R. We present results in Figure 1.12 for EN-III. It may be noted that change in R does not seem to impact as R increases. As expected, with increase in network bandwidth, MRC also increases.

Dependence on θ: We now consider impact of changing θ; recall that θ controls the importance given to the allocation cost of a request as compared to its normalized routing cost. When set to zero, only routing cost is accounted for in the optimization formulation; when set to one, both have the same weight and hence play equally important roles. Due to its role in determining the cost of a request, the measure FAD is the most relevant measure. We present results for EN-III in Figure 1.13. We note that FAD decreases with increase in the value of θ.

Due to the increase in θ, for some demands the penalty cost becomes less than the minimum cost of the associated cycle. Such a scenario forces the model to reject demand. More so, the cycles with higher number of hops pay heavier penalty. Observe that the impact of increasing θ is only realized when the overall cost of the minimal cost of accessible

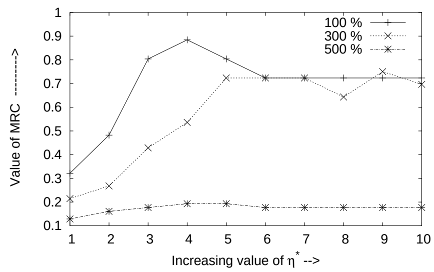


Figure 1.14. Value of MRC for EN-III with increasing η^*

path increases above the penalty cost for rejecting demand (η). Thus, we find regions in values of θ where FAD remains unchanged.

Dependence on $\boldsymbol{\eta^*}$: The parameter, η^*, appears in the form of penalty for not accepting a specific demand request. For a sufficiently high value of η^*, the network would accept all the demands that it can carry, leaving minimal or no bandwidth for best-effort traffic. However, if we choose very low value for η^*, most or all of the demands will be rejected and network would be largely under utilized. In this case, the measure, FAD, is of interest for different values of η^*. We present results for EN-III in Figure 1.14. For a tightly capacitated network, there is not much difference as η^* increases. However, a network with moderate bandwidth that can accept more demand, we do see that MRC is low when η^* is small, and as expected, increases dramatically as η^* increases.

5.3 Impact of number of Tunnels

In this section, we study the impact due to restriction on the number of tunnels on each link. For this study we use the Formulation ($\mathbf{P}_h$). We consider two measures, MRC and FAD, for increasing network capacity and increasing number of tunnels (see Figures 1.15- 1.18). We have set the value of other key parameters as follows: $R = 100$, $\eta^* = 10$, $\theta = 0.5$ and $\gamma = 0.5$.

Note that the value FAD increases with increasing capacity until it stabilizes based on the allowed number of tunnels on the links. After that, the excess amount of capacity is of no consequence since the links are “congested” in terms of number of tunnels. Increasing the number of tunnels on each link affects the solution in two ways. One way is the direct increase in the value of FAD and a consequent decrease in the

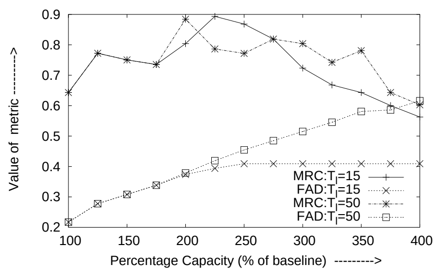


Figure 1.15. MRC and FAD for EN-I with increasing capacity

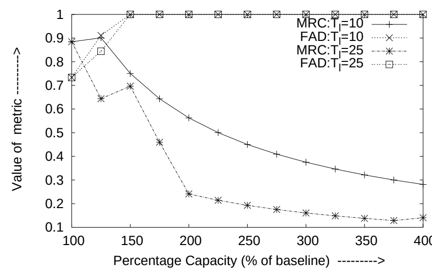


Figure 1.16. MRC and FAD for EN-II with increasing capacity

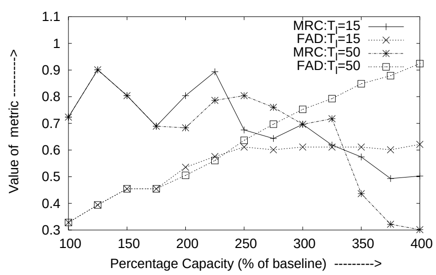


Figure 1.17. MRC and FAD for EN-III with increasing capacity

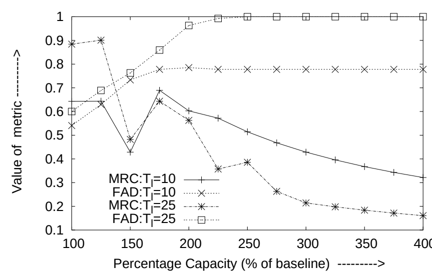


Figure 1.18. MRC and FAD for EN-IV with increasing capacity

value of MRC. The acceptance of more demands makes links utilization go higher and leave lesser bandwidth for best-effort traffic. It allows demands to use paths with less number of hops which had excess capacity but no extra tunnels. With increasing the limitation on tunnels on links, many more shorter paths (mostly less in overall cost too) become accessible.

The above mentioned behavior can lead to an increase in the value of MRC for increasing tunnels and increasing FAD. As demands begin to choose cycles with fewer hops, additional capacity is freed from the network. This capacity is used by other demands in a similar efficient way. Similar behavior on the part of all the demands leads to solutions with smaller value of MRC.

When we consider increasing capacity of links, we see a similar behavior. As we increase the amount of capacity, the value of MRC shows a sudden decrease (assuming same FAD), since some of the demands using the longer paths can now be moved on to shorter paths which leads to significant decrease in the value of MRC. However, once all the demands are moved to minimum hop cycles, the subsequent decrease in the value of MRC is only obtained by the increase in the capacity of each link.

We specifically consider the behavior of EN-II (see Figure 1.16). Note that the value of FAD reaches 1.0 at 150% of baseline capacity at which value MRC is almost equal for both the cases ($T_\ell = 10, 25$). When we increase the capacity to 200% of the baseline capacity, the value of MRC for $T_\ell = 25$ falls dramatically as compared to $T_\ell = 10$. This again is due to the freed up capacity in the network as the demands shift from longer to smaller hop cycles. At 200% of the baseline capacity, all the demands have preferably chosen minimal hop cycles and hence the subsequent decrease in MRC is only due to increasing capacity (similar to that of $T_\ell = 10$).

The results demonstrate that capacity and tunnels are equally important while provisioning a network. Presence of fewer number of tunnels nullifies the presence of abundant capacity leading to under utilized links. More so, having too many tunnels is useful only when sufficient capacity is available in the network. Thus, our inference is that accounting for both capacity and tunnels leads to effective traffic engineering solutions. Both of them could be viewed as resource which impact the amount of traffic carried by a network.

5.4 Effectiveness of formulation ($\mathbf{P}_s$)

In this section, we study the soft requirement formulation on its capability to incorporate the various objectives in an integrated fashion. Note that soft requirement is based on the circumstances that the backup paths are only found *apriori* but are not reserved, rather they are used by best-effort traffic during normal operating conditions. In the event of failures, the affected demands are routed over the already chosen backup paths preempting best-effort traffic. Here, there is no cost of reserving the backup paths since the capacity is still used by best-effort traffic; consequently, parameter γ has no role to play. Such a difference fundamentally changes the way the formulation ($\mathbf{P}_s$) is affected by changes in the parameters. In Figures 1.19-1.21, we present results on impact of changes in parameters R, θ and η^*. We have used the number of allowed tunnels as $T_\ell = 15, 20, 50, 50$, the number of candidate cycles as $5, 15, 15, 15$ and the capacity of the links as 150%, 200%, 400%, 500% of the baseline capacity for example networks EN-I, EN-II, EN-III, and EN-IV, respectively. The default values of the key parameters are: $R = 100$, $\theta_k^s = 0.5$, and $\eta^* = 5$.

The impact of increase in R on the minimum residual capacity (MRC) is shown in Figure 1.19. The behavior observed is similar to the hard

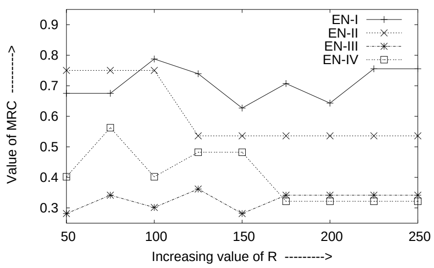


Figure 1.19. Value of MRC for example networks with increasing R

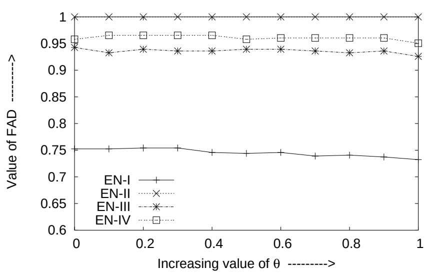


Figure 1.20. Value of FAD for example networks with increasing θ

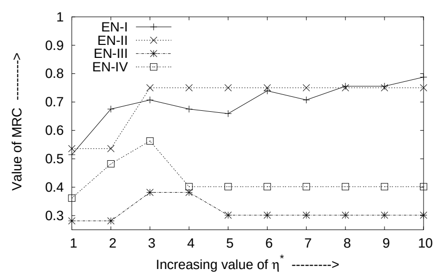


Figure 1.21. Value of MRC for example networks with increasing η^*

requirement formulation (1.12), although there are minor differences. Here, we have smaller values of routing related cost since this cost component does not include the cost of backup path; thus, smaller values of R provide the same overall cost to a demand. This leads to similar behavior of soft requirement formulation at smaller values of R as that of hard requirement formulation at higher values of R.

Similarly, as θ increases, we note that the behavior shown in Figure 1.20 is similar to that of hard requirement formulation. shown in Figure 1.13. On a closer look, we observe that the impact of θ has been diluted. The decrease in the value of FAD is much less than the one observed for hard requirement. This can be attributed to the absence of routing cost for the backup paths. Note that θ determines the routing cost vis-a-vis allocation cost of a cycle. Since for soft requirement, the cost of cycle is reduced to that of the primary path, the impact of θ towards the overall cost is reduced.

The impact of increase in η^* on the FAD is shown in Figure 1.21. The minor differences between the soft and hard requirements are due to the relative change of the value of the routing cost component. Here, we only have the cost of primary path (no cost for backup path) and hence smaller values of η^* provide same relative over all cost to a demand. This leads to similar behavior of soft requirement formulation at smaller values of η^* as that of hard requirement formulation at higher values.

In general, model with soft requirements, $(\mathbf{P}_s)$, leads to dilution of impact of parameters on the solution as compared to the hard requirement $(\mathbf{P}_h)$. Our results above confirm this behavior.

6. Summary

In this paper, we consider the problem of traffic engineering a backbone network supporting services with varied protection requirements. We have developed a novel modeling approach by using a cycle path concept so that service classes with different protection requirements can all be modeled in the same framework. We also introduce different objectives, with seemingly different goals, into a unified, single objective function. The objectives considered not only accounted for the allocation of the service classes but also the bandwidth available for the best-effort service class. Moreover, we have considered hard and soft requirements in regard to provisioning of backup paths ahead of time or only during a failure.

Models formulated are integer linear programming problems for which we have presented two heuristic methods. First heuristic is iterative in nature and uses continuous relaxations of the problem to derive integer solutions. Second heuristic is based on Simulated Allocation technique. We compare the two heuristics and show that a hybrid approach combining both the methods works quite well. We then proceed to show the effect on various key parameters that are integral to the integrated traffic engineering problem. We have presented extensive example results showing that solving the problem based on the aggregated objective function can satisfy a variety of measures typically of interest to a service provider in a satisfactory manner. We also discuss the interplay between various parameters and resources and show their relative impact on each other. Tradeoff between accepting new requests of survivable classes and the residual bandwidth for the best-effort services was also evaluated. The results also showed that capacity and tunnels can have equally important roles in ensuring effective traffic engineering of a network.

Acknowledgments

We thank Michał Pióro for his suggestions on how to apply the Simulated Allocation method for solving the formulation presented in this paper. This work is supported in part by DARPA and Air Force Research Lab under agreement no. F30602-97-1-0257.

References

Aubin, R. and Nasrallah, H. (2003). MPLS Fast Reroute and Optical Mesh Protection: A Comparative Analysis of the Capacity Required for Packet Link Protection. *Proc. Design of Reliable Communication Networks (DRCN'2003)*, pp. 349–355, Banff, Canada.

Awduche, D., Malcolm, J., Agogbua, J., O'Dell, M. and McManus, J. (1999). Requirements for Traffic Engineering Over MPLS. Internet RFC 2702, http://www.ietf.org/rfc/rfc2702.txt.

Davie, B. and Rekhter, Y. (2000). *MPLS: Technology and Applications*. Morgan Kaufmann Publishers.

Fumagalli, A., Cerutti, I., Tacca, M., Masetti, F., Jagannathan, R., and Alagar, S. (1999). Survivable networks based on optimal routing and wdm self-healing rings. In *Proceedings of INFOCOM*. IEEE Press.

Kajiyama, Y., Tokura, N., and Kikuchi, K. (1994). An ATM VP-based self healing ring. *IEEE Journal on Selected Areas of Communication*, 12(1):171–187.

Kawamura, R., Sato, K., and Tokizawa, I. (1994). Self healing ATM networks based on virtual path concept. *IEEE Journal on Selected Areas of Communication*, 12(1):120–127.

Kodialam, M and Lakshman, T. V. (2000). Dynamic routing of bandwidth guaranteed tunnels with restoration. In *Proceedings of INFOCOM*. IEEE Press.

Krithikaivasan, B., Srivastava, S., Medhi, D., and Pióro, M. (2003). Backup path restoration design using path generation technique. In *Proceedings of Design of Reliable Communication Networks (DRCN)*, pp. 77-84, Banff, Canada.

Lawler, E. L. (1976). *Combinatorial Optimization: Networks and Metroids*, Holt, Rinehart, and Winston.

Medhi, D. (1991). Diverse routing for survivability in a fiber-based sparse network. In *Proceedings of International Conference on Communications*. IEEE Press.

Medhi, D. (1994). A unified approach to network survivability for teletraffic networks: Models, algorithms and analysis. *IEEE Transaction on Communication*, 42:534–548.

Medhi, D. and Khurana, R. (1995). Optimization and performance of network restoration schemes for wide-area teletraffic networks. *Journal of Network and Systems Management*, 3:265–294.

Pióro, M. and Gajowniczek, P. (1997). Solving multicommodity integral flow problems by simulated allocation. *Telecommunication Systems*, 7(1-3):17–28.

Pióro, M. and Medhi, D. (2004). *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kauffmann Publishers.

Ramamurthy, S. and Mukherjee, B. (1999). Survivable wdm mesh networks, part I - protection. In *Proceedings of INFOCOM*. IEEE Press.

Srivastava, S., Krithikaivasan, B., Medhi, D., and Pióro, M. (2003). Traffic engineering in the presence of tunneling and diversity constraints: Formulation and Lagrangean decomposition approach. In *Proceedings of International Teletraffic Congress*, Berlin. Elsevier Science.

Suurballe, J. W. (1974). Disjoint paths in a network. *Networks*, 4:125–145.

Suurballe, J. W. and Tarjan, R. E. (1986). A quick method for finding shortest pairs of disjoint paths. *Networks*, 14:325–336.

Wu, T. H. (1992). *Fiber Network Service Survivability*. Artech House.

Xiong, Y. and Mason, L. G. (1999). Restoration strategies and spare capacity requirements in self healing ATM networks. *IEEE/ACM Transactions on Networking*, 7(1):98–110.

Le Faucheur, F. and Lai (2003). W. Requirements for Support of Differentiated Services-aware MPLS Traffic Engineering. Internet RFC 3564, http://www.ietf.org/rfc/rfc3564.txt, July 2003.