

Traffic Engineering of Tunnel-based Networks with Class Specific Diversity Requirements

Shekhar Srivastava⁺ and Deep Medhi*
 Computer Science & Electrical Engineering Department
 University of Missouri-Kansas City
 Kansas City, MO 64110

Abstract

Tunnel-based networks such as Multi-protocol Label switching (MPLS) are suitable for providing diversity guarantees to different service classes or customers. Based on the number of active tunnels to handle, router capabilities can be taxed due to the limited amount of memory and/or processing power of these routers. In this paper, we present a mixed-integer linear program formulation for a traffic engineering problem where such tunnel restrictions are taken into account in addition to standard capacity constraints while addressing diversity requirement of services. Due to large size of the formulation, we also present an accompanied solution approach based on Lagrangian relaxation and sub-gradient optimization. We then present results towards impact of diversity constraint upon the tunneling and capacity restrictions. We observed that the networks having higher amounts of capacity and demands with higher level of survivability are much more sensitive to number of allowed tunnels in the network. The impact is even more prominent for sparsely-connected, large-sized networks.

Index Terms

Diversity Constraint, Tunnel-based Networks, Optimization formulation, Lagrangian Decomposition Algorithm.

I. INTRODUCTION

Recent advances in next generation networks envision the presence of service class based protection along with the traditional quality-of-service approaches such as end-to-end delay and/or jitter. Such developments are fueled by the requirement of guaranteed protection for many critical and recently introduced applications. Note that such mechanisms would come at a premium in terms of cost and complexity. This is due to extra provisioning and signalling required to support many of these mechanisms such as p-cycles [15], fast reroute [7], or backup path [14] (see [12] for a detailed survey). However, many not-so-critical applications (or their users) might not be willing to pay the extra cost; applications that can tolerate delays and degraded performance but are extremely vulnerable to ‘total loss of connection’ fall into this category. Therefore, they need a different kind of protection which should be simple, should not require complex signalling/allocation mechanisms and should ensure that single link failures do not lead to total loss of connection. Such a simple-yet-effective protection can be provided by ‘diversity constraint’. Diversity constraint ([3], [8] and [16]) ensures that demands are allocated such that individual links/nodes carry less than a specified fraction (h) of any demand. This ensures that in the event of link/node failures, at least $(1-h)$ fraction of the total demand can still be carried. This averts the possibility of total loss of connection for these applications in the event of link or node failures.

Such a restriction can be enforced if the network has the capability to do so. For this purpose, we consider a tunnel-based network such as MPLS (multi-protocol label switching) network. Other tunnel-based networks include ATM (Asynchronous Transfer Mode), WDM (wavelength division multiplexing) based optical networks, and so on. For MPLS networks, multiple label switched path (LSP) tunnels could be set up between source and destination nodes; allocation to these tunnels could be such that individual links/nodes do not carry more than h fraction of a demand. It is obvious that having such a restriction would not require any complex reallocation or reconfiguration procedures but would drastically increase the active number of tunnels in a network. This poses a limitation for MPLS networks [1] because each LSP tunnel setup would require additional processing power and memory at the router interfaces (discussed further in Section II).

Therefore, our interest here is to consider traffic engineering of a network with multiple service classes, each having a pre-determined diversity requirement, and consider any restriction on number of tunnels imposed by the processing power and memory requirements of the LSRs. While we use MPLS and LSPs to explain the problem, the model is applicable to other tunnel-based networks as well.

Original submission: July 2005; Final submission: April 2006.

⁺ Current Affiliation: Schema, Inc., Rochelle Park, New Jersey, USA.

* Corresponding author: dmedhi@umkc.edu

A. Contributions of the Paper

We start with presenting our approach towards capturing the restriction of processing power and memory requirements of the LSRs. We determined that they translate into number of active LSP tunnels on a link referred to as tunneling restriction.

Furthermore, we present a mathematical formulation which incorporates tunneling, capacity and diversity constraint in an integrated mixed integer linear program. The formulation neatly enforces the tunneling constraint by introducing a tunnel activity variable. Diversity constraint is introduced as a bound on the flow variables. We also extend the basic model to introduce a tunnel cost. The formulation was found to be having large number of binary variables and constraints, which makes the formulation hard to solve for large sized networks. We also present a solution approach based on Lagrangian relaxation with dual subgradient optimization (LRDSO). Since the early paper by Held et al [4], LRDSO has been found to be an effective approach for solving many large-scale optimization problems; for a discussion and literature related to problems in communications network design that use LRDSO, see the recent book [12]. In our case, LRDSO results in solving smaller sub-problems in an iterative fashion and then in constructing the final solution. It may be noted that one of the sub-problems is of integer linear in nature; we also present a proof that showed that the continuous relaxation of the sub-problem had an optimal binary solution which can be obtained by solving its continuous relaxation using the simplex method. Our work here extends the work presented in the conference paper [13] in multiple ways: 1) we present an extended cost model to address for one of the key limitations of the earlier model, 2) we present the LRDSO method for the extended model, 3) we present here extensive computational results for the extended model by considering various factors and parameters, such as tunnel restriction and diversity, including an assessment on survivability due to diversity.

Using the solution technique, we study the interplay between capacity, tunneling and diversity constraints. It was observed that minor relaxations in the survivability requirement of demands translates into much fewer values of required tunnels to achieve similar performance. We also observed that the networks having higher amounts of capacity and demands with higher level of survivability are much more sensitive to number of allowed tunnels in the network. The impact is even more prominent for sparsely-connected, large-sized networks.

The problem of minimizing the required label space was discussed by Applegate and Thorup in [1]. They evaluated the number of required labels for each LSR in order to successfully embed any given optimal solution. This is achieved by using “to trees” and they present a construction and an accompanied proof to show that such an allocation is possible for any given optimal allocation. The approach is interesting and gives a lower bound towards the design specification for to-be-installed LSRs. We, however take a different view point. How can the capability of routers (in terms of the tunnels/labels that they can support) be embedded in a traffic engineering formulation and what is its impact? We assert that such a view point is more practical. This is for two reasons. First is that most traffic engineers are given LSRs connected by links and projected demands and asked to determine routing. The possibility of being able to enforce/upgrade the capabilities of the routers in the network might not be a viable option. Secondly, MPLS networks could be comprised of wide variety of LSRs ranging from slow/less capable legacy LSRs to fast/very capable, recently installed LSRs. Enforcing the same capability in terms of storage memory and processing power could require discarding of many otherwise-useable LSRs.

The rest of the paper is organized as follows. In Section II, we study the details of processing and memory requirements for an LSR to support large number of tunnels. In Section III, we present the traffic engineering formulation. In Section IV, we present the solution approach based on a decomposition algorithm to solve the formulation. In Section V, we present results for demonstrating the interplay between capacity, tunneling and diversity constraints. We conclude in Section VI

II. CONSTRUCTING TUNNEL CONSTRAINT

In this section, we discuss the functional details of an LSR in order to isolate the operations which could become a bottle-neck in its functioning, in the event of overwhelming number of tunnels being setup in the network.

An edge LSR does four fundamental operations in an MPLS network. First operation is the computation of the best path that a packet should take through the network to its destination. This computation could incorporate various policies and network constraints. The operation is performed on regular intervals by the route processor and the frequency of operation is configured by the service provider.

The second operation is performed when a new originating/terminating/passing LSP is being set up or an older originating/terminating/passing LSP is being re-provisioned. Set-up of LSP requires activation of label distribution protocol (LDP) which appends a label in the label forwarding information base (LFIB) of each incoming interface of the intermediate LSRs of the LSP. The route chosen by the LSP is determined by the best path computed during the first operation.

Third operation is performed for each and every incoming packet on an interface. Label values are extracted from the label field found in the incoming packet on an interface and used as an index in the LFIB. After a match is found, the interface replaces the label in the packet with the outgoing label from the subentry and sends the packet over the specified outgoing interface to the next hop specified by the subentry. In case the node happens to be the destination of the packet (or the corresponding LSP), the label is removed and the packet is sent to the local interface.

The fourth function corresponds to the actual transmission of the packet over the physical link on the interface determined by the subentry towards the next hop. In case the transmission channel is busy, the packet is placed in the specified queue

and transmitted in the order of its arrival or any scheduling mechanism followed by the outgoing interface. Although an LSR could be involved in many other operations and functions, we assume that these four operations as most impactful functions towards the consumption of the processing power and memory requirements of an interface of the LSR.

Observe that the first operation is independent of the number of tunnels set up in the MPLS network. If we assume that the traffic does not vary too much over medium to large time scales, then second operation of configuring/reconfiguring LSPs would not cause major concerns in terms of overload. However, the interface card needs to have sufficient memory to store all the required entries of the labels for each incoming LSP. Observe that the size of LFIB increases linearly with the increasing number of incoming LSPs on the interface. The fourth operation is impacted by the amount of allocated traffic and available bandwidth on the out going link and is mostly referred to as *capacity constraint* in the literature ([11], [17]).

However, third operation is performed for every incoming packet on the interface of a LSR. The processing power at each interface should be enough to successfully determine the output interface and label of each incoming packet in an acceptable time. Observe that the time required to find the entry (lookup operation) for each packet increases with larger size of the LFIB. Therefore, the processing requirement at an interface of a LSR increases with the number of active LSPs passing through the interface.

In conclusion, for tunnel-based networks, such as MPLS, number of tunnels incoming on an interface determine the memory and processing requirements of the interface. Hence restricting the active number of tunnels on an interface would avoid overloading the LSRs. This restriction is equivalent to restricting the number of tunnels passing through the link connected to the interface assuming that the interfaces connected to the link have the same capabilities. For links connected to interfaces having different capabilities (connecting slow and fast routers), we determine the tunnel restriction based on the slower/less capable router. We refer to the approach of restricting the active number of tunnels on a link as tunneling restriction/constraint.

III. TRAFFIC ENGINEERING FORMULATION

In this section, we present an integrated traffic engineering formulation which incorporates capacity, tunnel and diversity restrictions. Capacity restriction is incorporated by ensuring that the flow on any link is less than the capacity of the link. Tunnel constraint is honored by enforcing that the active number of tunnels on a link are restricted. Diversity constraint restricts the fraction of demand that can be allocated to any one tunnel.

We consider an aggregated-flow based network, where traffic data (packets) arriving to a source, for a specific destination needs to be sent over one of the active tunnels between the source and the destination. The data volume is estimated using either service level agreement based approaches [11] or using measurement based concrete measures [17]. Other techniques could also be used (see [9], and references there in). We assume that information is available regarding data volume for each service class of all the demands. We also assume that tunnels are not shared between the service classes. Each service class maintains its own set of tunnels between source and destinations. We first describe the notation:

\mathcal{N} :	Set of nodes in the Network
\mathcal{K} :	Set of nodegroups with traffic
\mathcal{L} :	Set of links
C_ℓ :	Capacity of link ℓ
\mathcal{S}_k :	Set of service classes at nodegroup k
T_ℓ :	Maximum number of tunnels allowed on link ℓ
ζ_k^s :	Revenue from carrying unit demand volume of service class s and nodegroup k
d_k^s :	Traffic demand for service class s and nodepair k
ε :	Minimum value of flow on any path
h_k^s :	Maximum fraction of flow for service class s and nodepair k allowed on any path

We are given the following information: \mathcal{N} , \mathcal{K} , \mathcal{L} , C_ℓ , T_ℓ , \mathcal{S}_k , ε , h_k^s , d_k^s and ζ_k^s . For demand d_k^s connecting a pair of ingress and egress nodes, due to the capacity, diversity and tunnel limitation, it is clear that just considering the shortest path is too limiting and does not address the overall traffic engineering problem. Hence we initially generate a set of candidate paths for each service class and demand pair. We use a k-shortest path algorithm to generate the set of paths (\mathcal{P}_k^s) for each service class of each demand.

A. Basic Model

Let $|P_k^s|$ be the number of candidate paths generated for service class $s \in \mathcal{S}_k$ of demand $k \in \mathcal{K}$. We now introduce the *fractional flow variable* x_{km}^s associated with the path m for service class $s \in \mathcal{S}_k$ of demand $k \in \mathcal{K}$ which takes a value between 0 and 1 as the fraction of demand d_k^s allocated to m^{th} path.

1) *Demand Constraint*: As discussed earlier, due to capacity, tunnel and diversity limitations, it is quite possible that a demand may not be routed (while proper network design would try to avoid such situations by over-engineering; from a traffic engineering modeling standpoint, it is necessary to incorporate this variable to avoid infeasibility of the problem). To consider this aspect, we have the following constraint

$$\sum_{m \in \mathcal{P}_k^s} x_{km}^s \leq 1.0, \quad s \in \mathcal{S}_k, k \in \mathcal{K} \quad (1a)$$

2) *Diversity Constraint*: In order to incorporate the diversity constraint, we need to restrict the fraction of the total demand which can be allocated to a tunnel. Therefore, we have to put an upper-bound on the fractional flow variable x_{km}^s . Such a bound would suffice to introduce diversity constraint. We consider h_k^s as the diversity restriction of the service class $s \in \mathcal{S}_k$ of demand $k \in \mathcal{K}$. The constraint is

$$x_{km}^s \in [0, h_k^s], \quad m \in \mathcal{P}_k^s, s \in \mathcal{S}_k, k \in \mathcal{K}. \quad (1b)$$

3) *Capacity Constraint*: Next, we construct the bandwidth requirements by restricting the flow on a link. To address the fraction of flow of a service class of the demand on each link (for each path), we now introduce the indicator notation to map between the demand, the service class and the link, as they relate to paths as follows:

$$\delta_{km}^{s\ell} = \begin{cases} 1 & \text{if path } m \text{ for service class } s \text{ of nodepair} \\ & k \text{ uses link } \ell \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the bandwidth needed on any link ℓ (denoted by F_ℓ) to carry flow for different service classes and demands can now be captured by the amount

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

Since each link ℓ has capacity C_ℓ , we thus have the following constraint for each link $\ell \in \mathcal{L}$:

$$\sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} \sum_{m \in \mathcal{P}_k^s} d_k^s \delta_{km}^{s\ell} x_{km}^s \leq C_\ell, \quad \ell \in \mathcal{L}. \quad (1c)$$

Note the constraints discussed, so far, are already present in the literature and have been thoroughly studied [11]. The following constraints are based on additional requirements enforced in this paper upon the engineering of a network.

4) *Tunnel Mapping*: Now, we consider the number of active tunnels sharing a link. At this end, we first need to have a variable that captures if any given path is being used or not. The value taken by such a variable depends on the value of the corresponding flow variable. Hence we define w_{km}^s as the (binary) tunnel activity variable, which is 1 if a path is being used to route flows and 0 otherwise. Such a functionality can be achieved by incorporating following two constraints:

$$\varepsilon w_{km}^s \leq d_k^s x_{km}^s, \quad m \in \mathcal{P}_k^s, s \in \mathcal{S}_k, k \in \mathcal{K} \quad (1d)$$

$$x_{km}^s \leq w_{km}^s, \quad m \in \mathcal{P}_k^s, s \in \mathcal{S}_k, k \in \mathcal{K}. \quad (1e)$$

These constraints incorporate and force dependencies between variables, \mathbf{x} and \mathbf{w} . Lets consider x_{km}^s is 0, constraint (1d) forces w_{km}^s to be 0. On the other hand, when w_{km}^s is 0, constraint (1e) forces x_{km}^s to be 0. Sometimes, we want to ensure that the minimal flow volume on any tunnel should be greater than a predetermined constant (ε units). The value of the constant ε can be determined based on the minimal flow volume that warrants the setting up and maintenance of a new tunnel. We use the parameter, ε , in order to limit the activation of tunnels having very low bandwidth.

5) *Tunnel Constraint*: Based on the discussion in Section II, we enforce that any link ℓ should have less than T_ℓ active tunnels. In the rest of the paper, we assume that such a number is determined before hand by considering each link and the interfaces that it is connected to. Now, we force tunnel constraint using

$$\sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} \sum_{m \in \mathcal{P}_k^s} \delta_{km}^{s\ell} w_{km}^s \leq T_\ell, \quad \ell \in \mathcal{L} \quad (1f)$$

$$w_{km}^s \in \{0, 1\}, \quad m \in \mathcal{P}_k^s, s \in \mathcal{S}_k, k \in \mathcal{K}. \quad (1g)$$

The objective of the formulation is to maximize the total revenue generated by the flow carried by the network. The traffic engineering Problem (**P**) can be formulated as

$$\bar{F} = \max_{\{\mathbf{x}, \mathbf{w}\}} f = \max_{\{\mathbf{x}, \mathbf{w}\}} \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} \sum_{m \in \mathcal{P}_k^s} \zeta_k^s d_k^s x_{km}^s \quad (2)$$

subject to the set of constraints (1). The above basic model elaborates the model presented in [13].

B. Extended Model

A limitation of the basic model is that it does not incorporate cost due to tunneling. That is, although capabilities of routers (in terms of processing/memory requirements) restrict the maximum number of tunnels on a link, each active tunnel incurs a setup/maintenance cost to the network. From the perspective of a service provider, setup and maintenance of each tunnel requires resources of the network. Certainly, enforcing the tunneling constraint ensures that such resources are available in the network, but the revenue and tunnel cost dependency captures the decision whether the resources should be allocated to a demand or not. Thus, we now discuss how the basic model can be extended.

Consider a demand from source s_k to t_k which can be carried over a path $p \in \mathcal{P}_k$ and ζ_k be the revenue generated from carrying the demand. Assume that the value of ζ_k is positive and hence carrying the demand by path p would lead to higher revenue. Also assume that the links of the path p can support the additional tunnel in terms of capacity and number of active tunnels. Then based on the basic model, the demand should be accepted. But, from the perspective of a service provider, it is possible that cost of set-up/maintenance of the tunnel using path p is more than the revenue generated by carrying the demand volume. Such a scenario is possible when path p has high number of hops because of which the cost of setting up such a tunnel and then maintaining it would be more than the revenue generated by carrying the demand. A service provider might choose to refuse such a demand to ensure overall optimum.

The current objective function considers the revenue generated by the carried flow, which is equal to

$$f_r = \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} \sum_{m \in \mathcal{P}_k^s} \zeta_k^s d_k^s x_{km}^s. \quad (3)$$

Since maximizing the function f_r does not account for the cost to the network in carrying the above mentioned flow \mathbf{x} , we need to capture the routing/maintenance cost of these flows. Let c_{km}^s is the routing cost of the candidate tunnel w_{km}^s , $m \in \mathcal{P}_k^s$, $s \in \mathcal{S}_k$ and $k \in \mathcal{K}$. Then, the total routing cost will be

$$f_c = \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} \sum_{m \in \mathcal{P}_k^s} c_{km}^s w_{km}^s. \quad (4)$$

The function f_r captures the revenue generated by the network which needs to be maximized and function f_c computes the total routing cost which needs to be minimized. Since, both the objectives are relevant in engineering a network, it is desirable to combine them into an integrated objective function. This is accomplished by weighing one of the functions by a weight factor and taking the difference with the other function and then maximizing the overall function. If we use the normalized weight for the function f_r , then the weight factor θ (specifically, θ_k^s for each $s \in \mathcal{S}_k$ and $k \in \mathcal{K}$) is needed only for function f_c . Thus, we have the combined objective function as

$$f_{rc} = \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} \zeta_k^s \sum_{m \in \mathcal{P}_k^s} d_k^s x_{km}^s - \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} \theta_k^s \sum_{m \in \mathcal{P}_k^s} c_{km}^s w_{km}^s. \quad (5)$$

Note that in general, routing cost of a tunnel can be determined based on many different aspects. On one hand, one could determine the cost of maintaining the tunnel based on only the setup and maintenance cost which will be additive in the number of hops taken by the tunnel. Such a criteria can be referred to as Hop Based Routing (HPR) cost and will be equal to

$$c_{km}^s = \sum_{\ell \in \mathcal{L}} \delta_{km}^{s\ell}.$$

On the other hand, cost of tunnel can be determined by considering the forwarding cost associated with the maintenance of the tunnel. In which case, it will be computed based on the flow allocated to the tunnel. Such a cost could be referred to as Flow Based Routing (FBR) cost and can be captured as

$$c_{km}^s = x_{km}^s d_k^s.$$

However, in our case, we compute the cost of routing a tunnel based on both the criteria, that is based on setup and maintenance as well as forwarding required to maintain the tunnel. Hence the total routing cost depends on the volume of flow on the tunnel and the number of hops and is additive in both. Such a cost could be referred to as Flow and Hop based Routing (FHR) cost and can be captured as

$$c_{km}^s = \sum_{\ell \in \mathcal{L}} \delta_{km}^{s\ell} x_{km}^s d_k^s. \quad (6)$$

Hence substituting the value of c_{km}^s in the combined objective function, we get

$$f = \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} \zeta_k^s \sum_{m \in \mathcal{P}_k^s} d_k^s x_{km}^s - \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} \theta_k^s \sum_{m \in \mathcal{P}_k^s} \left[\sum_{\ell \in \mathcal{L}} \delta_{km}^{s\ell} x_{km}^s d_k^s \right] w_{km}^s.$$

Note that the function f is non-linear in nature due to term $x_{km}^s w_{km}^s$. But, when coupled with constraints (1d) and (1e), it can be directly inferred that $x_{km}^s w_{km}^s = x_{km}^s$. Hence function f can be rewritten and rearranged as

$$\sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} \sum_{m \in \mathcal{P}_k^s} (\zeta_k^s - \theta_k^s \sum_{\ell \in \mathcal{L}} \delta_{km}^{s\ell}) d_k^s x_{km}^s. \quad (7)$$

For simplicity, we use the notation

$$\xi_{km}^s = (\zeta_k^s - \theta_k^s \sum_{\ell \in \mathcal{L}} \delta_{km}^{s\ell}), \quad m \in \mathcal{P}_k^s, s \in \mathcal{S}_k, k \in \mathcal{K}$$

We now present the extended formulation (**P**) as

$$\bar{F} = \max_{\{\mathbf{x}, \mathbf{w}\}} f = \max_{\{\mathbf{x}, \mathbf{w}\}} \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} \sum_{m \in \mathcal{P}_k^s} \xi_{km}^s d_k^s x_{km}^s \quad (8)$$

subject to the set of constraints (1).

C. Problem Size

The Problem (**P**) is a Mixed Integer Linear Program with number of variables being

- $\sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} |\mathcal{P}_k^s|$ (\mathbf{w} , binary)
- $\sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} |\mathcal{P}_k^s|$ (\mathbf{x} , continuous).

The number of constraints required to be satisfied are

- $\sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} 2 |\mathcal{P}_k^s|$ (Constraints 1d and 1e)
- $\sum_{k \in \mathcal{K}} |\mathcal{S}_k|$ (Constraint 1a)
- $2 |\mathcal{L}|$ (Constraints 1c and 1f).

For the experimental networks presented in Figures (2-5), we discuss the size of the Problem (**P**) in Table III-C. We consider three service classes and demand between every pair of nodes. We consider, 15, 6, 15 and 15 candidate paths (\mathcal{P}_k^s) for each service class of each demand of experimental networks I, II, III and IV, respectively. To illustrate the growth of the problem size, we show in Table III-C the number of variables and constraints for networks from 50- to 100-node networks, considering just one service class and five candidate paths for each demand pair.

Table 1. Size of Problem (**P**) for ENs

ENs	#Variables		#constraints
	binary	continuous	
EN I	2970	2970	6174
EN II	270	270	609
EN III	2970	2970	6188
EN IV	2025	2025	4237

Table 2. Size of Problem (**P**) for Larger Networks

Networks		#Variables		#constraints
$ \mathcal{N} $	$ \mathcal{L} $	binary	continuous	
50	250	6125	6125	13975
60	300	8850	8850	20070
70	350	12075	12075	27265
80	400	15800	15800	35560
90	450	20025	20025	44955
100	500	24750	24750	55450

Typically, generic MIPs are solved using the branch-and-bound algorithm and/or Gomory's cutting plan method. But such approaches are not quite practical for problems of large size. The Formulation (**P**) presented above grows with the size of the network, number of service classes and with the number of candidate paths considered. Therefore, using direct methods will restrict the applicability of the approach to smaller networks or fewer demands. Hence in the following section we present a solution approach based on the decomposition of the problem into smaller subproblems.

IV. DECOMPOSITION ALGORITHM

In this section, we describe a decomposition algorithm for solving mixed-integer linear programming problem (**P**) using Lagrangian relaxation with duality and subgradient optimization which has been successfully used before ([2] and [5]). Observe that constraints (1d) and (1e) are coupling constraints between variables \mathbf{x} and \mathbf{w} . We take Lagrangian relaxation around these two constraints so that the remaining constraints get decoupled from the objective function. In the process, we will show that one of the subproblems with integrality constraints can be solved by relaxing the integrality requirement. First we start with the Lagrangian,

$$L(\mathbf{x}, \mathbf{w}; \mathbf{u}, \mathbf{v}) = \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} \sum_{m \in \mathcal{P}_k^s} [\zeta_{km}^s d_k^s x_{km}^s + u_{km}^s (d_k^s x_{km}^s - \varepsilon w_{km}^s) + v_{km}^s (w_{km}^s - x_{km}^s)].$$

Rearranging, we get

$$L(\mathbf{x}, \mathbf{w}; \mathbf{u}, \mathbf{v}) = \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} \sum_{m \in \mathcal{P}_k^s} [(\xi_{km}^s d_k^s + u_{km}^s d_k^s - v_{km}^s) x_{km}^s + (v_{km}^s - \varepsilon u_{km}^s) w_{km}^s].$$

This can be written as

$$L(\mathbf{x}, \mathbf{w}; \mathbf{u}, \mathbf{v}) = L_x(\mathbf{x}; \mathbf{u}, \mathbf{v}) + L_w(\mathbf{w}; \mathbf{u}, \mathbf{v}).$$

The dual Problem (D) is

$$s_D = \min_{\{\mathbf{u}, \mathbf{v} \geq 0\}} g(\mathbf{u}, \mathbf{v}).$$

where,

$$g(\mathbf{u}, \mathbf{v}) = \max_{\{\mathbf{x}, \mathbf{w}\}} L(\mathbf{x}, \mathbf{w}; \mathbf{u}, \mathbf{v})$$

Note that for a given \mathbf{u} and \mathbf{v} , the Lagrangian L is separable in \mathbf{x} and \mathbf{w} and reduces to solving two independent subproblems

$$\max_{\{\mathbf{x}, \mathbf{w}\}} L(\mathbf{x}, \mathbf{w}; \mathbf{u}, \mathbf{v}) = g_x(\mathbf{u}, \mathbf{v}) + g_w(\mathbf{u}, \mathbf{v})$$

where

$$\begin{aligned} g_x(\mathbf{u}, \mathbf{v}) &= \max_{\{\mathbf{x}\}} L_x(\mathbf{x}; \mathbf{u}, \mathbf{v}) \\ &= \max_{\{\mathbf{x}\}} \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} \sum_{m \in \mathcal{P}_k^s} (\xi_{km}^s d_k^s + u_{km}^s d_k^s - v_{km}^s) x_{km}^s \end{aligned}$$

subject to constraints (1a), (1c) and (1b) and

$$\begin{aligned} g_w(\mathbf{u}, \mathbf{v}) &= \max_{\{\mathbf{w}\}} L_w(\mathbf{w}; \mathbf{u}, \mathbf{v}) \\ &= \max_{\{\mathbf{w}\}} \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}_k} \sum_{m \in \mathcal{P}_k^s} (v_{km}^s - \varepsilon u_{km}^s) w_{km}^s \end{aligned}$$

subject to constraints (1f) and (1g).

Thus, to solve the original Problem (P), we use the algorithmic steps shown in **Algorithm 1**.

Algorithm 1 Decomposition Algorithm

- 1: Generate \mathcal{P}_k^s for all $s \in \mathcal{S}_k, k \in \mathcal{K}$
 - 2: $\mathbf{u} = \mathbf{1}$ and $\mathbf{v} = \mathbf{1}$.
 - 3: Solve $g_x(\mathbf{u}, \mathbf{v})$ and $g_w(\mathbf{u}, \mathbf{v})$ and derive \mathbf{x} and \mathbf{w}
 - 4: Solve for s_D and update the values of \mathbf{u} and \mathbf{v}
 - 5: Check for the convergence of the iterations, if not converged: go to Step 3
-

Based on the discussion, so far, the solution to the Problem (P) consists of solving three smaller problems namely, $g_x(\mathbf{u}, \mathbf{v})$, $g_w(\mathbf{u}, \mathbf{v})$ and s_D . In the following subsections, we present approaches to solve these three problems separately.

A. Solving $g_x(\mathbf{u}, \mathbf{v})$

Observe that Problem $g_x(\mathbf{u}, \mathbf{v})$ is a linear continuous programming problem which can be efficiently solved using Simplex method for fairly large number of variables and constraints.

B. Solving $g_w(\mathbf{u}, \mathbf{v})$

Problem $g_w(\mathbf{u}, \mathbf{v})$ is a special case of Multiple Knapsack Problem (MKP). MKP is known to be NP hard. Thus, solving the General MKP problem by direct methods imposes a severe constraint on the scalability of the solution approach. In our case, the volume of each item is equal to 1 or 0 in all the knapsacks and size of each knapsack is integral, making this special case effectively solvable by faster approaches. Below we shall show that there always exists an integral solution for the LP

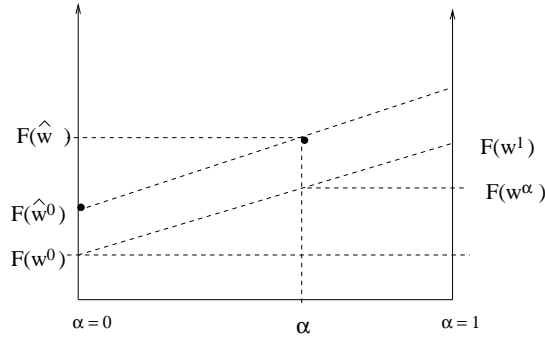


Fig. 1. Diagram with location of points

version of MKP. Moreover, at least one feasible optimum solution of the relaxed $g_w(\mathbf{u}, \mathbf{v})$ is integral and hence the use of the Simplex algorithm will give us the integral solutions. *RMKP* (Relaxed MKP) can be written as:

$$(RMKP) \left\{ \begin{array}{l} \bar{F} = \max_{\{\mathbf{w}\}} F(\mathbf{w}) = \max_{\{\mathbf{w}\}} \sum_{j=1}^n c_j w_j \\ \text{subject to} \\ \sum_{j=1}^n \mathbf{A}_j w_j \leq \mathbf{b} \\ 0 \leq w_j \leq 1, \quad (j = 1, 2, \dots, n) \end{array} \right.$$

where \mathbf{A} is a $m \times n$ matrix with non-binary entries (\mathbf{A}_j is the j^{th} column of \mathbf{A} capturing the a_{ij} 's, amount of resources required by j^{th} object in i^{th} sack), \mathbf{w} is an n -vector of variables (w_1, w_2, \dots, w_n), \mathbf{b} a m -vector with nonnegative integer components. The Dynamic Programming framework [10] is used to solve the RMKP. It is sufficient to consider the family of Problems $RMKP_k(\mathbf{E})$:

$$(RMKP_k(\mathbf{E})) \left\{ \begin{array}{l} F_k(\mathbf{b}') = \max_{\{\mathbf{w}\}} F_k(\mathbf{b}', \mathbf{w}) \\ = \max_{\{\mathbf{w}\}} \sum_{j=k}^n c_j w_j \\ \text{subject to} \\ \sum_{j=k}^n \mathbf{A}_j w_j \leq \mathbf{b} - \mathbf{b}' \\ 0 \leq w_j \leq 1 (j = k, k+1, \dots, n) \end{array} \right.$$

for k varying from 1 to n , and where \mathbf{b}' is a state vector of dimension m , $\mathbf{b}' = (b'_1, b'_2, \dots, b'_m)$, every component b'_i is continuous and can take values in $[0, b_i]$. The recurrence relation between the values of $F_k(\mathbf{b}')$ reads:

$$F_k(\mathbf{b}') = \max_{0 \leq w_k \leq 1} \{c_k w_k + F_{k+1}(\mathbf{b}' + \mathbf{A}_k w_k)\}$$

for k decreasing from n to 1 and $\bar{F} = F_1(\mathbf{0})$.

The following is a useful result for solving RMKP. The sketch of the proof was originally given in [13] and is included here in its entirety.

Theorem 1: Relaxed Multiple Knapsack Problem (RMKP) with $a_{ij} = \{0, 1\}$ and integer b has at least one binary solution in the set of all optimal solutions.

Proof: We prove the theorem by mathematical induction using the Dynamic Programming approach. We intend to prove that for each, $k = n, n-1, \dots, 1$ and each integer vector $\mathbf{b}' \leq \mathbf{b}$, we have an optimal binary solution of $RMKP_k(\mathbf{b}')$ with $w_j \in \{0, 1\}$ for $j = k, k+1, \dots, n$.

The initial inductive assumption is satisfied as the solution of $RMKP_n(\mathbf{b}')$ is binary, since

$$F_n(\mathbf{b}') = \max_{0 \leq w_n \leq 1} c_n w_n = \begin{cases} c_n & (w_n = 1) \\ 0 & (w_n = 0) \end{cases} \quad \text{if } \mathbf{b} - \mathbf{b}' \geq \mathbf{A}_n \\ \text{otherwise.}$$

Note that for integral \mathbf{b} and \mathbf{b}' , $\mathbf{b}' \leq \mathbf{b}$, as an object either fills totally or not at all. Hence for $k = n$, variable w_n takes either the value 0 or 1.

Now assume that for all integer vectors \mathbf{b}' such that $\mathbf{b}' \leq \mathbf{b}$, there exists a binary optimal solution for $RMKP_{k+1}(\mathbf{b}')$. We now need to show that $RMKP_k(\mathbf{b}')$ has a binary solution for each $\mathbf{b}' \leq \mathbf{b}$. Let $(w_{k+1}^0, w_k^0, \dots, w_n^0)$ be the optimal solution

for $RMKP_{k+1}(\mathbf{b}')$ and $(w_{k+1}^1, w_k^1, \dots, w_n^1)$ for $RMKP_{k+1}(\mathbf{b}' + \mathbf{A}_k)$, and consider the two feasible binary solutions of $RMKP_k(\mathbf{b}')$:

$$\begin{aligned} \mathbf{w}^0 &= (0, w_{k+1}^0, \dots, w_n^0): \text{ with } F(\mathbf{w}^0) = F_{k+1}(\mathbf{b}') \\ \mathbf{w}^1 &= (1, w_{k+1}^1, \dots, w_n^1): \text{ with } F(\mathbf{w}^1) = c_k + F_{k+1}(\mathbf{b}' + \mathbf{A}_k), \end{aligned}$$

$w_i^0, w_i^1 \in \{0, 1\}$ for $i = k+1, k+2, \dots, n$. Let

$$\begin{aligned} \mathcal{O}_1 &= \{w_j \mid w_j^0 = 0 \text{ and } w_j^1 = 1, j = k+1, k+2, \dots, n\} \\ \mathcal{O}_2 &= \{w_j \mid w_j^0 = 1 \text{ and } w_j^1 = 0, j = k+1, k+2, \dots, n\} \\ \mathcal{O}_3 &= \{w_j \mid w_j^0 = w_j^1, j = k+1, k+2, \dots, n\}. \end{aligned}$$

Consider a set of feasible solutions of $RMKP_k(\mathbf{b}')$ defined as $\mathbf{w}^\alpha = \mathbf{w}^0 + \alpha(\mathbf{w}^1 - \mathbf{w}^0)$. The solution \mathbf{w}^α consists of fractional values for some of its components and can be written in the following form:

$$\begin{aligned} w_j^\alpha &= w_j^0 + \alpha & j \in \mathcal{O}_1 \\ w_j^\alpha &= w_j^0 - \alpha & j \in \mathcal{O}_2 \\ w_j^\alpha &= w_j^0 & j \in \mathcal{O}_3 \end{aligned}$$

Observe that \mathbf{w}^α is a feasible point since by construction \mathbf{w}^α , is a convex combination of \mathbf{w}^0 and \mathbf{w}^1 , since $\mathbf{w}^\alpha = (1-\alpha)\mathbf{w}^0 + \alpha\mathbf{w}^1$. We define $F(\mathbf{w}^\alpha) = c_k\alpha + F_{k+1}(\mathbf{b}' + \mathbf{A}_k\alpha)$ which is the revenue when the knapsack contains exactly a fraction α of object k and some amounts of objects $\{k+1, k+2, \dots, n\}$ (fractional are possible) which are required to be chosen to fill optimally a knapsack smaller by $\mathbf{A}_k\alpha$ in volume. We show that the Problem $RMKP_k(\alpha\mathbf{A}_k)$ for all $0 \leq \alpha \leq 1$, the optimal revenue is $F(\mathbf{w}^\alpha) = F_{k+1}(\mathbf{b}') + \sum_{j=k}^n c_j\alpha (w_j^1 - w_j^0)$, and thus, \mathbf{w}^α is the optimal and binary solution of $F_k(\mathbf{b}')$. We show this by contradiction. Suppose, there exists $\hat{\mathbf{w}} = (\alpha, \hat{w}_{k+1}, \hat{w}_{k+2}, \dots, \hat{w}_n)$ such that $F(\hat{\mathbf{w}}) = F(\mathbf{w}^\alpha) + \gamma$ and $\gamma > 0$. Consider a feasible solution $\hat{\mathbf{w}}^0 = \hat{\mathbf{w}} - (\mathbf{w}^\alpha - \mathbf{w}^0)$ of $RMKP_k(0)$ (see Figure 1). Because the considered problem is linear, we have

$$F(\hat{\mathbf{w}}^0) = F(\hat{\mathbf{w}}) - (F(\mathbf{w}^\alpha) - F(\mathbf{w}^0)) \quad (9)$$

and hence we have $F(\hat{\mathbf{w}}^0) = F(\mathbf{w}^0) + \gamma$. This is a contradiction, since we have assumed that $F(\mathbf{w}^0)$ is the optimal solution among the all possible values of $w_{k+1}^0, w_{k+2}^0, \dots, w_n^0$. Hence the values of $F(\mathbf{w}^\alpha)$, $0 < \alpha < 1$ lie on the line segment joining points $F(\mathbf{w}^0)$ and $F(\mathbf{w}^1)$. Due to optimality of all $F(\mathbf{w}^\alpha)$, the solution set always contains either \mathbf{w}^1 ($F_k(\mathbf{E}) = c_k + F_{k+1}(\mathbf{E} + \mathbf{A}_k)$) or \mathbf{w}^0 ($F_k(\mathbf{E}) = F_{k+1}(\mathbf{E})$). ■

Remark 1: An intuitive reasoning is to consider each constraint $\sum_{j=1}^n a_{ij} w_j \leq b_i$ for $i = 1, \dots, m$ as a hyperplane being placed in the n dimensional space. Since all $a_{ij} \in \{0, 1\}$, these hyperplanes intersect with the axes and with each other at points of type $\mathbf{w} = (w_j, j = 1, \dots, n)$ only, where $w_j \in \{0, 1\}$. Hence all the extreme points (points of intersection of planes) have binary values. And for the same reason, we can effectively find solutions using simplex approach which checks the extreme points only.

C. Solving the master dual: s_D

Observe that the Problem s_D is an unconstrained optimization problem with variables \mathbf{u} and \mathbf{v} . The function to be minimized is nonsmooth, we use subgradient approach to solve the dual Problem s_D . This method iterates on the dual variables \mathbf{u} and \mathbf{v} . Thus given the value of \mathbf{u} and \mathbf{v} , once the solutions to the subproblems $g_x(\mathbf{u}, \mathbf{v})$ and $g_w(\mathbf{u}, \mathbf{v})$ are obtained, a dual subgradient, $\boldsymbol{\pi}_{(u)} = (\pi_{km}^{s(u)})$ and $\boldsymbol{\pi}_{(v)} = (\pi_{km}^{s(v)})$, for $g(\cdot)$ can be computed using subgradient methods.

$$\begin{aligned} \pi_{km}^{s(u)} &= (x_{km}^s d_{km}^s - \varepsilon w_{km}^s) & m \in \mathcal{P}_k^s, s \in \mathcal{S}_k, k \in \mathcal{K} \\ \pi_{km}^{s(v)} &= (w_{km}^s - x_{km}^s) & m \in \mathcal{P}_k^s, s \in \mathcal{S}_k, k \in \mathcal{K}. \end{aligned} \quad (10)$$

Then dual multipliers, \mathbf{u} and \mathbf{v} are updated using

$$u_{km}^s = \max\{0, u_{km}^s - \lambda_u \pi_{km}^{s(u)}\}, v_{km}^s = \max\{0, v_{km}^s - \lambda_v \pi_{km}^{s(v)}\}.$$

We have used the subgradient method with relaxation to minimize the non-smooth dual function. Hence the sizes, λ_u and λ_v , are given by

$$\lambda_u = \rho \frac{g(\mathbf{u}, \mathbf{v}) - g^\#}{\|\boldsymbol{\pi}_{(u)}\|^2}, \quad \lambda_v = \rho \frac{g(\mathbf{u}, \mathbf{v}) - g^\#}{\|\boldsymbol{\pi}_{(v)}\|^2}$$

where, $g^\#$ is the value of the best primal feasible solution obtained so far. We take $\rho = 2.0$ and half it, if the solution value does not change for consecutive 40 iterations. We put the maximum iteration bound as 1000 and if reached accept the maximum revenue solution among the already encountered ones as the optimal solution.

We estimate the proximity to the optimal point based on the value of λ_u and λ_v . As the value of $g^\#$ and $g(\mathbf{u}, \mathbf{v})$ get closer, the step size gets smaller and smaller. In order to evaluate the current convergence state of the algorithm, we define

$$\epsilon_i = \sqrt{\lambda_u^2 + \lambda_v^2}. \quad (11)$$

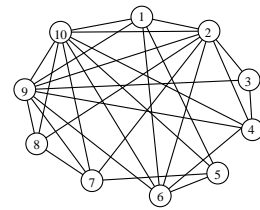
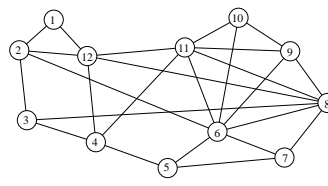
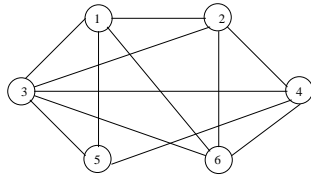
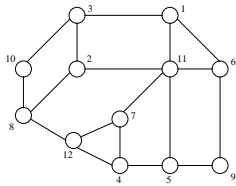
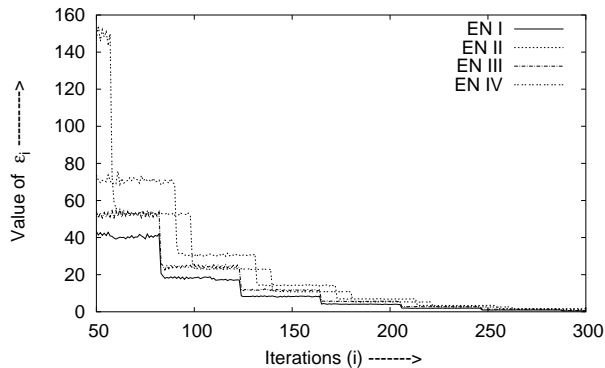
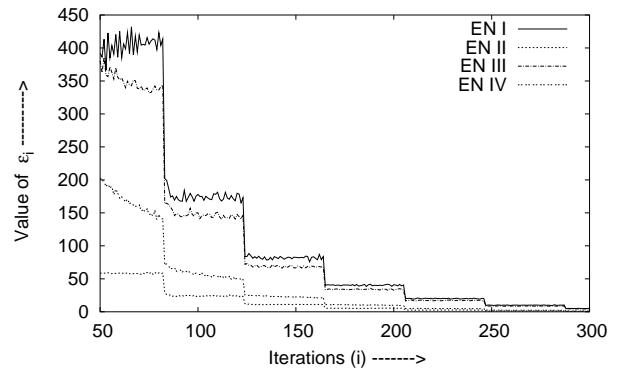


Fig. 2. Experimental Network I

Fig. 3. Experimental Network II

Fig. 4. Experimental Network III

Fig. 5. Experimental Network IV

Fig. 6. Convergence results with $T_\ell = 20$ and $C_\ell = 622$ Fig. 7. Convergence results with $T_\ell = 20$ and $C_\ell = 1866$

We assume that the iterations(i) have converged as soon as $\epsilon_i \leq \epsilon$, for chosen value of ϵ as error margin. Regarding the generation of a feasible solution, it is known that the final solution derived from the algorithm presented above might not be feasible (some of the constraints might be violated) and hence feasible solution needs to be derived at each step of the iteration from the given values of x and w . The approach we use is to assume the values of w (the binary variable) as given and solve the complete Problem (P), which is a linear program, to get feasible values of x and compute the value of the objective function f . During the course of the algorithm, we store the solution which has the maximum value of objective function f and it is the solution \bar{F} of the Formulation (P). Corresponding to the solution, we also store the values of x and w which are used in further analyzing the solution.

V. RESULTS AND DISCUSSION

The aim of this section is two-folds: (a) to study the convergence behavior of the decomposition algorithm, (b) to show the interaction between tunneling, capacity and diversity constraints, At this end, we have implemented the decomposition algorithm in C^{++} , where, we solve the subproblems using CPLEX callable libraries [6].

We conduct the study for experimental networks shown in Figures 2– 5. These networks are taken from already published literature [14]. For these experimental networks we provide detailed results and help user derive insights into the behavior. EN I has 12 nodes, 18 edges 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. Observe that EN IV is the most well connected network where as EN I is the least.

For the given experimental networks, we consider capacity of 622 Mbps for each link (referred to as baseline capacity). We assume demand between all pairs of nodes. We also assume that each demand has three service classes, where each class has a volume of 100 Mbps. We define the value of diversity constraint $h_k^s = h^s h^*$ for $k \in \mathcal{K}$, where h^s is the relative diversity requirement of service class s and h^* is the normalizing factor. We assume that service class 1 ($s = 1$) has highest survivability requirement and hence assume $h^1 = 0.4$, service class 2 has less stringent requirement and hence, we assume $h^2 = 0.6$ and for service class 3, we assume $h^3 = 0.8$. Since, we assumed that service class 1 has highest survivability requirement, we also assumed that the revenue is higher for that service class and accordingly for other service classes as well. Hence we assumed $\zeta_k^1 = 5$, $\zeta_k^2 = 3$ and $\zeta_k^3 = 2$ for $k \in \mathcal{K}$.

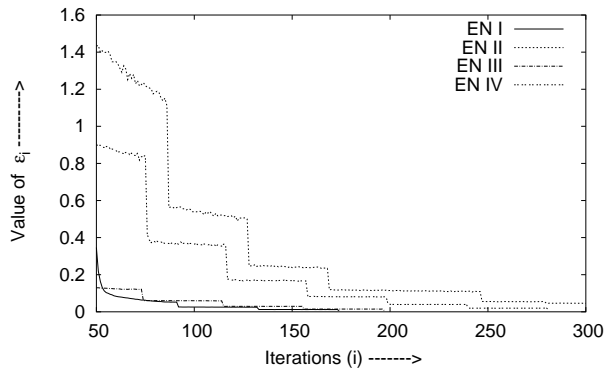
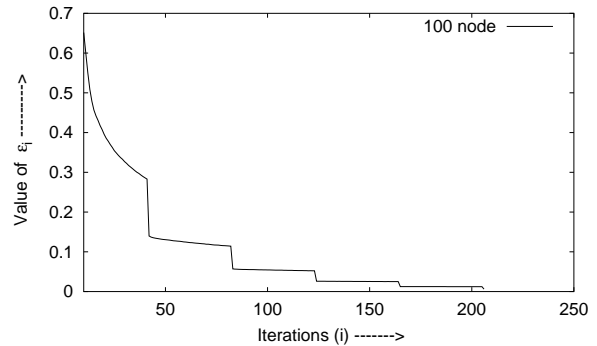
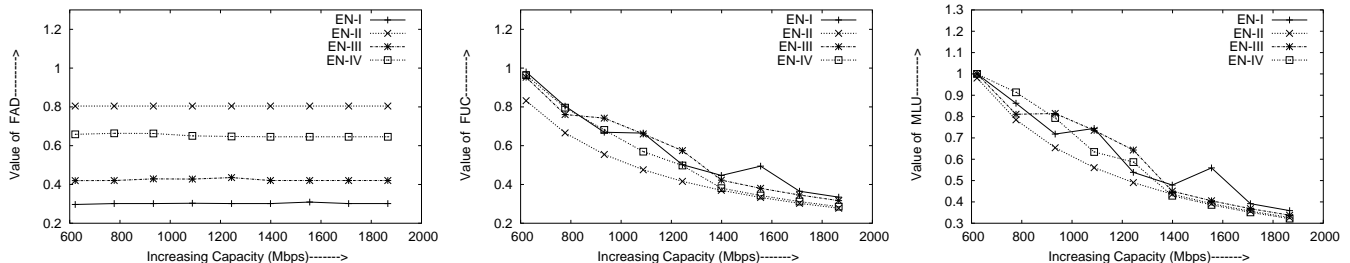
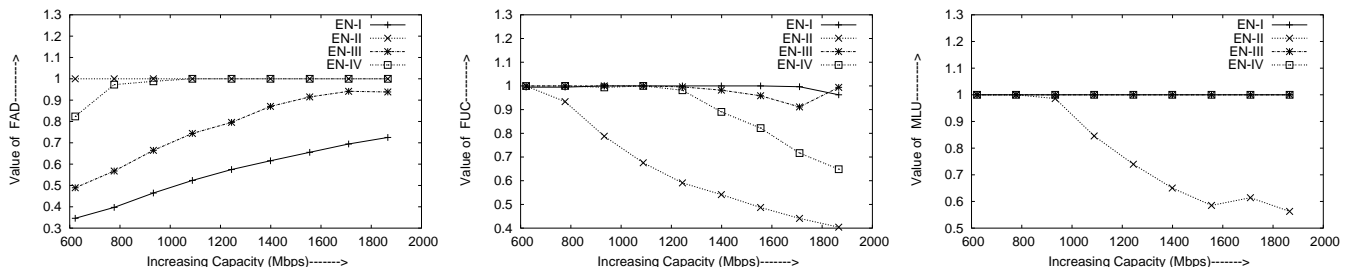
Fig. 8. Convergence results with $T_\ell = 40$ and $C_\ell = 622$ 

Fig. 9. Convergence results for 100 node network

Fig. 10. FAD, FUC and UML for $h^* = 0.5$, $T_\ell = 20$ with increasing capacity for ENsFig. 11. FAD, FUC and UML for $h^* = 1.0$, $T_\ell = 40$ with increasing capacity for ENs

A. Convergence Behavior

We start with the first aim where we establish the convergence based on the value of ϵ_i , where i is the iteration count. Notice that ϵ captures the resultant step-size based on which duals are updated at each iteration. As the algorithm progresses, the value of ϵ decreases and the algorithm stops when $\epsilon_i < 0.01$. For the study, we assumed that the normalizing factor $h^* = 0.5$.

In order to fully understand the convergence behavior, we study the convergence under three different scenarios. In the first scenario, we ensure that both the tunneling and the capacity constraints are stringent. Hence we assume that $T_\ell = 20$ and each link has capacity of 622 Mbps. We present the value of ϵ_i for increasing i for the experimental networks in Figure 6. In the next scenario, we ensure that the tunneling constraint is stringent where as network is over-provisioned in terms of capacity. Hence we take $T_\ell = 20$ and make capacity of each link equal to three times its present value. We present convergence results in Figure 7. Further, we evaluate the counter scenario where $T_\ell = 40$ (over-provisioned in number of tunnels) and capacity is still 622 Mbps (stringent in capacity) and present convergence results in Figure 8.

Similar results were observed for other scenarios as well. Note that for all the scenarios the value of ϵ_i decreases in steps. This can be attributed to the parameter ρ , which is halved if the solution does not improve for 40 consecutive iterations. Hence the value of ϵ_i will decrease due to smaller value of ρ . Moreover, we also observed that the for tunnel constrained scenarios, the primal feasible solution ($g^\#$) is much smaller than the relaxed primal solution ($g_x(\mathbf{u}, \mathbf{v}) + g_w(\mathbf{u}, \mathbf{v})$). This can be explained by observing that the relaxation is taken around the tunneling constraint. Hence when the tunnels are fewer in number, the primal feasible solution has to reject a lot of flows due to non-availability of tunnels. However, for the scenarios which are over-provisioned in number of available tunnels, the tunnels constraint is nearly obviated and the capacity constraint determines the flow allocation. In this case the feasible primal solution closely follows the relaxed primal solution.

Note that we have also solved a 100-node network (the last one in Table III-C) which took less than 300 seconds on a shared machine; its convergence behavior is presented in Figure 9. Thus, the convergence property of the algorithm is found

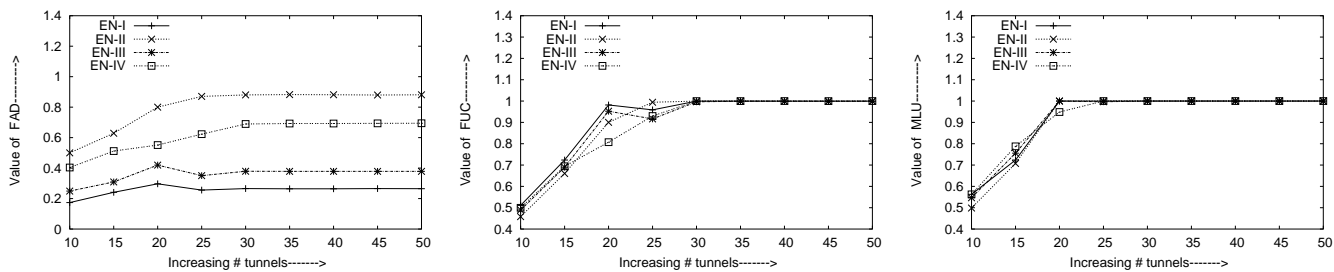


Fig. 12. FAD, FUC and UML for ENs with $h^* = 0.5$ and baseline capacity for increasing # of tunnels

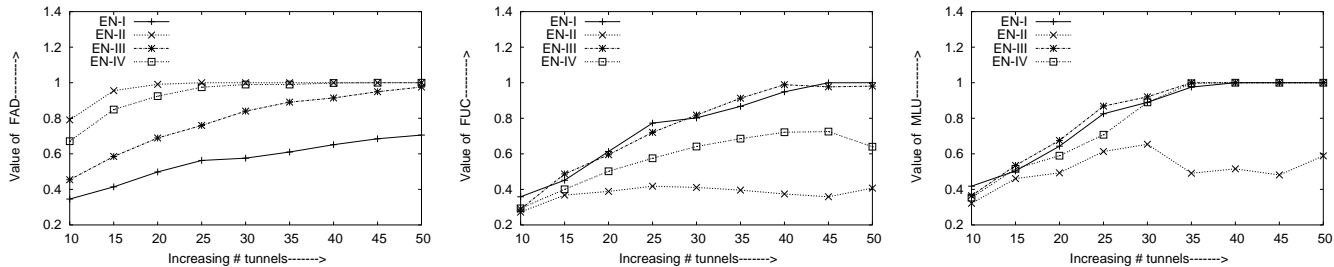


Fig. 13. FAD, FUC and UML for ENs with $h^* = 1.0$ and 300% of baseline capacity for increasing # of tunnels

to be good.

In the following sections, we use the decomposition algorithm to study the formulation to understand the interplay between tunneling, capacity and diversity constraints, as well as tunnel cost. The intention is to understand the performance of network under various provisioning scenarios and the role of various parameters towards the solution of the formulation. Thus, only the experimental networks (EN-I, EN-II, EN-III, EN-IV) are used in the rest of the study.

B. Impact of Capacity Constraint

In order to study the role of capacity constraint, we conducted experiments where we varied the values of tunneling and diversity constraints while increasing capacity from base line value (100%) to three times the baseline capacity (300%). We present the value of fraction of accepted demands (FAD), the fraction of used capacity (FUC) and the utilization of maximum loaded link (UML) for $T_\ell = 20$ and $h^* = 0.5$ in Figure 10 and $T_\ell = 40$ and $h^* = 1.0$ in Figure 11, respectively.

We observed that the presence of fewer number of tunnels prevents the demands from utilizing the excess capacity in the network. The effect gets amplified when coupled with smaller values of h_k^s which prevents a tunnel from accepting more flow, even if the links have unused capacity and therefore demands get refused. Furthermore, higher survivability requirement translates into demands requiring more and more tunnels, thus leading to increased requirement of tunnels in the network. Therefore, minor relaxations in the survivability requirement of demands translates into much fewer values of required tunnels to achieve similar performance.

C. Impact of Tunnel constraint

Next, we study the role of tunneling constraint and the way in which the formulation responds to increasing number of tunnels for various provisioning scenarios. Here, we present results showing the changes in values of FAD, FUC and MLU where available number of tunnels are increased from 10 to 50. We consider $h^* = 0.5$ and baseline capacity in Figure 12 and $h^* = 0.5$ and 300% of the baseline capacity in Figure 13. The intention is to isolate the regions where tunneling constraint is critical to achieve higher utilization of the network resources.

We observed that for network operating under the state of overload in terms of capacity, fairly small number of tunnels are required to sustain the network in its high load state. These small numbers are further lowered by decreasing the survivability level of the demands supported by the network. Interestingly, we found that the networks having higher amounts of capacity and higher levels of survivability are much more sensitive to number of allowed tunnels in the network. The impact is even more prominent for sparsely-connected, large-sized networks like EN-I and EN-III.

D. Impact of Diversity Constraint

In this section, we study the role of diversity constraint towards the utilization of available capacity in the network. Hence we present the value of FAD, FUC and MLU for values of h^* from 0.15 to 0.95 with $T_\ell = 20$ and baseline capacity in Figure 14 and $T_\ell = 40$ and 300% of the baseline capacity in Figure 15. Note that changing the value of h^* does not alter

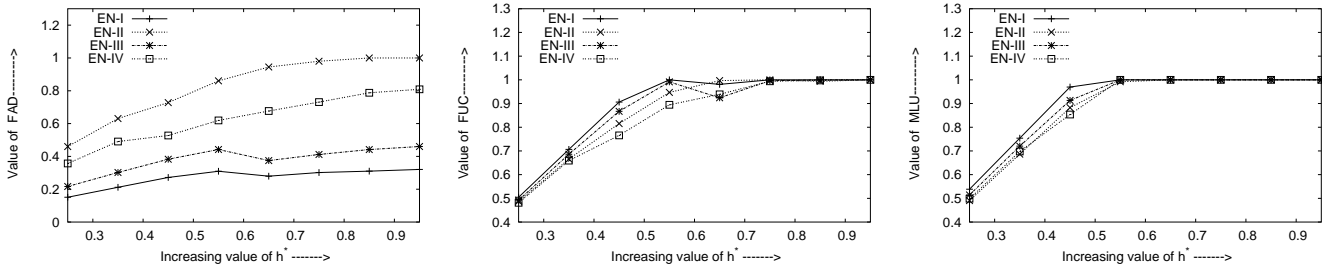


Fig. 14. FAD, FUC and UML for ENs with $T_\ell = 20$ and baseline capacity with increasing h^*

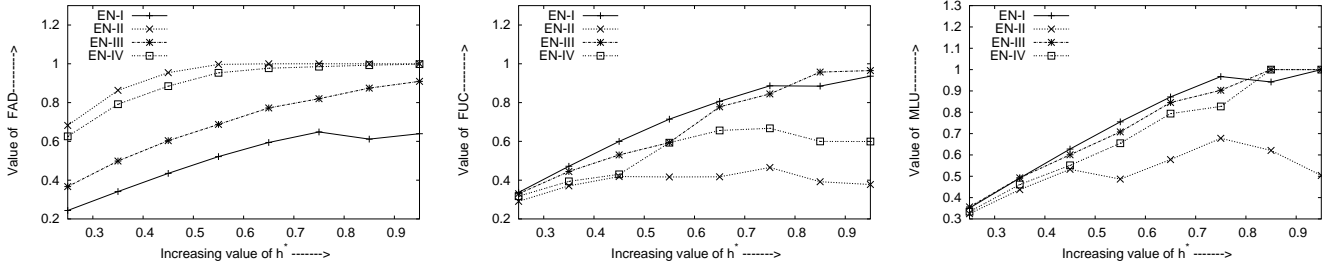


Fig. 15. FAD, FUC and UML for ENs with $T_\ell = 40$ and 300% of baseline capacity with increasing h^*

the relative survivability requirement amongst the service classes. When we decrease the value of h^* , we scale up the over all survivability requirement of all the service classes in the network.

Interestingly, a reasonably high h^* (0.65) suffices to ensure high levels of utilization of network capacity. In other cases with fewer tunnels, even for much lower levels of survivability ($h^* = 0.95$), the available capacity is not utilized. That is to say that the demands are refused (value of FAD is less than one) although un-utilized capacity is available in the network (FUC and MLU are less than one). Hence such small amounts of tunnels are extremely unacceptable for such well-provisioned (in terms of capacity) networks. More so, in order to provide higher levels of survivability in a large, sparsely-connected network, a sacrifice has to be made in terms of fraction of accepted demands. That is, in extreme cases, it might require rejection of demands in order to ensure higher survivability standards to the accepted demands.

E. Impact of Tunnel Cost

Note that we had assumed $\zeta_k^1 = 3$, $\zeta_k^2 = 2$ and $\zeta_k^1 = 1$ for $k \in \mathcal{K}$. In order to study the impact of weight factor θ_k^s on the allocation of flows and tunnels, we vary the value of $\theta_k^s = \theta$ from 0.1 to 0.9. We present the values of FAD, FUC and MLU with $T_\ell = 20$ and baseline capacity in Figure 16 and $T_\ell = 40$ and 300% of the baseline capacity in Figure 17.

Note that as the value of θ increases, longer (multi-hop) paths are no more profitable and hence solution moves towards allocating more and more flows to min-hop paths. Such a behavior translates directly into much lower values of FUC and MLU. More so, the value of θ and ζ_k^s determine the set of acceptable paths for a service class of a demand. For given ζ_k^s and $\theta = 0.5$, service class 3 ($s = 3$) tunnels with fewer than four hops would still lead to positive revenue, where as for service class 2 tunnels with less than six hops are still acceptable and for service class 1 tunnels with 10 hops can be used for allocating flows. Hence the diversity constraint and the tunnel cost have a balancing impact. Smaller value of the diversity constraint forces a demand volume to take multiple paths (which are also longer path), but such long paths are no longer profitable (for a given θ) and hence formulation chooses to reject the extra demand volume in spite of the presence of capacity and tunnels on the links.

VI. CONCLUSION

In this paper, we present a diversity based protection approach for low cost, best-effort services in a tunnel based network to avoid any total loss of connection due to node/link failures. Introduction of such a diversity based approach could lead to over whelming number of tunnels in the network. Hence we also introduced tunneling constraint by restricting the number of active tunnels on a link. Diversity reservation is embedded by restricting the fraction of allocated demand on a tunnel. We also extend the model to incorporate a hop and flow based tunnel cost. The integrated model is a mixed integer linear program. We also present a Lagrangian relaxation based decomposition approach to solve the problem for large networks. The approach requires solving multiple programs of smaller size and simultaneously updating the dual so as to arrive at the final solution. Using the solution approach, we present results demonstrating the interplay between tunneling, capacity and diversity constraints. The results showed that the tunnel constraint has similar effect as that of the capacity. Particularly, for large-sized, sparsely-connected networks, tunneling and diversity constraints affect the solution more drastically. More so, for networks having stringent survivability requirements, presence of higher number of tunnels on each link becomes crucial.

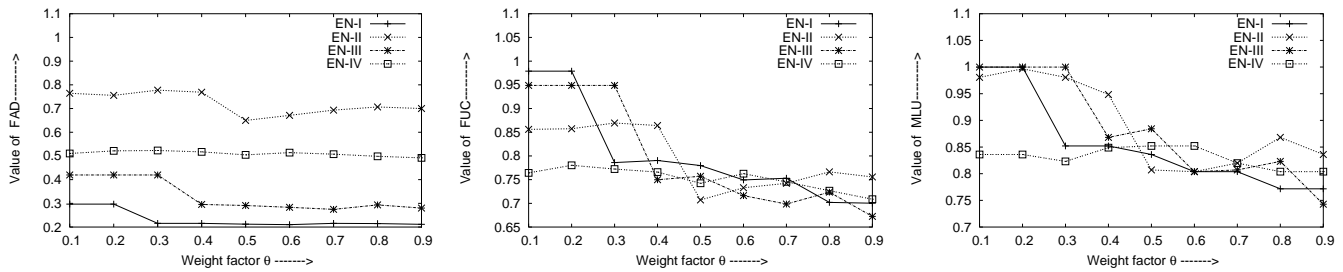


Fig. 16. FAD, FUC and UML for $T_\ell = 20$ and baseline capacity with increasing θ

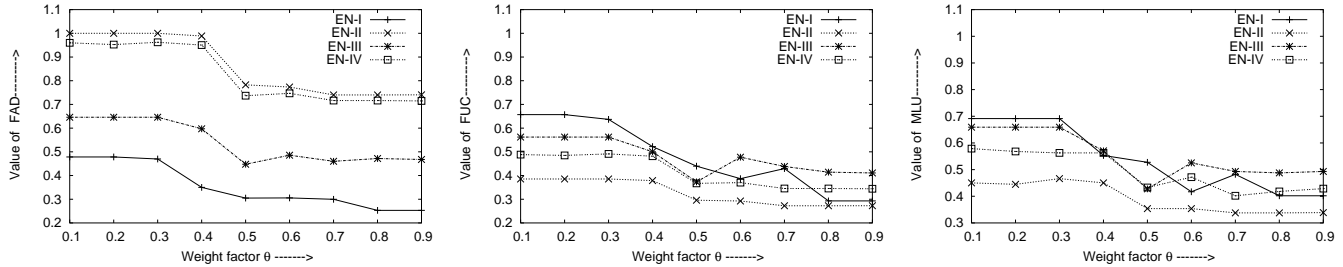


Fig. 17. FAD, FUC and UML for $T_\ell = 40$ and 300% of baseline capacity with increasing θ

ACKNOWLEDGEMENT

We thank Michał Pióro and Balaji Krithikaivasan for their contribution to the earlier conference paper [13] where the initial version of the basic model was presented.

REFERENCES

- [1] D. Applegate and M. Thorup. Load Optimal MPLS routing with N+M Labels. In *Proceedings of INFOCOM*, San Francisco, 2003. IEEE.
- [2] L. Bahiense, F. Barahona, and O. Porto. Solving Steiner Tree Problem in Graphs with Lagrangian Relaxation. *Journal of Combinatorial Optimization*, 7:259–282, 2003.
- [3] B. Gavish, P. Trudeau, M. Dror, M. Gendreau, and L. Mason. Fiberoptic Circuit Network Design under Reliability Constraints. *IEEE Journal on Selected Areas of Communication*, 7(8):1181–1187, 1989.
- [4] M. Held, P. Wolfe, and H. Crowder. Validation of Sub-Gradient Optimization. *Mathematical Programming*, pages 62–88, 1974.
- [5] K. Holmberg and J. Hellstrand. Solving the uncappeditated network design problem by a Lagrangian heuristic and branch-and-bound. *Operations Research*, 46:247–259, 1998.
- [6] ILOG Corporation, USA. *Manual of Cplex Callable Libraries*.
- [7] M. Kodialam and T. V. Lakshman. Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels Using Aggregate Link Information. In *Proceedings of INFOCOM*, pages 376–385, 2001.
- [8] D. Medhi. Diverse Routing for Survivability in a fiber-based Sparse Network. In *Proceedings of International Conference on Communications (ICC'91)*, pages 672–676. IEEE, June 1991.
- [9] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic Matrix Estimation: Existing Techniques and New Directions. In *Proceeding of SIGCOMM*. ACM, 2002.
- [10] M. Minoux. *Mathematical Programming - Theory and Algorithms*. J. Wiley & Sons, 1986.
- [11] D. Mitra and K. Ramakrishnan. A case study of Multiservice, Multipriority Traffic Engineering Design of Data Networks. In *Proceedings of GLOBECOM*. IEEE, 1999.
- [12] M. Pióro and D. Medhi. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. Morgan Kaufmann Publishers, 2004.
- [13] S. Srivastava, B. Krithikaivasan, D. Medhi, and M. Pióro. Traffic Engineering in the Presence of Tunneling and Diversity Constraints: Formulation and Lagrangean Decomposition Approach. In *Proceedings of 18th International Teletraffic Congress*, pages 461–470, Berlin, 2003. Elsevier Science.
- [14] S. Srivastava, S. R. Thirumalasetty, and D. Medhi. Network Traffic Engineering with varied levels of Protection in Next Generation Internet. In A. Girard, B. Sansó, and F. Vázquez-Abad, editors, *Performance Evaluation and Planning Methods for the Next Generation Internet: GERAD 25th Anniversary GERAD Volume*. Springer, 2005.
- [15] D. Stamatelakis and W. Grover. Theoretical underpinnings for the efficiency of restorable networks using preconfigured cycles (p-cycles). *IEEE Transactions on Communications*, 48(4), 2000.
- [16] T. H. Wu. *Fiber Network Service Survivability*. Artech House, 1992.
- [17] X. Xiao, A. Hannan, B. Bailey, and L. Ni. Traffic Engineering with MPLS in the Internet. *IEEE Network*, pages 28–33, March 2000.