Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Computer Networks 44 (2004) 319–333

COMPUTER
NETWORKS

www.elsevier.com/locate/comnet

Traffic engineering for MPLS-based virtual private networks

Chun Tung Chou *

School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, Northfields Avenue, Wollongong, NSW 2522, Australia

Received 14 November 2002; received in revised form 21 August 2003; accepted 2 October 2003

Responsible Editor: J. Roberts

Abstract

This paper considers the traffic engineering of MPLS-based Virtual Private Networks (VPNs) with multiple classes of service in an offline and centralised setting. We focus on two main issues. Firstly, we point out that the one label switched path (LSP) per ingress–egress pair constraint can be relaxed for the case of MPLS-based VPNs due to the ease in classifying flows on a per-VPN basis. This allows us to use LSP with finer granularity and thus better load balancing. Secondly, we point out that the single objective traffic engineering formulations proposed in literature address only one particular aspect of the traffic engineering problem. In this paper, we propose a multiobjective traffic engineering problem which takes resource usage, link utilisation and number of LSPs into account. The proposed optimisation problem is NP-complete and involves a large number of variables. We propose an heuristic to solve this problem.
© 2003 Elsevier B.V. All rights reserved.

Keywords: Traffic engineering; MPLS; Virtual private networks; Quality of Service (QoS)

1. Introduction

MPLS-based traffic engineering has been proposed as a solution to overcome the congestion problem of IP networks. The key idea is to use the path pinning capability of MPLS to direct the traffic flows so as to avoid network congestion and hot spots. This paper considers the traffic engineering of MPLS-based Virtual Private Networks (VPNs) with multiple classes of service in a centralised setting.

There are numerous design choices in designing an MPLS-based traffic engineering scheme, see [1] for a comprehensive discussion. For example, a traffic engineering scheme can be centralised or distributed. In a centralised scheme, global optimisation is used to compute the routes used by the label switched paths (LSPs) based on the traffic demand; these LSPs are then implemented in the network to enable better load balancing, see [21,22] for example. A distributed traffic engineering scheme can be implemented in a number of different ways. For example, in the scheme described in [5], each ingress–egress node pair can send traffic over a number of different paths and a distributed optimisation algorithm is used to

* Corresponding author. Tel.: +61-2-42214650; fax: +61-2-4227-3277.
E-mail address: ctchou@titr.uow.edu.au (C.T. Chou).

1389-1286/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2003.10.001

determine the proportion of traffic to be sent over each possible paths.

In this paper, we propose a traffic engineering algorithm for VPN in a centralised setting. In particular, we will consider the effect of two design choices, namely the granularity of LSP and the performance objectives, in the traffic engineering of MPLS-based VPNs.

In terms of LSP granularity, we can classify the various MPLS-based traffic engineering schemes into two categories. The first category consists of all those schemes [9,11,12,21,22] which assume that the traffic from an ingress–egress pair is to be put on one LSP. The rationale behind this assumption is that it will take considerable amount of work to classify the packets at the edge routers so that a flow is not routed over multiple paths. In contrast, the schemes in the second category [2,5,15] allows the traffic from an ingress–egress pair to be transported over multiple LSPs. This finer granularity has the potential to improve network resource usage balance but a classification scheme (typically based on hashing) will have to be added to the edge routers.

The work that we mention in the last paragraph assume that all the traffic flows are belonging to one single network service provider (NSP). However, for the case where multiple VPNs are sharing a single network, we argue in this paper that it is possible to achieve fine grain load balancing without incurring large overhead in packet classification. Furthermore, the implementation only requires minor modification in the edge routers of the BGP/MPLS VPN [18] scheme and does not require hashing.

Another aspect of traffic engineering that we will consider in this paper is the choice of optimisation criterion for traffic engineering. Two common optimisation criteria have been proposed in the literature. The first one is to minimise a linear function of the link bandwidth usage [9]. From a NSP's point of view, this optimisation criterion minimises the network operation cost. However, a drawback of this criterion is that it may result in an uneven distribution of traffic in the network where some links are over-utilised and some links are under-utilised. This is demonstrated in the example in Section 4 of this paper.

Another optimisation criterion that has been proposed in literature is to minimise the maximum link utilisation [21]. This optimisation criterion will produce an even traffic distribution and will also maximise the room for traffic growth. However, a drawback of this criterion is that the network resource usage is not minimised. In fact, the example in Section 4 shows that this criterion may use 70% more network resources than the case where network resources are minimised.

In this paper, we propose a multiobjective formulation of the traffic engineering problem which takes into account resource usage (which can be viewed as network operating cost), link utilisation and the number of LSPs. The proposed multiobjective optimisation problem is NP-complete and has a large number of binary integer variables. A contribution of this paper is that we propose an efficient heuristic to solve this problem.

1.1. Related work

Previous work on mixed-integer programming (MIP) approach to MPLS-based traffic engineering [9,15,21,22] are focused on the case where all traffic belongs to one NSP. The problem of VPN traffic engineering had previously been considered in [16] based on Poisson traffic model with 3the goal of maximising a revenue objective that depends on call blocking probability. However, our work is based on a deterministic setting which takes load balancing, resource usage and number of LSPs into consideration.

The authors of [15] also uses a multiobjective framework based on utilisation and resource usage. However, our work has been developed independently of theirs. Moreover, the emphases of [15] are substantially different from ours. Firstly, it does not consider traffic engineering in the VPN setting which results in different optimisation problem formulations. Secondly, it solves the MIP problem using standard optimisation software but we develop heuristics for our problems. Due to the NP-completeness of the problem, the use of standard optimisation software limits the size of the problem that can be solved in practice.

A few other traffic engineering optimisation criteria can also be found in literature. For example, [6] uses a 6-segment piecewise linear monotonically increasing function of link utilisation which imitates the response time of an M/M/1 queue; [2] proposes a 2-segment piecewise linear criterion which penalises links with an utilisation above a pre-determined level. By using a piecewise linear function, these two objective functions take load balancing and resource usage into account simultaneously. Due to the use of piecewise linear objective function, the size of the optimisation problems grows in proportion to the number of segments being used. We take an alternative approach here where load balancing and resource usage are considered respectively in two consecutive linear programming problems. Moreover, the heuristic proposed in Section 3.2 allows the tradeoff between maximum link utilisation (i.e. congestion level) and the number of LSPs (i.e. management complexity). This feature is not found in other existing traffic engineering solutions.

1.2. Paper organisation

The rest of this paper is organised as follows. Section 2 discusses the MPLS-based VPN traffic engineering problem. This section discusses the issues of LSP granularity and optimisation criterion, and ends with a mixed integer multiobjective programming formulation of the VPN traffic engineering problem. The proposed optimisation problem is NP-complete and involves a large number of binary decision variables. In Section 3, we propose a heuristic solution to tackle this problem. Finally, an example is given in Section 4 and the conclusions are presented in Section 5.

2. Traffic engineering of MPLS-based VPN

In this section, we will formulate the traffic engineering problem for MPLS-based VPN with multiple classes of service. Section 2.1 gives an overview of the VPN traffic engineering problem. Section 2.2 addresses two issues: the granularity of the LSP and the optimisation criterion. Based on the discussion in Section 2.2, we present a multi-objective formulation of the VPN traffic engineering problem in Section 2.3.

2.1. Overview of the traffic engineering problem for MPLS-based VPN

According to [1], Internet traffic engineering is concerned with performance optimisation of operational IP networks. A common problem that is faced in today's Internet, which is caused by the use of destination based shortest path routing, is that part of the network is over-utilised while another part of the network is under-utilised. A goal of traffic engineering is to correct this imbalance in resource usage. This can be achieved by using the route pinning property of MPLS which allows the NSP to control the routes used by the different LSPs. The route pinning property of MPLS is also important in providing QoS in the Internet. A fundamental requirement of being able to provide QoS guarantee is to ensure that there are sufficient resources for the QoS traffic. By using MPLS and resource reservation, a NSP can ensure that QoS traffic is given sufficient network resources.

Since the main degree-of-freedom in MPLS traffic engineering is the choice of routes for the LSPs, a traffic engineering problem is often formulated as an integer or mixed-integer optimisation problem whose aim is to find a suitable route for each of the LSPs [9,21]. In the VPN traffic engineering problem to be considered in this paper, we assume this computation is performed in an off-line centralised manner. Furthermore, we assume that the NSP owns a physical network for providing the VPN service. In order to simplify the discussion here, we assume for the time being that only one service class is offered by this NSP. We also assume that each VPN customer provides the NSP with a traffic demand matrix whose elements are the bandwidth requirement between an ingress–egress pair of the VPN. In this context, the goal of the VPN traffic engineering problem is to find a route for each of these demands. However, this description has overlooked two important issues:

1. The granularity of the LSPs to be used to implement the VPNs in the physical network.
2. The optimisation criterion to be used.

We will discuss these two issues further in the next section.

2.2. Traffic engineering issues

2.2.1. Granularity of LSPs

In the traffic engineering overview in the previous section, we mention that the goal of the VPN traffic engineering is to find a suitable path for each demand of each VPN. (We continue to assume a single service class in this section). An issue is how these demands should be mapped to the LSPs. On one extreme, we can aggregate all the demands using the same ingress–egress pair from all VPNs into an LSP. For a physical network with N nodes, this will result in $\mathcal{O}(N^2)$ LSPs in the network.

The idea of using only one LSP per ingress–egress pair is implicit in the implementation of the BGP/MPLS VPN scheme presented in the IETF RFC 2547 [18]. The implementation makes use of MPLS label stack where the bottom label is VPN specific while the top label is VPN independent. The core routers in the network only require the top label for routing and are therefore completely oblivious of the existence of the various VPNs. This results in a scalable implementation where the number of routes in the network can be made independent of the number of VPNs.

However, sometimes it may not be possible to put the aggregate of all the demands of an ingress–egress pair in one LSP. This happens if the aggregate demand is larger than the capacity of any single link in the network. Also, an aggregate with a large demand may be hard to load balance.

The mapping of the aggregate demand between an ingress–egress pair onto a single LSP represents the coarsest granularity that we can use. On the other extreme, each of the demands of each VPN can be mapped onto an individual LSP. This will result in $\mathcal{O}(N^2 \times \#\text{VPNs})$ LSPs or routes in the network. This is clearly a non-scalable solution and is precisely what the authors of RFC 2547 [18] are trying to avoid. However, we see in the last paragraph that there are occasions where it is appropriate to use more than one LSP for the aggregate demand between an ingress–egress pair. We therefore believe that the granularity of a LSP should not be fixed a priori but should be determined by the optimisation process. However, a limit on the number of LSPs should be imposed in order to avoid an unscalable number of routes.

Note that it requires only minor modification to the edge routers in the BGP/MPLS scheme in order to have multiple LSPs between an ingress–egress pair. For example, if we are to set up two LSPs between an ingress–egress pair, we can divide the VPNs using this ingress–egress pair into two groups where traffic from each group will be assigned to one particular LSP. The edge router will again insert two labels into the packets. The bottom label is VPN specific while the top label will specify which one of the two LSPs will be used according to the identity of the VPN. Note that even if multiple LSPs are used, only the edge routers have to know about the different VPNs but the core routers remain unaware of the existence of various VPNs.

We do not advocate the use of granularity that is finer than per-VPN level because significant workload, in the form of IP packet classification, will be required to ensure that an IP flow is not split across multiple LSPs. Thus, unlike many load balancing schemes which use multiple LSPs, packet classification here does not require the use of hashing.

2.2.2. Optimisation criterion

A goal of traffic engineering is to optimise network performance. Various optimisation criteria have been proposed for this traffic engineering optimisation problem. For example, [9] proposes a criterion which minimises a weighted linear sum of per-link bandwidth usage. However, such an optimisation criterion has the same drawback as minimising the resource usage, i.e. some links being over-utilised. This will be illustrated in an example in Section 4.

An alternative optimisation criterion suggested in the literature is to minimise the maximum link utilisation in the network [21]. The motivation for introducing such criterion is that, in the case of fixed routing and linear traffic growth, the minimisation of the maximum link utilisation will maximise the linear growth factor before re-routing will be required. However, such criterion has two drawbacks. Firstly, it ignores the resource usage as a factor. Secondly, it puts its emphasis on the bottleneck link only. In fact, the example in Section 4 shows that this criterion may use 70% more network resources than the case where network resources are minimised.

Note that both of these criteria, if used on their own, address only one aspect of the traffic engineering problem. In Section 2.3, we propose a multiobjective programming optimisation problem which uses both of these criteria. This results in a solution which takes both network resource usage and network traffic growth into account. We will demonstrate in Section 4 that this multiobjective programming formulation gives a near Pareto optimal solution in both resource minimisation and maximum link utilisation. Moreover, we will show in Section 3.2 that this multiobjective framework also provides a way to adjust the number of LSPs to be used.

2.3. Mathematical formulation of the VPN traffic engineering problem

2.3.1. Notation

This section defines the notation that will be used. We assume that the physical network is given by a capacitated directed graph $\mathscr{G} = \langle \mathscr{V}, \mathscr{E} \rangle$ where $\mathscr{V}$ and $\mathscr{E}$ are respectively the set of network nodes and links. The elements in $\mathscr{E}$ are denoted by e_{uv} where $u, v \in \mathscr{V}$ are the end points of the link. The link $e_{uv} \in \mathscr{E}$ has a capacity (bandwidth) given by b_{uv}. The cost per unit bandwidth on e_{uv} will be denoted by c_{uv}.

We assume the NSP offers a number of different service classes indexed by $s \in \mathscr{S} = \{1, 2, \ldots\}$. The total number of service classes is denoted by $|\mathscr{S}|$.

We assume there are altogether M different VPNs and they will be indexed by m. Each of these VPNs will supply the NSP with $|\mathscr{S}|$ traffic demand matrices, one for each traffic class. Let $t_{ij}^{m,s}$ be the traffic demand of the mth VPN for service class s between ingress–egress pair (i,j) where $i, j \in \mathscr{V}$. Note that each VPN may have different virtual topologies and may not have demands for all the different service classes. In this case, a zero value in the demand matrix will be used.

Let i, j be two distinct nodes in $\mathscr{V}$. For each ingress–egress pair (i,j) and service class s, each individual demand between i and j will be routed over one of the potential paths in the set $P_{ij}^{s} = \{p_{ij}^{s,1}, p_{ij}^{s,2}, \ldots, p_{ij}^{s,k}, \ldots\}$. These paths are assumed to be loop free. Note that the set of potential paths is dependent on the ingress–egress pair and the service class, and is independent of individual VPNs. Let P denote the order of magnitude of the number of potential routes per ingress–egress pair per service class. This quantity will be used later on to quantify the number of variables in the optimisation problem.

In the optimisation problems to be formulated, we will need to ensure that the total capacity allocated to any physical link does not exceed its physical capacity. We therefore require a way to keep track of whether a particular potential path uses a certain physical link. We define the following indicator functions:

$$\mathscr{I}_{ij}^{uv,s,k} = \begin{cases} 1 & \text{if } e_{uv} \in \mathscr{E} \text{ is on the path } p_{ij}^{s,k} \in P_{ij}^{s}, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Let μ_{uv} denote the link utilisation of the physical link $e_{uv} \in \mathscr{E}$.

Also let $\bar{R}$ denote a pre-specified upper limit on the total number of LSPs or routes in the NSP's physical network.

Note that in the above definitions, and in the rest of the paper, we have adopted the convention of using i and j to index the ingress and egress of a VPN demand. The end points of a physical link will be indexed by u and v.

2.3.2. A multiobjective VPN traffic engineering problem

The aim of this section is to formulate the multiobjective VPN traffic engineering problem. In this paper, we will make the assumption that the physical network $\mathscr{G}$ has sufficient capacity to meet the demands from all VPNs. With this assumption, the traffic engineering problem becomes one of choosing a suitable physical route for each VPN demand such that a certain criterion is optimised. Define the decision variables

$$\delta_{ij}^{m,s,k} = \begin{cases} 1 & \text{if demand } t_{ij}^{m,s} \text{ uses path } p_{ij}^{s,k} \in P_{ij}^{s}, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

In terms of these decision variables and the indicator function defined in Eq. (1), the capacity allocated on link $e_{uv} \in \mathscr{E}$ for the VPN requests is

$$y_{uv} = \sum_{s} \sum_{m} \sum_{i,j} t_{ij}^{m,s} \left(\sum_{k} \mathscr{I}_{ij}^{uv,s,k} \delta_{ij}^{m,s,k} \right). \tag{3}$$

In order to control the number of LSPs to be used, we introduce an additional set of decision variables

$$\eta_{ij}^{s,k} = \begin{cases} 1 & \text{if the LSP between ingress–egress} \\ & \quad \text{pair } (i,j) \text{ uses path } p_{ij}^{s,k} \in P_{ij}^{s}, \\ 0 & \text{otherwise.} \end{cases}$$

The total number of LSPs or routes R that will be used to implement these VPNs will be

$$R = \sum_{ij} \sum_{s} \sum_{k} \eta_{ij}^{s,k}. \tag{4}$$

The multiobjective programming VPN traffic engineering problem consists of two steps. In the first step, we minimise the maximum link utilisation and is stated as follows:

Optimisation problem OPT1a

$$\min \mu \tag{5}$$

subject to the constraints

$$y_{uv} = \sum_{s} \sum_{m} \sum_{i,j} t_{ij}^{m,s} \left(\sum_{k} \mathscr{I}_{ij}^{uv,s,k} \delta_{ij}^{m,s,k} \right) \leqslant \mu b_{uv} \quad \forall e_{uv} \in \mathscr{E}, \tag{6}$$

$$\sum_{k} \delta_{ij}^{m,s,k} = 1 \quad \forall i,j \in \mathscr{V}, \quad m = 1,\ldots,M,\ s = 1,\ldots,|\mathscr{S}|, \tag{7}$$

$$\delta_{ij}^{m,s,k} \leqslant \eta_{ij}^{s,k} \quad \forall i,j \in \mathscr{V}, \quad m = 1,\ldots,M,\ s = 1,\ldots,|\mathscr{S}|, \tag{8}$$

$$\sum_{ij} \sum_{s} \sum_{k} \eta_{ij}^{s,k} \leqslant \bar{R}, \tag{9}$$

$$\delta_{ij}^{\ell,s,k}, \eta_{ij}^{s,k} \in \{0,1\}. \tag{10}$$

The constraint (7) ensures that only one path is chosen for the demand $t_{ij}^{m,s}$. The constraint (8) enforces the fact that if the LSP between ingress–egress pair (i,j) does not use path $p_{ij}^{s,k}$, no demands in $t_{ij}^{m,s}$ can use this path. Finally, the inequality (9) is a constraint on the number of routes.

Let μ_{OPT1a}^{*} be the optimal value of μ obtained in the first optimisation step. The second optimisation step is to minimise the cost subject to the constraint that all link utilisation remains under μ_{OPT1a}^{*}. The problem can be stated as follows:

Optimisation problem OPT1b

$$\min \sum_{uv} c_{uv} y_{uv} \tag{11}$$

subject to the constraints

$$y_{uv} = \sum_{s} \sum_{m} \sum_{i,j} t_{ij}^{m,s} \left(\sum_{k} \mathscr{I}_{ij}^{uv,s,k} \delta_{ij}^{m,s,k} \right) \leqslant \mu_{\mathrm{OPT1a}}^{*} b_{uv} \quad \forall e_{uv} \in \mathscr{E}, \tag{12}$$

$$\sum_{k} \delta_{ij}^{m,s,k} = 1 \quad \forall i,j \in \mathscr{V}, \quad m = 1,\ldots,M,\ s = 1,\ldots,|\mathscr{S}|, \tag{13}$$

$$\delta_{ij}^{m,s,k} \leqslant \eta_{ij}^{s,k} \quad \forall i,j \in \mathscr{V}, \quad m = 1,\ldots,M,\ s = 1,\ldots,|\mathscr{S}|, \tag{14}$$

$$\sum_{ij} \sum_{s} \sum_{k} \eta_{ij}^{s,k} \leqslant \bar{R}, \tag{15}$$

$$\delta_{ij}^{\ell,s,k}, \eta_{ij}^{s,k} \in \{0,1\}. \tag{16}$$

The constraints in the second optimisation step are the same as those in the first step except for constraint (12), where we enforce the condition that the maximum utilisation of the network remains at the same level as that given by the first optimisation.

In order to understand why the second optimisation step is necessary, we need to realise that the solution to **OPT1a** is generally not unique. Without loss of generality, we will assume in the following discussion that $c_{uv} = 1$. This means the objective of **OPT1b** is to minimise the total resource usage. We now argue that there are many solutions to **OPT1a** which give the same value of μ_{OPT1a}^{*} but they consume different level of network resources. Consider the network depicted in Fig. 1(a) where all links are assumed to have the same

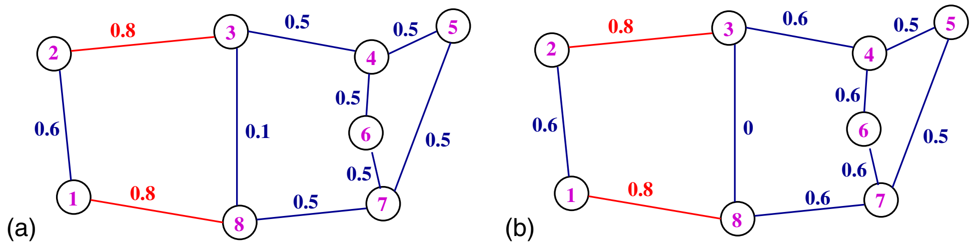

Fig. 1. This figure is used in Section 2.3.2 to explain why the solution to minimising the maximum utilisation is generally not unique.

capacity. The number next to a link in the figure indicates the utilisation of that link. The maximum link utilisation is therefore 0.8. If we re-route the traffic currently on link 3–8 to the path 3–4–6–7–8, the resulting link utilisations are given in Fig. 1(b). We see that the routing patterns in both Fig. 1(a) and (b) have the same maximum link utilisation, but that in Fig. 1(b) has a higher resource usage.

2.3.3. Choice of paths

The basic idea behind using the set of potential paths P^s_{ij} is that we will use this selection of paths to enforce the QoS specifications for each service class. For example, we may use P^s_{ij} to set a limit on the number of hops or propagation delay used by each traffic class or some other constraints. These paths will be generated before the optimisation begins. The choice of paths can have an important effect on the quality of the solution. For example, if we provide only two possible choices of paths between an ingress–egress pair and these two paths have 1 and 10 hops respectively. In case the 1-hop path is a bottleneck, the traffic between this ingress–egress pair will be forced onto the 10-hop path, which is undesirable. This problem can be avoided by providing a good selection of paths, for example, all paths within a certain hop limit or all paths whose propagation delay is below a certain margin. Providing such choices is generally not a problem in a well designed network. Our experience in using the proposed method (see Section 4) shows that most of the traffic is routed along the shortest path and all traffic is routed along a path whose hop count is within 2 hops from the shortest path. Similar conclusion also appeared in [15]. This points to the possibility of using an incrementally larger set of paths (e.g. set of paths whose hop count is within k hop from the shortest path, for $k = 1, 2, \ldots$) in optimisation in order to ensure that the choice of paths does not significantly affect the quality of the solution.

Note an optimisation problem formulated in terms of paths can be formulated equivalently in terms of links [3] (see for example [9,15,21] for traffic engineering problems formulated in terms of links), the path formulation removes constraints such as path hop count, which can be hard to handle, from the optimisation problem.

2.3.4. Complexity of the problem

The optimisation problems **OPT1a** and **OPT1b** formulated earlier belong to the class of mixed linear integer programming (MILP). A special case of the optimisation problem **OPT1a** is considered in [21] where there is only 1 VPN and 1 service class, and without constraints on the number of LSPs. That problem is proved in [21] to be NP-hard. Thus the problem **OPT1a** is also NP-hard. The second optimisation problem **OPT1b** is NP-complete [9].

In addition, these two optimisation problems also involve a large number of binary decision variables, which is of the order $\mathcal{O}(N^2 \times M \times |\mathcal{S}| \times P)$. We will provide an heuristic solution to this optimisation problem in the following section.

2.3.5. Discussion

We assume in this paper that all demands will be met. If this does not hold, the result to **OPT1a** will return a value greater than 1. The next step will be to continue with **OPT1b** to minimise the resource usage. A third optimisation problem will need to be solved in order to select a number of VPNs to be admitted. The selection can be based on choosing those VPNs which maximises the revenue by admitting them. Note that there are a number of ways of defining what a VPN constitutes. For example, we can impose the condition that a VPN will only be admitted if all of its constituent service classes can be admitted. Alternatively, we can admit the VPNs one service class at a time starting from the highest service class. Note that the VPN admission control problem differs significantly from the admission control for a single LSP as the admissibility of a VPN requires all its constituent virtual links be admitted at the same time, not a proper subset of it. We have formulated a few different VPN admission control problems and they can be shown to be NP-hard by their relation to the multidimensional knapsack problem [4]. The solution method to these problems will be the topic of a future paper.

3. An heuristic solution

In this section, we present an heuristic solution to the optimisation problems **OPT1a** and **OPT1b** that we have formulated earlier. In order to reduce the complexity of the problem, we will perform the optimisation with one service class at a time. Besides making the optimisation problem more tractable, this allows us to balance the amount of traffic of different service classes on each link. For example, a NSP may want to limit the proportion of real-time traffic on each link so as to achieve reasonable delay and jitter responses [14].

In Section 3.1, we give an heuristic solution to **OPT1a** and **OPT1b** by first ignoring the constraints on the number of routes. We will then show in Section 3.2 how the number of routes can be indirectly adjusted in our multiobjective programming framework.

3.1. Heuristic solution without constraints on the number of routes

In this section, we present an heuristic solution to the optimisation problems **OPT1a** and **OPT1b** with the following simplifications:

1. We will perform the optimisation with one service class at a time. This reduces the number of binary decision variables per optimisation problem to the order $\mathcal{O}(N^2 \times M \times P)$.
2. We ignore the constraints on the number of routes for the time being. In other words, we drop the constraints (8) and (9) for **OPT1a**, and the constraints (14) and (15) for **OPT1b**.

Note: The removal of the constraint on the number of routes means that we have lost control over this requirement. However, we will demonstrate how we can indirectly adjust the number of paths in Section 3.2.

Even with the first simplification in place, the number of binary decision variables that we have to deal with is still large. In fact, the complexity of the problem grows with the number of VPNs. We will approach this problem in two steps. We will show in Section 3.1.1 how we can obtain an approximate solution using linear programming (LP). In Section 3.1.2, we show how we can obtain an integer solution using the approximation obtained in Section 3.1.1.

3.1.1. A continuous approximation

The aim of this section is to formulate two LP problems which give us an approximation of the simplified version of **OPT1a** and **OPT1b**.

Let T_{ij}^s be the aggregate demand from all VPNs for ingress–egress pair (i,j) for service class s, i.e.

$$T_{ij}^s = \sum_{m=1}^{M} t_{ij}^{m,s}. \tag{17}$$

We further assume that $T_{ij}^s \neq 0 \ \forall s = 1,\dots,|S|$, $i,j \in \mathscr{V}$.

Since we will be performing the optimisation on a per-class basis, the index s should be treated as a constant here. We have chosen to retain the index s instead of dropping it so that we do not have to redefine the notation.

We now define a set of continuous decision variables in the range $[0,1]$. Define

$$x_{ij}^{s,k} = \text{the fraction of aggregate demand } T_{ij}^s \text{ to be routed over the path } p_{i,j}^{s,k}.$$

Based on these decision variables and the indicator function (1), the capacity being used on physical link e_{uv} can be written as

$$z^s_{uv} = \sum_{ij} T^s_{ij} \sum_k x^{s,k}_{ij} \mathscr{I}^{uv,s,k}_{ij}. \tag{18}$$

Based on the simplifications that we have introduced earlier, we define the following two LP problems.

OPT2a

$$\min \mu \tag{19}$$

subject to the constraints

$$z^s_{uv} = \sum_{ij} T^s_{ij} \sum_k x^{s,k}_{ij} \mathscr{I}^{uv,s,k}_{ij}$$
$$\leqslant \mu b_{uv} \quad \forall e_{uv} \in \mathscr{E}, \tag{20}$$
$$\sum_k x^{s,k}_{ij} = 1 \quad \forall i,j \in \mathscr{V}, \tag{21}$$
$$x^{s,k}_{ij} \in [0,1] \quad \forall i,j \in \mathscr{V},\ \forall k. \tag{22}$$

Let μ^*_{OPT2a} be the minimum value of μ given by **OPT2a**. The second LP is:

OPT2b

$$\min \sum_{uv} c_{uv} z^s_{uv} \tag{23}$$

subject to the constraints

$$z^s_{uv} = \sum_{ij} T^s_{ij} \sum_k x^{s,k}_{ij} \mathscr{I}^{uv,s,k}_{ij}$$
$$\leqslant \mu^*_{\text{OPT2a}} b_{uv} \quad \forall e_{uv} \in \mathscr{E}, \tag{24}$$
$$\sum_k x^{s,k}_{ij} = 1 \quad \forall i,j \in \mathscr{V}, \tag{25}$$
$$x^{s,k}_{ij} \in [0,1] \quad \forall i,j \in \mathscr{V},\ \forall k. \tag{26}$$

The problem **OPT2a** is in fact the continuous relaxation of the version of **OPT1a** with the constraint on the number of paths removed. μ^*_{OPT2a} is therefore a lower bound of μ^*_{OPT1a}. However, **OPT2b** is *not* the continuous relaxation of **OPT1b** because μ^*_{OPT2a} is used in **OPT2b** but not **OPT1b**. Note that both of these LPs have $\mathscr{O}(N^2 \times P)$ variables, which is independent of the number of VPNs.

3.1.2. Recovering the integer solution

We will show in this section, how we can recover the integer solution from the continuous solution to **OPT2b**. The solution to **OPT2b** tells us how the aggregate demand T^s_{ij} is split among the potential paths $\{p^{s,1}_{ij}, p^{s,2}_{ij}, \ldots, p^{s,k}_{ij}, \ldots\}$. The suggested heuristic is to distribute the VPN demands $t^{m,s}_{ij}$ for $m = 1, \ldots, M$ among the potential routes with non-zero traffic (i.e. those routes with non-zero $x^{s,k}_{ij}$ values in the solution to **OPT2b**) such that after the distribution process, the actual fraction of aggregate demand in each potential route with non-zero traffic matches as closely as possible to that given by the continuous solution. Note that this heuristic may not be able to recover the optimal integer solution as paths with zero traffic in the continuous solution may be used in the optimal solution. Based on the problem description earlier, we will define the problem in a general setting.

Let $\{t_1, \ldots, t_D\}$ be a set of non-zero demands to be distributed over B (where $B \geqslant 2$) different LSPs. The fraction of demands to be distributed over the hth LSP ($h = 1, \ldots, B$) is $\rho_h \in (0,1)$ with $\sum_{h=1}^{B} \rho_h = 1$. Furthermore, we define $T = \sum_{g=1}^{D} t_g$ and the cost of hth LSP as

$$\gamma_h = \sum_{e_{uv} \in h\text{th LSP}} c_{uv}. \tag{27}$$

Note that B and ρ_h's are given by the solution of the optimisation problem **OPT2b**.

In order to formulate this problem of sorting demands into the LSPs, we define binary decision variables

$$q_{gh} = \begin{cases} 1 & \text{if demand } t_g \text{ is to be put into bin } h, \\ 0 & \text{otherwise.} \end{cases} \tag{28}$$

This sorting problem can be stated as the following optimisation problem:

OPT3

$$\min \sum_{h=1}^{B} \gamma_h \left| T\rho_h - \sum_{g=1}^{D} t_g q_{gh} \right| \tag{29}$$

subject to the constraint

$$\sum_{h=1}^{B} q_{gh} = 1 \quad \forall g = 1, \ldots, D, \tag{30}$$

$$q_{gh} \in \{0,1\}. \tag{31}$$

Theorem 1. *The optimisation problem* **OPT3** *is NP-complete.*

Proof. See Appendix A. □

We propose an heuristic to solve this problem based on the subset sum problem [17] which is a special case of knapsack problem where the cost and weight are equal. For the subset sum problem, one is given items of weight $w_1, \ldots, w_n$ and a bin with size W. The 0–1 decision variables are $x_1, \ldots, x_n$ and the goal is to maximise $\sum w_i x_i$ such that $\sum w_i x_i \leqslant W$. The subset sum problem is known to be NP-complete [8, p. 247]. It can be solved exactly in pseudo-polynomial time by dynamic programming [17]. Recent progress on exact algorithms for the subset sum problem can be found in [19]. There are a number of heuristics for solving the subset sum problem (see [7,13,17]) with an algorithm in [13] having the best worst case relative error bound with respect to the optimal. Our computation experience with applying these algorithms to randomly generated weights and bin sizes suggests that the following greedy algorithm for the subset sum problem works well.

Algorithm Greedy subset sum

```
1. Sort $w_g$'s in descending order, i.e. $w_1 \geqslant \cdots \geqslant w_D$.
2. Initialisation: $g = 1$.
3. If $w_g \leqslant B$, then
   (a) Add $w_g$ to the bin,
   (b) $B \leftarrow B - w_g$.
   Otherwise, end.
4. $g \leftarrow g + 1$. Go to 3.
```

We propose the following heuristic to solve **OPT3**:

Algorithm Heuristic for OPT3

```
1. Sort the demands in descending order $d_1 \geqslant \cdots \geqslant d_D$.
2. Sort the available bandwidth on each LSP in ascending order $\rho_1 \leqslant \cdots \leqslant \rho_B$.
3. Let $h = 1$ and $\mathscr{D} = \{d_1, \ldots, d_D\}$.
4. Use the greedy subset sum heuristic to solve the subset sum problem with $\mathscr{D}$ as the items and $T\rho_h$ as bin size.
5. Remove demands that have been assigned from $D$.
6. If $h = B$, go to step 7, else increase $h$ by 1 and go to step 4.
7. Let $\mathscr{D}_r = \{d_1, \ldots, d_{D_r}\}$ denote the demands that are yet to be placed.
   (a) Compute $\eta_{gh}$ which is the objective function value if the remaining demand $d_g \in \mathscr{D}_r$ is placed in LSP $h$. Do this for all demands in $\mathscr{D}_r$ and $h = 1, \ldots, B$.
   (b) Place demand $d_{\hat{g}}$ in LSP $\hat{h}$ if $\eta_{\hat{g}\hat{h}}$ is minimum.
   (c) Remove $d_{\hat{g}}$ from $\mathscr{D}_r$. Goto 7 if $\mathscr{D}_r$ is non-empty, otherwise end.
```

Example. A number of Monte Carlo experiments have been carried out to test the performance of **Heuristic for OPT3**. For each experiment, demands are generated randomly using uniform distribution in [100, 2000]. The number of demands D used is either 100 or 1000, and the number of LSPs B is 2, 3 or 4. The relative proportion of traffic in each LSP ρ_h $(h = 1, \ldots, B)$ is randomly generated. Let $\hat{\rho}_h$ denote the proportion of traffic assigned to LSP h by the proposed heuristic. We measure the performance by using the error measure

$$\max_{h=1,\ldots,B} \frac{\rho_h - \hat{\rho}_h}{\rho_h}. \tag{32}$$

For each value of D and B, ten random sets of data are generated. The maximum error over the ten experiments for different values of D and B are shown in Table 1. It can be seem from the table that the algorithm performs very well.

Remark 1. Note that the heuristic given in this section may produce a solution which violates the per-link capacity constraint, especially when the utilisation is high. In this case, the excess capacity

Table 1
Performance of greedy heuristic for optimisation problem **OPT3**

B	$D = 100$	$D = 1000$
2	1.4×10^{-2}	3.2×10^{-4}
3	1.2×10^{-2}	2.3×10^{-5}
4	8.1×10^{-3}	6.7×10^{-5}

which causes the violation can be re-routed by for example the widest-shortest path method [20]. If this fails, an admission control problem will have to be solved, see Section 2.3.5.

3.2. Controlling the number of LSPs

Section 3.1 presents an heuristic solution to **OPT1a** and **OPT1b** by ignoring the constraint on the number of LSPs or routes. Note that the removal of this constraint means that we have lost control over the number of LSPs to be used. The aim of this section is to present an indirect method to control the number of LSPs.

Recall from Section 3.1, the decision variables for optimisation problems **OPT2a** and **OPT2b** are $x_{ij}^{s,k}$ which is defined as the fraction of aggregate demand between ingress–egress pair (i,j) of service class s to be routed over the path $p_{i,j}^{s,k}$. It can be seen from this definition that the number of LSPs being used is given by the number of nonzero $x_{ij}^{s,k}$'s in the solution to **OPT2b**. In order to adjust the number of LSPs being used, we introduce a multiplication factor $\gamma \geqslant 1$ and modify **OPT2b** to **OPT2b**$'$ as follows:

OPT2b$'$

$$\min \sum_{uv} c_{uv} z_{uv}^s \tag{33}$$

subject to the constraints

$$z_{uv}^s \leqslant \gamma \mu^* b_{uv} \quad \forall e_{uv} \in \mathscr{E}, \tag{34}$$

$$\sum_k x_{ij}^{s,k} = 1 \quad \forall i,j \in \mathscr{V}, \tag{35}$$

$$x_{ij}^{s,k} \in [0,1] \quad \forall i,j \in \mathscr{V},\ \forall k. \tag{36}$$

The optimisation problem **OPT2b**$'$ is identical to **OPT2b** except that the multiplication factor γ appears in constraint (34). For $\gamma = 1$, **OPT2b**$'$ is the same as **OPT2b**. This multiplication factor γ allows us to indirectly control the number of LSPs being used. We first prove the following result.

Theorem 2. *Let $x_{ij}^{s,k*}$ denote the optimal solution to* **OPT2b**$'$ *and z_{uv}^{s*} the resulting bandwidth usage in physical link e_{uv}. If the linear programming* **OPT2b**$'$ *is non-degenerate, then*

$$\begin{aligned} &\text{Number of non-zero } x_{ij}^{s,k*} \\ &\quad = \text{number of links such that ``} z_{uv}^{s*} = \gamma\mu^* b_{uv} \text{''} \\ &\qquad + |\mathscr{V}|(|\mathscr{V}|-1). \end{aligned} \tag{37}$$

Proof. See Appendix A. □

As the number of LSPs being used is given by the number of non-zero $x_{ij}^{s,k}$, the number of LSPs is therefore given by Eq. (37). With the assumption that the aggregate demand for each ingress–egress pair is non-zero and the fact that at least one LSP will be used between each ingress–egress pair, the minimum possible number of LSP required per traffic class is thus $|\mathscr{V}|(|\mathscr{V}|-1)$, as given by the fully meshed topology. Eq. (37) says that the number of additional LSPs required, on top of a set of fully meshed LSPs, is given by the number of links such that $z_{uv}^{s,*} = \gamma\mu^* b_{uv}$, in other words, the number of additional LSPs required is given by the number of links with maximum allowed utilisation (i.e. bottleneck links) in the optimisation problem **OPT2b**$'$. By increasing the value of γ, the number of bottleneck links decreases and so does the number of LSPs.

Theorem 2 also shows the tradeoff between achieving good load balancing (as indicated by having a value of γ close to unity) and the number of LSPs being used. A designer can therefore use γ to adjust this tradeoff. This will be demonstrated in Section 4.

Remark 2. The term $|\mathscr{V}|(|\mathscr{V}|-1)$ appears in Theorem 2 because we assumed that the aggregate demand for each ingress–egress pair is non-zero. If this assumption does not hold, we should replace this term by the number of non-zero aggregate demands.

4. Example

In this section we demonstrate the effectiveness of our algorithms using a network with 17 nodes and 58 links.

4.1. Quality of the heuristic

We apply our heuristic to a number of simulated scenarios. In each simulation, M VPNs are generated and each VPN demand is chosen randomly from the range $[t_{\min}, t_{\max}]$. The set of potential paths is all paths within a hop limit $h_{\max}$. The results of these simulations are summarised in Table 2. It shows the maximum link utilisation $\mu_{\text{heuristic}}$ and resource usage $u_{\text{heuristic}}$ given by our heuristic, as well as the optimal objective function value of **OPT2a** and **OPT2b**, which are denoted by, μ_{OPT2a} and u_{OPT2b}, respectively. Note that μ_{OPT2a} is a lower bound of $\mu_{\text{heuristic}}$. The table shows that sometimes $u_{\text{heuristic}}$ and u_{OPT2b} have equal value, this is due to the fact that all split demands are routed over paths with the same number of hops. We can see from the table that the heuristic generally gives very accurate solution.

If the set of potential paths is restricted to 6 hops or less, than there are about 11,000 paths. For the case of unrestricted hop limit, the number of potential paths is almost 50,000. If we are to solve the integer programming problem for this case, it will have over 5 million binary decision variables if there are 100 VPNs. Thus, standard MIP software is unlikely to be able to solve **OPT1a** and **OPT1b**. However, our heuristic solution based on **OPT2a** and **OPT2b** requires solving two LPs with 50,000 variables, which can be solved in approximately 13 s by CPLEX [10], as indicated in the last column of Table 2.

4.2. Comparing different optimisation objectives

The simulation in this section assumes there are 100 VPNs and the demand for these VPNs are randomly generated. There are altogether 3 service classes. The set of potential paths for Service Class 1 has 6 hops or less. Those for Service Class 2 have 9 hops or less, and there is no restriction on the number of hops for Service Class 3.

We first consider the case where the constraint on the number of LSPs is removed. We will compare the effect of the choice of optimisation criterion on the traffic distribution. Three criteria are used. The first criterion is based on minimising the total network resource usage alone. The second criterion is based on minimising only the maximum link utilisation. The last criterion is the multiobjective programming formulation proposed in this paper. For each choice of optimisation criterion, we solve the optimisation problem first for Service Class 1, and then for Service Class 2 using the residual network, and finally for Service Class 3. The results are summarised in Table 3. We see that if we minimise the resource usage alone, it gives the smallest total resource usage among the three criteria but some links (6 in this case) are fully utilised. In contrary, minimising the maximum utilisation gives the smallest maximum link utilisation but results in a large resource usage. However, the multiobjective formulation gives a near Pareto optimal result.

We discuss in Section 2.3.2 (in the paragraph above Fig. 1) that there are numerous solutions

Table 2
This table compares the solutions obtained from the heuristic and continuous relaxation

M	$[t_{\min}, t_{\max}]$	$h_{\max}$	$\mu_{\text{heuristic}}$	$u_{\text{heuristic}}$	μ_{OPT2a}	u_{OPT2b}	CPU time for LPs (s)
2	[500,6000]	6	0.4725	4410022	0.4504	4410022	1.54
2	[500,8000]	6	0.5703	5553002	0.5607	5554165	1.52
5	[100,3000]	6	0.5264	5062718	0.5236	5063423	1.46
5	[500,3000]	6	0.6226	6237955	0.6165	6238369	1.67
10	[500,2000]	6	0.9264	9288769	0.9245	9288799	1.90
20	[100,1000]	6	0.7523	7478913	0.7517	7478913	1.63
50	[100,400]	6	0.9139	9325011	0.9137	9325039	1.60
50	[50,500]	6	0.9038	9291247	0.9036	9291280	1.28
100	[50,200]	unrestricted	0.90074	9278071	0.9007	9278066	12.83

Table 3
Results for the example in Section 4

Optimisation criterion	Maximum link utilisation	Network resource usage (Gbps)	Mean path hop counts	Number of aggregate demands using the shortest paths (total # of aggregate demands = 816)
Minimum resource	1.000	84.1	2.2885	813
Minimax link util.	0.815	143.0	3.8818	211
Multiobjective	0.815	84.2	2.2932	803

which minimise the maximum utilisation but with different resource usage. Fig. 2 shows the distribution of the link utilisation in the network for the optimal solution of each the three optimisation criteria we consider. Note the we have sorted in the link utilisation before plotting Fig. 2. It can be seen that if we optimise only the resource usage, a number of links have utilisation close to one, which means uneven link utilisation distribution. For the case where we optimise only maximum link utilisation, we see that a significant number of links are having an utilisation which is close to the maximum link utilisation of the network. This is due to the fact that resource usage is not minimised. If resource usage is taken into account in addition to maximum link utilisation, as in the proposed multiobjective framework, we see that the number of links with an utilisation close to the maximum has significantly reduced.

In the multiobjective formulation, the total number of LSPs required for all three service classes together is 854, or it requires 38 more LSPs than if we use a fully meshed topology for each service class. This means that most of the aggregate demands are routed using one path. Of all those aggregate demands that are split into multiple routes, all but four demands uses 2 routes and the other four demands use 3 routes. Also, 25 of the 38 split demands are routed along paths with the same hop count, 12 are routed along paths with one hop difference and 1 is routed along paths with 2 hop difference. This property of routing split demands over paths with small difference in path length is precisely what the second optimisation problem is trying to achieve.

In terms of the path chosen by the multiobjective optimisation, 803 out of the 854 aggregate demands are routed along the shortest paths or split among multiple shortest paths. Also, it is observed that all traffic is routed along a path whose hop count is within 2 hops of the shortest path. A similar observation also appeared in [15]. This explains why the resource usage in the multiobjective case is similar to that for minimising resource usage alone. The mean hop count in the second last column of Table 3 provides similar evidence.

The above result is obtained by using the method discussed in Section 3.1 where the constraints on the number of LSPs is being set aside. We use the method presented in Section 3.2 to study the effect of the multiplication factor γ on the number of LSPs required. The same value of γ is used for each traffic class. Fig. 3 shows the variation in the the total number of LSPs, maximum link utilisation and resource usage as we vary γ. The figure clearly shows there is a tradeoff between achieving good load balancing and minimising the number of paths. However, the total resource usage does not seem to vary much as γ changes.

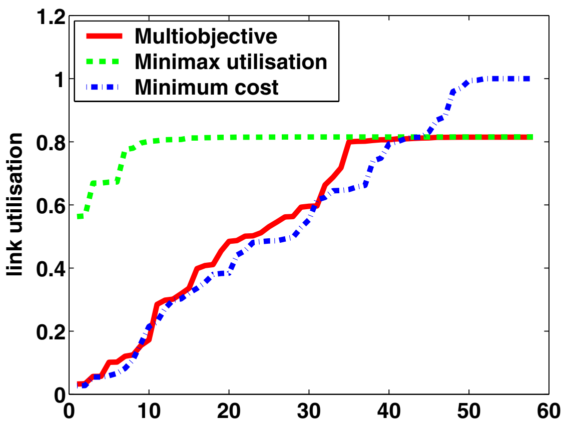

Fig. 2. This graph shows the link utilisation resulted from using the three optimisation criteria. The link utilisation have been sorted in ascending order.

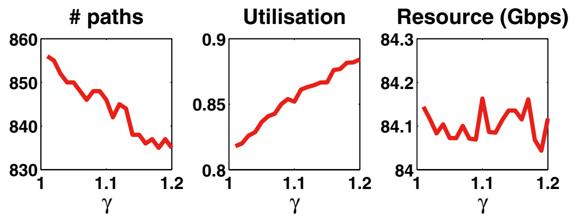

Fig. 3. This graph shows how the number of LSPs, maximum link utilisation and total network resource usage as the multiplication factor γ (see **OPT2b**′ in Section 3.2) varies.

This can be explained by the fact that the number of split demands is small compared with the total number of demands and the demands tend to split between paths with similar length. This indicates that the tradeoff in the VPN optimisation problem is between the goodness in load balancing and the number of LSPs being used.

5. Conclusions

In this paper we have proposed a multiobjective formulation of the MPLS-based VPN traffic engineering problem. This multiobjective formulation takes resource usage, maximum link utilisation and number of LSPs into account. We demonstrate that this multiobjective formulation overcomes the problems of single objective formulations (e.g. minimising resource usage and minimising maximum link utilisation) that have appeared in the literature. The optimisation problem that we have formulated is NP-complete and involves a large number of binary decision variables. We have proposed an heuristic solution, which allows tractable solution.

Although we have formulated this multiobjective problem in terms of the MPLS-based VPN traffic engineering problem, this multiobjective framework is equally applicable to traffic engineering problems in other settings.

Appendix A

Proof of Theorem 1. We will prove that for the special case $B = 2$. In this case, the decision variables are q_{g1} and q_{g2}. By substituting $q_{g2} = 1 - q_{g1}$ in the optimisation problem, **OPT3** is equivalent to

$$\min \left| T\rho_1 - \sum_{g=1}^{D} t_g q_{g1} \right| \quad \text{with } q_{g1} \in \{0,1\}.$$

The solution to this optimisation is given by the minimum of the following two optimisation problems:

$$\min T\rho_1 - \sum_{g=1}^{D} t_g q_{g1} \quad \text{s.t. } T\rho_1 \geqslant \sum_{g=1}^{D} t_g q_{g1},$$
$$q_{g1} \in \{0,1\},$$

$$\min -T\rho_1 + \sum_{g=1}^{D} t_g q_{g1} \quad \text{s.t. } T\rho_1 \leqslant \sum_{g=1}^{D} t_g q_{g1},$$
$$q_{g1} \in \{0,1\}.$$

The first optimisation is equivalent to

$$\max T\sum_{g=1}^{D} t_g q_{g1} \quad \text{s.t. } T\rho_1 \geqslant \sum_{g=1}^{D} t_g q_{g1}, \quad q_{g1} \in \{0,1\}.$$

By using the fact that $T = \sum_{g=1}^{D} t_g$ and $q_{g2} = 1 - q_{g1}$, the second problem is equivalent to

$$\max T\sum_{g=1}^{D} t_g q_{g2} \quad \text{s.t. } T\rho_2 \geqslant \sum_{g=1}^{D} t_g q_{g2}, \quad q_{g2} \in \{0,1\}.$$

Both of these problems are subset sum problems (a special case of knapsack problem where the cost and weight of each item are equal) [17], which are known to be NP-complete [8]. □

Proof of Theorem 2. By introducing slack variables σ^s_{uv}, **OPT2b**′ is equivalent to the following formulation

$$\min \quad \sum_{uv} c_{uv} z^s_{uv} \tag{A.1}$$

subject to the constraints

$$z^s_{uv} + \sigma^s_{uv} = \gamma\mu^* b_{uv} \quad \forall e_{uv} \in \mathscr{E}, \tag{A.2}$$

$$\sum_k x^{s,k}_{ij} = 1 \quad \forall i,j \in \mathscr{V}, \tag{A.3}$$

$$x^{s,k}_{ij} \geqslant 0 \quad \forall i,j \in \mathscr{V},\ \forall k, \tag{A.4}$$

$$\sigma^s_{uv} \geqslant 0 \quad \forall e_{uv} \in \mathscr{E}. \tag{A.5}$$

Let σ^{s*}_{uv} denote the optimal value of σ^s_{uv}. By non-degeneracy, we have the number of non-zero variables equals to the number of equality constraints. Hence

$$\begin{aligned}&\text{Number of non-zero } x_{ij}^{s,k*}\\ &\qquad + \text{number of non-zero } \sigma_{uv}^{s*}\\ &\quad = \text{number of links} + |\mathscr{V}|(|\mathscr{V}|-1).\end{aligned}$$

The result then follows from the fact that

$$\begin{aligned}&\text{Number of links} - \text{number of non-zero } \sigma_{uv}^{s*}\\ &\quad = \text{number of zero } \sigma_{uv}^{s*}\\ &\quad = \text{number of links such that ``} z_{uv}^{s*} = \gamma\mu^* b_{uv} \text{''}.\end{aligned}$$

□

References

[1] D. Awduche et al. Requirements for traffic engineering over mpls, Request for Comments 2702, Internet Engineering Task Force (IETF), 1999.

[2] H. Abrahamsson, B. Ahlgren, J. Alonso, A. Andersson, P. Kreuger, A multi path routing algorithm for IP networks based on flow optimisation, in: Workshop on Quality of Future Internet Services, 2002.

[3] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows, Prentice Hall, Englewood Cliffs, NJ, 1993.

[4] P.C. Chu, J.E. Beasley, A genetic algorithm for the multidimensional knapsack problem, Journal of Heuristics 4 (1998) 63–86.

[5] A. Elwalid, C. Jin, S. Low, I. Widjaja, MATE: MPLS adaptive traffic engineering. in: INFOCOM, 2001.

[6] B. Fortz, M. Thorup, Optimizing OSPF/IS-IS weights in a changing world, IEEE Journal of Selected Areas in Communications 20 (4) (2002) 756–767.

[7] D. Ghosh, N. Chakravarti, A competitive local search heuristic for the subset sum problem, Computers and Operations Research 26 (3) (1999) 271–279.

[8] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman, San Francisco, CA, 1979.

[9] M.K. Girish, B. Zhou, J.Q. Hu, Formulation of the traffic engineering problems in MPLS based IP networks, in: Proceedings of the 5th International Symposium on Computers and Communications (ISCC 2000), 2000, pp. 214–219.

[10] ILOG, ILOG CPLEX 8.0 User's Mannual, July 2002.

[11] A. Juttner, B. Sziatovszki, A. Szentesi, D. Orincsay, J. Harmatos, On-demand optimization of label switched paths on mpls networks, in: Proceedings of the 9th International Conference on Computers Communications and Networks 2000, 2000, pp. 107–113.

[12] K. Kar, M. Kodialam, T.V. Lakshman, Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering, IEEE Journal of Selected Areas in Communications 18 (12) (2000) 2566–2579.

[13] H. Kellerer, R. Mansini, M.G. Speranza, Two linear approximation algorithms for the subset-sum problem, European Journal of Operational Research 120 (2000) 289–296.

[14] F. Le Faucheur et al. Requirements for support of diff-serv-aware MPLS traffic engineering, Internet Draft draft-ietf-tewg-diff-te-reqts-06.txt, Internet Engineering Task Force (IETF), 2002. Work in progress.

[15] Y. Lee, Y. Seok, Y. Choi, C. Kim, A constrained multipath traffic engineering scheme for MPLS networks, in: IEEE International Conference on Communications, 2002, pp. 2431–2436.

[16] D. Mitra, J.A. Morrison, K.G. Ramakrishnan, Virtual private networks: joint resource allocation and routing design, in: Proceedings of the IEEE INFOCOMM'99, vol. 2, 1999, pp. 480–490.

[17] S. Martello, P. Toth, Knapsack Problems, Wiley, New York, 1990.

[18] E. Rosen, Y. Rekhter, BGP/MPLS VPNs, Request for Comments 2547, Internet Engineering Task Force (IETF), 1999.

[19] N.Y. Soma, P. Toth, An exact algorithm for the subset sum problem, European Journal of Operational Research 136 (2002) 57–66.

[20] Z. Wang, J. Crowcroft, Quality-of-service routing for supporting multimedia applications, IEEE Journal of Selected Areas in Communications 14 (7) (1996) 1228–1234.

[21] Y. Wang, Z. Wang, Explicit routing algorithms for Internet traffic engineering, in: Proceedings of the 8th International Conference on Computer Communications and Networks, 1999, pp. 582–588.

[22] X. Xiao, A. Hannan, B. Bailey, L.M. Ni, Traffic engineering with MPLS in the Internet, IEEE Network 14 (2) (2000) 28–33.