Supporting Differentiated QoS in MPLS Networks

Roberto A. Dias[1], Eduardo Camponogara[2], and Jean-Marie Farines[2⋆]

[1] Federal Technology Center of Santa Catarina, Florianópolis, 88020-300, Brazil
[2] Federal University of Santa Catarina, C.P. 476, Florianópolis, 88040-900, Brazil
roberto@cefetsc.edu.br, camponog@das.ufsc.br, farines@das.ufsc.br

Abstract. This paper proposes a new approach for providing different levels of QoS in IP networks over MPLS. The system's operator wishes to maximize the throughput of the high priority flows by choosing the less congested paths while taking care of the applications' requirements. These requirements give rise to an optimization problem that consists of finding resource-constrained shortest paths in a directed graph. The problem can be formally cast in mathematical programming and its solution can follow two directions: (i) a centralized, heuristic approach that aims at approximating the optimal solution in a fast and efficient way; (ii) a distributed approach that relies on Lagrangean relaxation to decompose the optimization problem in small subproblems and thereby divide the computational burden between distributed routers. By means of numerical analysis and simulation, this paper demonstrates the effectiveness of the proposed approaches and shows QoS improvements of the high priority flows.

1 Introduction

In a context of Internet expansion, Traffic Engineering (TE) can deliver a network operation that meets stringent QoS requirements of applications by optimizing the use of network resources. A key element to support TE in IP networks is the *Multiprotocol Label Switching* (MPLS) technology, to a great extent because MPLS implements explicit routing whereby data packets are transmitted in virtual paths denominated *Label Switched Paths* (LSPs). In this paper we propose a TE approach based on an optimization problem which consists of finding shortest paths in a directed graph, while respecting multiple constraints.

Further, the TE problem implements an admission control policy that, under circumstances of traffic congestion, favors the admission of high priority flows in detriment of low priority flows. However, the mathematical formulation of the TE operation renders a computationally hard problem (Section 2): the knapsack problem [1] can be easily reduced to the TE problem in polynomial time. To circumvent both the computational hardness and the large size of typical instances, we propose two approaches: (i) a simple and effective centralized heuristic (Section 3); and (ii) the decomposition of the TE problem into a set of smaller subproblems to be solved distributively (Section 4).

⋆ This research was supported in part by CNPq (Brazil)

2 Traffic Engineering Problem (TEP) Formulation

Our approach to service differentiation in MPLS networks is based on the formulation and solution of a *Traffic Engineering Problem* (TEP): the problem goal is throughput maximization of the flows transmitted by the communication network; the constraints are bandwidth limits and end-to-end maximum transmission delay; and a prioritization scheme is implemented to differentiate among service classes. Although our model can support an arbitrary number of priority levels, the numerical experiments herein use only two priority levels, which can be mapped in two classes of services: (i) *high priority class* which corresponds to premium applications; and (ii) *low priority class* which encompasses the best effort flows. Our model can cope with multiple classes of service which, in turn, can be mapped into service classes of the DiffServ architecture at ease.

The flows are forwarded in LSPs configured along paths connecting source and destination nodes, which adequately allocate bandwidth to avoid congestion. Path and bandwidth are subject to network constraints, as link bandwidth limitations and maximum end-to-end delays. Thus, the problem can be thought of as a *weighted maximum-flow multi-path problem subject to multiple constraints* [2] that, in its single-path form, renders an NP-Hard problem [3].

Our formulation of TEP also implements an admission control policy. Every flow forwarded in its respective LSP is labelled with a priority tag to indicate its importance. Under conditions of network congestion, the transmission rate of low priority flows can be reduced or, in extreme circumstances, dropped to zero to ensure QoS of high priority flows.

The bandwidths of the flow requests are discretized in levels that vary from a maximum desired value to zero, the latter one means the rejection of the request.

The network topology consists of a directed graph $G = (V, E)$, where $V = \{1, \ldots, N\}$ is the set of network nodes and $E \subseteq V \times V$ is the set of transmission links. The capacity of link (i, j) is μ_{ij} $Kbps$, and the time delay for transmitting data through this link is denoted by c_{ij}. The goal is to maximize the throughput weighted by the priority parameters. The variables define the paths and bandwidth levels for the LSPs. More specifically, $y_k^l \in \{0, 1\}$ takes on value 1 if the bandwidth level l is chosen for the k^{th} LSP, while $x_{ij}^{kl} \in \{0, 1\}$ assumes value 1 if arc (i, j) appears in the path from the source to the destination node of the k^{th} LSP when transmission level l is chosen.

The flow parameters are: (1) the number of LSP requests (K); (2) the source node of the k^{th} LSP ($s_k \in V$); (3) the destination of the flow originated from s_k ($d_k \in V$); (4) the number of bandwidth levels of the k^{th} LSP request (l_k); (5) the transmission rate of the k^{th} LSP forwarded in the l^{th} level (λ_k^l)[3]; (6) the maximum end-to-end transmission delay tolerated by the k^{th} LSP (h_k); and (7) the priority of LSP k (δ_k), where high values mean high priority.

The constraints comprise the bandwidth limits of the links, the limits on transmission delay of the LSPs, and the connectivity constraints that ensure

[3] We assume that the LSP levels are arranged in increasing order, that is, $\lambda_k^l < \lambda_k^{l+1}$ for $l = 1, \ldots, l_k - 1$, thus $\lambda_k^1 = 0$ is used to reject admission of the LSP request.

linkage between source and destination nodes. After introducing the terminology, we can formulate TEP in mathematical programming:

$$z = \text{Max} \quad \sum_{k=1}^{K} \sum_{l=1}^{l_k} \delta_k \lambda_k^l y_k^l \tag{1.1}$$

$$\text{S. to :} \quad \sum_{l=1}^{l_k} y_k^l = 1, \qquad \forall k \in \mathcal{K} \tag{1.2}$$

$$\sum_{k=1}^{K} \sum_{l=1}^{l_k} \lambda_k^l x_{ij}^{kl} \leq \mu_{ij}, \qquad \forall (i,j) \in E \tag{1.3}$$

$$\sum_{l=1}^{l_k} \sum_{(i,j)\in E} c_{ij} x_{ij}^{kl} \leq h_k, \qquad \forall k \in \mathcal{K} \tag{1.4}$$

$$\sum_{\{j:(i,j)\in E\}} x_{ij}^{kl} - \sum_{\{j:(j,i)\in E\}} x_{ji}^{kl} = b_i^k y_k^l, \ \forall i \in V, \forall k \in \mathcal{K}, \forall l \in \mathcal{L}_k \tag{1.5}$$

$$x_{ij}^{kl} \in \{0,1\}, \qquad \forall (i,j) \in E, \forall k \in \mathcal{K}, \forall l \in \mathcal{L}_k \tag{1.6}$$

$$y_k^l \in \{0,1\}, \qquad \forall k \in \mathcal{K}, \forall l \in \mathcal{L}_k \tag{1.7}$$

where: $b_i^k = 1$ if $i = s_k$, $b_i^k = -1$ if $i = d_k$, and $b_i^k = 0$ otherwise. Further, $\mathcal{K}$ is a shorthand for $\{1, \ldots, K\}$ and $\mathcal{L}_k$ denotes the set $\{1, \ldots, l_k\}$. Expression (1.2) ensures that each LSP is configured in precisely one transmission level. Expression (1.3) imposes bandwidth constraints on the communication links. Expression (1.4) models the maximum delay limits of each LSP. Expression (1.5) spells out the connection-oriented constraints of LSPs (i.e., LSP k is configured along one path). Expressions (1.6) and (1.7) define the Boolean constraints of the decision variables.

3 Centralized Solution of TEP

This section shows a heuristic-based centralized solution for TEP called *purely heuristic procedure* (PHP). The steps of the PHP algorithm are detailed below.

Note that PHP forwards a set of flows in the MPLS network at a time, as opposed to approaches driven by one-by-one flow admission and forwarding [4]. By letting $N = |V|$ denote the number of vertices of G and $M = |E|$ denote the cardinality of its edge set, it is straightforward to verify that the running time of PHP is $O(\sum_{k=1}^{K} l_k(N \log N + M))$ if we use Dijkstra's algorithm and a Fibonacci heap as the priority queue to compute shortest paths.

Purely Heuristic Procedure (PHP)

Let $\mathcal{L} = (k_1, \ldots, k_K)$ be a permutation of $\mathcal{K}$ s.t. $\delta_{k_j} \lambda_{k_j}^{l_{k_j}} \geq \delta_{k_{j+1}} \lambda_{k_{j+1}}^{l_{k_{j+1}}}$, $\forall j < K$
Let $t = 1$ be the iteration number
Let $G^t = G$ be the residual network modeling the remaining transmission capacity
Let $\Psi_H = \{x_{ij}^{kl}, y_k^l\}$ be the initial, candidate solution to TEP
    where $x_{ij}^{kl} = 0$ and $y_k^l = 0$ for all $k \in \mathcal{K}$, $(i,j) \in E$, and $l \in \mathcal{L}_k$
For $t = 1, \ldots, K$ do
    Let $k \leftarrow k_t$
    For $l = l_k, \ldots, 1$ and while a path has not been found do
        Use Dijkstra's algorithm to find a path p_k in G^t where (i,j)'s cost is:
            $c_{ij}^t = \infty$ if $\mu_{ij}^t < \lambda_k^l$, and otherwise $c_{ij}^t = c_{ij}$
        If a path p_k has been found and $\sum_{(i,j) \in p_k} c_{ij}^t \leq h_k$ then
            Setup LSP k to follow path p_k and update Ψ_H
            Reduce the capacity of G^t along p_k by λ_k^l units to obtain G^{t+1}

3.1 Using *TEP* for Dynamic Operations

This section aims to present a solution for the *Dynamic Traffic Engineering Problem* (referred to as $DTEP$). Our procedure consists of solving a sequence of TEP problems, $\{TEP_t\}$, which are instantiated over time as flow requests arrive at the computer network and other ones terminate. Consider the time iterate $t = 0$ and the first element TEP_0 of the series $\{TEP_t\}$. By solving TEP_0 with PHP, a solution $(x^{(0)}, y^{(0)})$ is obtained and implemented by the network: the network rejects every LSP request k whose transmission bandwidth allocation is zero, while forwarding the remaining LSPs with the chosen bandwidths along the prescribed paths. During the time interval that elapses while TEP_0 is solved and its solution is implemented, all of the requests received by the computer network are enqueued in a waiting list. These unprocessed requests will be reconsidered for admission when TEP_1 is solved.

3.2 An Experimental Analysis

To validate PHP as a viable procedure to solve TEP and assess the network operation quality resulting from the solution of $\{TEP_t\}$, we considered three experiments. *(i) Performance analysis:* it consists of the computation of the running time of PHP and the assessment of its solution quality for a representative instance of TEP, obtained by way of a comparison with the optimal solutions (or upper bounds computed via linear programming relaxation) produced by the CPLEX solver[4]. *(ii) Quality of service (QoS) analysis:* it is intended to evaluate the impact of PHP in a representative scenario as a computational tool for LSP request admission and routing; the analysis covers the typical QoS parameters of throughput and end-to-end transmission delay. *(iii) Admission control analysis:* its purpose is to evaluate admission-control parameters induced by the solution of a representative $DTEP$ instance with PHP.

[4] ILOG CPLEX 9.0: Getting Started, ILOG Corporation, October, 2003.

Performance Analysis The flows of the experiments were of *constant bit rate* (CBR) type and consisted of two priority types: *(i) low priority flows*: transmission rate between 20 and 150 Kbps with end-to-end delay constraint varying between 100 and 150 ms; *(ii) high priority flows*: transmission rate ranging from 380 to 870 Kbps with end-to-end delay constraint varying from 30 to 150 ms.

Each flow can be configured in one of 7 levels of transmission rate, i.e. $l_k = 7$ for $k = 1, \ldots, K$. The number of levels is an administrative decision that should take into account the traffic patterns. In our experiments, the use of 7 levels improved admission of low priority flows. The lowest level is set to zero Kbps, $\lambda_k^1 = 0$, which corresponds to rejecting the admission of the flow. The transmission rate of the subsequent levels increases exponentially up to the maximum desired transmission rate. The set of flow requests are generated randomly, according to a uniform distribution over the range of the parameters that define the requests. The optimal solution to each TEP instance was found by CPLEX solver, one of the leading optimization packages for mixed integer linear programming optimization. The network adopted for experimental analyses consists of 32 nodes and 18 transmission links, whose topology was taken from [4].

Figure 1 (left) gives the computation time[5] taken by PHP and CPLEX optimizer to solve a number of TEP instances in which we varied the workload (e.g. the number of flow requests). As expected, the computation time to obtain optimal solutions with CPLEX is high—TEP is an NP-Hard problem whose integer linear programming formulation has several decision variables and constraints. On the other hand, PHP is reasonably fast even for the largest instances.

Figure 1 (right) depicts the objective values of the solutions produced by PHP and CPLEX (optimal). The figure reveals that PHP can find nearly optimal solutions within a short time window, thereby supporting the claim that PHP can be effective at solving dynamic traffic engineering problems.

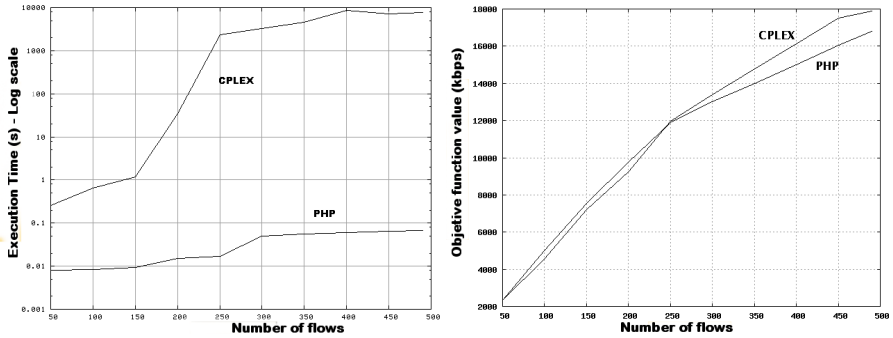

Fig. 1. Running time of PHP versus ILOG CPLEX for varying workloads (left) and quality of PHP solution compared with optimal solutions (right)

[5] Note that the computation times are in logarithmic scale.

Table 1. QoS parameters for high priority flows

Flow Id	Throughput (Kbps)		Delay (ms)		QoS level
	Desired	Granted	Maximum	Incurred	
102	567	567	50	20	optimal
014	608	450	60	50	good
391	680	500	40	15	good
154	753	390	60	50	satisfactory

QoS Analysis The purpose of the analysis herein is to assess the potential of PHP to solve $DTEP$. As a means to analyze simulation results qualitatively, we define three levels of QoS satisfaction: *(i) optimal*, when the maximum throughput is attained; *(ii) good*, when the throughput is at least 70% of the maximum; and *(iii) satisfactory*, when the throughput is at least 50% of the maximum. For a sample of the high priority flows, Table 1 shows the mean values of throughput and end-to-end transmission delay measured in the simulation time interval and compared with the respective maximum values. It is worth mentioning that the end-to-end delay constraints of all of the flow requests were not violated. The table shows that no delay violations are incurred to the high priority flows whose majority obtains good or optimal QoS level.

Admission Control Policy Analysis Herein, we discuss some results regarding the admission control policy implemented by $DTEP$. In the experimental scenario, the mean of flow rejection in PHP solutions was small: about only 3% of the total number of flow requests were rejected, all of which had low priority.

The high priority flows were forced to wait less than 5 ms before being admitted by the network, which is sufficient time for PHP to find a solution to any problem of the sequence $\{TEP_t\}$ instantiated by $DTEP$. Around 3% of the low priority flows wait a time varying from 5 to 10 s before admission, but only about 12% of these low priority requests were dropped from the admission queue after 60 s due to time out.

The simulation results showed that the low priority flows experience a small QoS degradation under highly loaded operating conditions.

4 Distributed Solution of TEP

Heretofore, we have formulated the problem of operating a network as a series $\{TEP_t\}$ of problems, which are periodically solved to respond to the arrival and termination of LSP requests. Despite the fast speed of PHP, its reliance on central computations may become a liability with respect to fault tolerance, flexibility, and scalability. To this end, we will develop a framework to solve TEP_t approximately but distributively, whereby TEP_t is broken down in a set of decoupled Lagrangean subproblems. Section 4.1 deals with the decomposition rendered by relaxing constraints on bandwidth and maximum delay—this

leads to a weaker decomposition but the subproblems are quickly solved with a modified Dijkstra's shortest-path algorithm. Section 4.2 addresses the decomposition obtained by relaxing only the bandwidth constraints—this decomposition is stronger and the computations can be carried out by distributed processes, but it entails solving singly constrained shortest-path problems which are NP-Hard.

4.1 Relaxing Bandwidth and Maximum Delay Constraints

By relaxing the bandwidth constraints (1.3) with Lagrangean multipliers $v = [v_{ij} : (i,j) \in E]$, $v \geq 0$, and the maximum delay constraints (1.4) with $w = [w_k : k \in \mathcal{K}]$, $w \geq 0$, we obtain the Lagrangean dual subproblem $L_1(v,w)$:

$$z_1(v,w) = \text{Max} \quad \sum_{k=1}^{K} \sum_{l=1}^{l_k} \delta_k \lambda_k^l y_k^l + \sum_{(i,j)\in E} v_{ij} \left(\mu_{ij} - \sum_{k=1}^{K} \sum_{l=1}^{l_k} \lambda_k^l x_{ij}^{kl} \right) + \sum_{k=1}^{K} w_k \left(h_k - \sum_{l=1}^{l_k} \sum_{(i,j)\in E} c_{ij} x_{ij}^{kl} \right) \tag{2.1}$$

$$\text{S. to : constraints } (1.2), (1.5), (1.6), \text{ and } (1.7) \tag{2.2}$$

Variable v_{ij} acts as a penalty for bandwidth excess on arc (i,j), while w_k is the penalty for violation of the maximum transmission delay constraint on LSP k.

From the above formulation, it is evident that the variables of different LSPs are decoupled—the couplings originated from the constraints that now appear in the objective function, enabling us to search for LPS routes in parallel. Let $L(n) = \{k : s_k = n\}$ be the LSPs whose source node is n. Notice that $\bigcup_{n=1}^{N} L(n) = \mathcal{K}$. Then, we can break L_1 into a set $\{L_1(v,w,n)\}$ of subproblems, one for each $n \in V$, such that $L_1(v,w,n)$ is the restricted version of $L_1(v,w)$ including only the LSP requests originating from node n (the elements of $L(n)$). Let $z_1(v,w,n)$ denote the objective value of an optimal solution to $L_1(v,w,n)$.

It so happens that each subproblem $L_1(v,w,n)$ consists of a set of decoupled subproblems $L_1(v,w,n,k)$, namely a subproblem for each $k \in L(n)$ defined by:

$$z_1(v,w,n,k) = \text{Max} \quad \sum_{l=1}^{l_k} \delta_k \lambda_k^l y_k^l - \sum_{l=1}^{l_k} \sum_{(i,j)\in E} (v_{ij}\lambda_k^l + w_k c_{ij}) x_{ij}^{kl} \tag{3.1}$$

$$\text{S. to : constraints (2.2) restricted to all } l \in \mathcal{L}_k \tag{3.2}$$

Clearly, $z_1(v,w,n) = \sum_{k\in L(n)} z_1(v,w,n,k)$. An optimal solution to $L_1(v,w,n,k)$ can be obtained by computing l_k shortest paths with a modified Dijkstra's algorithm[6]. With a Fibonacci heap as priority queue, $L_1(v,w,n,k)$ can be solved in $O(l_k(N \log N + M))$ time. Thus, the router at node n can solve $L_1(v,w,n)$ in

[6] Notice that Dijkstra's algorithm can be applied because the cost of each arc (i,j) is given by $(v_{ij}\lambda_k^l + w_k c_{ij}) > 0$. If costs could be negative, then we would have to resort to slower Bellman-Ford's or Johnson's algorithm.

$O(\sum_{k \in L(n)} l_k(N \log N + M))$ time. Because $z_1(v, w)$ establishes an upper bound for z, the natural course of action is to solve the Lagrangean Dual:

$$\begin{aligned} LD_1 : z_1 = \text{Min } z_1(v, w) &= \text{Min } \sum_{n \in V} z_1(v, w, n) + \sum_{k \in \mathcal{K}} w_k h_k + \sum_{(i,j) \in E} v_{ij} \mu_{ij} \\ &= \text{Min } \sum_{n \in V} \sum_{k \in L(n)} (z_1(v, w, n, k) + w_k h_k) + \sum_{(i,j) \in E} v_{ij} \mu_{ij} \end{aligned}$$

Although the Lagrangean dual LD_1 is not likely to produce a lower upper bound than linear relaxation of TEP (Section 10.2 of [1]), the dual solution tends to inherit features of the optimal solution to TEP, in that penalties are incurred for constraint violation. The issues that remain to be resolved are how we (approximately) solve LD_1 and whether the dual solution is satisfactory. Below, we present a distributed implementation of the subgradient algorithm [1] to solve LD_1 and thereafter assess the quality of the dual solutions numerically.

Distributed Subgradient Algorithm The Lagrangean dual LD_1 is convex but nondifferentiable, making it unsuitable to apply efficient gradient-based procedures. The subgradient algorithm can be applied to solve LD_1, which can be viewed as a modified steepest-descent algorithm for nondifferentiable functions. The good news is that the subgradient algorithm can be implemented in a distributed fashion, not unlike the way the Lagrangean subproblem $L_1(v, w)$ can be solved distributively. Put simply, the subgradient algorithm uses the Lagrangean multipliers v_{ij} and w_k as penalty factors to discourage constraint violations: if the bandwidth constraint of a link (i, j) is not violated, then its Lagrangean multiplier v_{ij} is decreased by an amount that depends on the value of the subgradient; or else, the multiplier value is raised. The process is similar for the multipliers w_k. Iterations are performed until convergence is attained.

To implement the subgradient algorithm distributively, it suffices to have each network node i measuring the excess on bandwidth of each link (i, j) and broadcast this excess. This excess is the subgradient π_{ij} associated with the multiplier v_{ij}: v_{ij} will decrease if $\pi_{ij} < 0$ and increase otherwise. The multiplier θ_k for the delay constraint of an LSP k, $k \in L(n)$, can be computed locally by router n. A sketch of the distributed subgradient algorithm follows below.

Notice that $\pi^t_{n,j}$, as computed by router n, is the subgradient associated with arc (n, j): $\pi^t_{nj} = (\mu_{nj} - \sum_{k=1}^{K} \sum_{l=1}^{l_k} \lambda^l_k x^{kl}_{nj})$. At termination, the subgradient algorithm yields multipliers v and w that induce an upper bound $z_1(v, w)$ for z_1. The solution (x, y) to $L_1(v, w)$ is an approximate solution to TEP that, in the absence of constraint violation, is an optimal solution. Because the distributed subgradient algorithm is essentially a decentralized implementation of the standard subgradient algorithm, convergence to optimal Lagrangean multipliers can be ensured if certain rules for decreasing the step length δ^t are followed [1].

The Distributed Subgradient Algorithm

Let $t = 0$ be the iteration number, T be the max. number of iterations,
$\epsilon > 0$ be a small constant, $\delta^t > 0$ be the initial decrement step,

$v^t \in \mathbb{R}^M_+$ and $w^t \in \mathbb{R}^K_+$ be arbitrary Lagrangean multipliers
For $t = 1$ to T do
Wait the routers coordinate to use the same values of t, δ^t, v^t, and w^t
Each router n solves $L_1(v^t, w^t, n)$ to obtain a solution (x^t_n, y^t_n),
where $x^t_n = [x^{kl}_{ij} : k \in L(n), l = 1, \ldots, l_k, \forall (i,j) \in E]$ and
$y^t_n = [y^l_k : k \in L(n), l = 1, \ldots, l_k]$
Let π^t be a subgradient for v^t, whereby each router n computes π^t_{nj}
for all $(n,j) \in E$: π^t_{nj} is the excess on transmission in (i,j)
Let θ^t be a subgradient for w^t. Each router n computes θ^t_k
for all $k \in L(n)$: $\theta^t_k \leftarrow (h_k - \sum_{l=1}^{l_k} \sum_{(i,j)\in E} c_{ij} x^{kl}_{ij})$
Let v^{t+1} be the next Lagrangean multip. for bandwidth. Each router n
computes distributively: $v^{t+1}_{nj} \leftarrow \max\{0, v^t_{nj} + \delta^t \pi^t_{nj}\},\ \forall (n,j) \in E$
Let w^{t+1} be the next multipliers for maximum-delay. Each router n
computes distributively: $w^{t+1}_k \leftarrow \max\{0, w^t_k + \delta^t \theta^t_k\}, \forall k \in L(n)$
Let the routers obtain δ^{t+1} by decreasing δ^t and set $t \leftarrow t + 1$
If $\delta^t < \epsilon$ then stop

Computational Experiments We intend to evaluate the upper bounds from the approximate solutions to LD_1 and, last but not least, assess their quality as approximate solutions to TEP with respect to objective function, bandwidth constraint violation, and maximum delay. We adopted a decreasing schedule for the subgradient step whereby $\delta^0 = 2$ and $\delta^{t+1} = 0.9875\delta^t$. The initial multipliers v^t and w^t were selected at random within $[0, 10]$. The algorithm was allowed to iterate while $\delta^t \geq 10^{-6} = \epsilon$.

Table 2 depicts the results obtained with the (distributed) subgradient algorithm to the instances discussed in Section 3.2. For each instance, let $(\tilde{v}, \tilde{w})$ and $(\tilde{x}, \tilde{y})$ be the Lagrangean multipliers and solution to $L_1(\tilde{v}, \tilde{w})$, respectively. The 1^{st} row gives the upper bound $z_1(\tilde{v}, \tilde{w})$. The 2^{nd} row has the objective value induced by $(\tilde{x}, \tilde{y})$: $f(\tilde{y}) = \sum_{k=1}^{K} \sum_{l=1}^{l^k} \delta_k \lambda^l_k \tilde{y}^l_k$. The 3^{rd} row gives the objective of the optimal solution to TEP. The 4^{th} row gives the relative excess on bandwidth: $e_b(\tilde{x}) = [\sum_{(i,j)\in E} \max\{\sum_{k=1}^{K} \sum_{l=1}^{l_k} \lambda^l_k \tilde{x}^{kl}_{ij} - \mu_{ij}, 0\}] / [\sum_{k=1}^{K} \sum_{l=1}^{l_k} \sum_{(i,j)\in E} \lambda^l_k \tilde{x}^{kl}_{ij}]$. The 5^{th} row gives the delay excess: $e_d(\tilde{x}) = \sum_{k=1}^{K} \sum_{l=1}^{l_k} \max\{\sum_{(i,j)\in E} c_{ij} \tilde{x}^{kl}_{ij} - h_k, 0\}$. The 6^{th} row depicts the average computation time taken by the routers located at backbone nodes to run the subgradient algorithm.

From the table, we can infer that the objective value of the solutions produced by the subgradient algorithm are not significantly inferior to the optimal ones, and they can be attained far more quickly than using CPLEX. The results show that the excess on bandwidth capacity is significant, of the order of 23% of the total bandwidth allocation, while the excess on transmission delay is quite low.

4.2 Relaxing Bandwidth Constraints

Here we look into the decomposition of TEP yielded by relaxing only the bandwidth constraints (1.3). Given a set of Lagrangean multipliers $v = [v_{ij} \in \mathbb{R}_+ :$

Table 2. Quality of the solution to the Lagrangean dual LD_1

Parameters	Number of LSP Requests (K)									
	50	100	150	200	250	300	350	400	450	490
$z_1(\tilde{v}, \tilde{w})$	2354	4539	7222	9503	12276	13178	14996	16430	17585	18020
$f(\tilde{y})$	2354	4539	7217	9227	11996	13178	14909	16200	16358	16995
Optimal objective	2354	4539	7217	9227	11955	13173	14771	16134	17150	17898
$e_b(\tilde{x})$ (%)	0	0	20.11	24.26	23.88	33.25	29.62	23.26	23.83	25.22
$e_d(\tilde{x})$	0	0	0	8	0	0	14	13	0	1
Comp. time (s)	0.215	0.324	1.418	1.986	2.451	2.923	3.505	3.994	4.483	4.870

$(i,j) \in E$] for bandwidth, the Lagrangean subproblem $L_2(v)$ can be cast as:

$$z_2(v) = \text{Max} \quad \sum_{k=1}^{K} \sum_{l=1}^{l_k} \delta_k \lambda_k^l y_k^l + \sum_{(i,j)\in E} v_{ij} \left(\mu_{ij} - \sum_{k=1}^{K} \sum_{l=1}^{l_k} \lambda_k^l x_{ij}^{kl} \right) \tag{4.1}$$

$$\text{S. to :} \quad \text{constraints } (1.2), (1.4), (1.5), (1.6), \text{ and } (1.7) \tag{4.2}$$

As in the above developments, $L_2(v)$ consists of K decoupled subproblems of choosing the transmission level of each LSP k, y_k^l, and selecting a path from the source s_k to the destination d_k without exceeding maximum delay in transmission—the bandwidth capacity constraints were dualized and placed as penalty factors in the objective. First, let us break up $L_2(v)$ into N subproblems to be solved at each backbone node n, hereafter denoted by $L_2(v,n)$ whose optimal solution has value $z_2(v,n)$. Because a solution to $L_2(v)$ can be obtained by aggregating the solutions to the elements of $\{L_2(v,n)\}$, the objective value of the Lagrangean subproblem becomes: $z_2(v) = \sum_{n\in V} z_2(v,n) + \sum_{(i,j)\in E} v_{ij}\mu_{ij}$.

Continuing the problem break up, each $L_2(v,n)$ can be spliced into $|L(n)|$ decoupled subproblems, one for each $k \in L(n)$. Let $\{L(v,n,k)\}$ be the subproblems obtained by decomposing $L(v,n)$, with $L(v,n,k)$ defined by:

$$z_2(v,n,k) = \text{Max} \quad \sum_{l=1}^{l_k} \delta_k \lambda_k^l y_k^l - \sum_{l=1}^{l_k} \sum_{(i,j)\in E} v_{ij} \lambda_k^l x_{ij}^{kl} \tag{5.1}$$

$$\text{S. to :} \quad \text{constraints } (4.2) \text{ restricted to all } l \in \mathcal{L}_k \tag{5.2}$$

The end result is the break up of $L_2(v)$ into a set $\{L_2(v,n,k) : n \in V, k \in L(n)\}$ of subproblems that can be solved asynchronously. The subset $\{L_2(v,n,k)\}$ is solved by router n and the solution to $L_2(v)$ is obtained by combining the distributed solutions. Thus: $z_2(v) = \sum_{n\in V} \sum_{k\in L(n)} z_2(v,n,k) + \sum_{(i,j)\in E} v_{ij}\mu_{ij}$.

Calculating $z_2(v,n,k)$ entails solving l_k constrained shortest path problems. So $L_2(v,n,k)$ is an NP-Hard problem [5]. One way of solving $L_2(v,n,k)$ consists in computing l_k constrained shortest path with dynamic programming, one for each level of transmission of the k^{th} LSP. Given the penalty vector v, a router node $n \in V$, an LSP $k \in L(n)$, and a transmission level $l \in \{1, \ldots, l_k\}$, the problem of finding a shortest path from s_k to d_k subject to the maximum delay

Table 3. Quality of the solution to the Lagrangean dual LD_2

Parameters	Number of LSP Requests (K)							
	50	100	150	200	250	300	350	400
Upper bound $z_2(\tilde{v})$	2354	4539	7217	9283	12126	13455	14862	16336
Obj func $f(\tilde{y})$	2354	4539	7217	9227	11996	13379	14909	16107
Bandwidth excess $e_b(\tilde{x})$ (%)	0	5.68	18.15	34.24	40.48	46.79	30.29	21.97
Comp time (s)	88.62	95.50	144.25	183.12	237.37	287.62	339.37	385.25

constraint can be expressed recursively, provided that the parameters h_k are integers. Let $d(i,t,h)$ be the distance of a shortest path from node i to node d_k that contains at most t arcs and whose transmission delay is at most h. Having introduced this terminology, the recurrences can be stated as:

$$\begin{cases} d(i,0,h) &= +\infty,\ \forall i \in V - \{d_k\}, h = 0,\ldots,h_k \\ d(d_k,0,h) &= 0,\ h = 0,\ldots,h_k \\ d(i,t,h) &= \min\{d(i,t-1,h), \\ & \qquad \min\{v_{ij}\lambda_k^l + d(j,t-1,h-c_{ij}) : (i,j) \in E, c_{ij} \leq h\}\}, \\ & \qquad \forall i \in V, t = 1,\ldots,N-1, h = 0,\ldots,h_k \end{cases} \tag{6}$$

An optimal solution to $L(v,n,k)$ can be obtained by solving a set $\{L(v,n,k,l) : l = 1,\ldots,l_k\}$ of subproblems, where $L(v,n,k,l)$ is the problem $L(v,n,k)$ in which $y_k^l = 1$. It is straightforward to solve the recurrences (6) with dynamic programming to reach a solution to $L(v,n,k,l)$. For details on the steps of the dynamic programming algorithm, the interested reader can refer to [5].

The issues regarding the solution of LD_2 via a subgradient procedure and convergence to an optimal value z_2 are, in essence, identical to the ones raised in Section 4.1. To divide the computational burden of solving LD_2 among the routers, we can design a distributed subgradient algorithm by discarding the Lagrangean multipliers w of the preceding (distributed) subgradient algorithm, and solving $z_2(v)$ distributively in each step rather than $z_1(v,w)$.

Computational Experiments We implemented the subgradient algorithm to solve LD_2 approximately. The numerical results reported in Table 3 provide evidence that the upper bound $z_2(\tilde{v})$ is tighter than $z_1(\tilde{v},\tilde{w})$. However, the dual solution $(\tilde{x},\tilde{y})$ induced by solving LD_2 is inferior to that obtained by solving LD_1, namely the bandwidth violation is higher than that incurred by the solution to LD_1, and the computational cost far exceeds the cost of solving LD_1. An alternative to expedite the solution process is to resort to an approximate algorithm to solve $L_2(v,n,k,l)$ such as those from [6]. An alternative to reduce the duality gap is to solve the Lagrangean dual with the methods from [5].

5 Discussion and Concluding Remarks

Several works aim to solve similar network operating problems to ours. But the differences among the models, mainly in the order of LSP admission, prevents

a quantitative comparison. In [7], the authors propose a load balancing scheme based on an admission control policy that considers only the link bandwidth capacity. But their proposal does not optimize the network performance. In [4], the authors deal with a generalization of a multi-objective shortest path problem subject to multiple constraints. To date, our developments do not handle multiple objectives but they can be extended as in [4]. In [8], the authors propose a distributed Lagrangean optimization approach to solve distributively a TE problem. However, they assume that the routes connecting source and destination nodes are given and they consider only constraints on link bandwidth, whereas we design paths and currently tackle link bandwidth and also transmission limits.

Our work has shown that the PHP framework attains a high degree of optimality, demanding low computational cost and being simpler than alternative algorithms for solving TE problems. Unlike other solution approaches, our framework implements an admission control policy to manage flows with different levels of priority.

The Lagrangean relaxation of the constraints on bandwidth and end-to-end transmission delay decomposed TEP into a set of smaller subproblems, $\{TEP_n\}$, one for each router n. The preliminary analysis of the distributed solution of $\{TEP_n\}$ has shown promise. The decomposition LD_1, obtained by dualizing bandwidth and end-to-end delay constraints, yielded tight upper bounds and was relatively fast to compute. The decomposition LD_2, obtained by dualizing only the bandwidth constraints, yielded tighter upper bounds but at the expense of higher computational cost and relying on a more complex, time-consuming dynamic programming algorithm. Two possibilities are the design of heuristics to keep the violations under limits and the application of multi-objective optimization algorithms, which would simultaneously maximize weighted throughput and minimize constraint violation.

References

1. Wolsey, L.A.: Integer Programming. John Wiley & Sons (1998)
2. Girish, M., Zhou, B., Hu, J.Q.: Formulation of the traffic engineering problems in MPLS based IP networks. In: Proc. IEEE ISCC. (2000)
3. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company (1979)
4. Banerjee, G., Sidhu, D.: Comparative analysis of path computation techniques for MPLS traffic engineering. Computer Networks **40** (2002) 149–165
5. Ziegelmann, M.: Constrained Shortest Paths and Related Problems. PhD thesis, Universitat des Saarlandes, Germany (2001)
6. Hassin, R.: Approximation schemes for the restricted shortest path problem. Mathematics of Operations Research **17** (1992) 36–42
7. Salvadori, E., Batiti, R.: A load balancing scheme for congestion control in MPLS networks. In: Proc. IEEE ISCC. (2003)
8. Xiaojun, L., Ness, S.B.: An optimization based approach for QoS routing in high bandwidth networks. In: Proc. IEEE INFOCOM. (2004)