IGP Link Weight Assignment for Operational Tier-1 Backbones

Antonio Nucci, *Senior Member, IEEE*, Supratik Bhattacharyya, Nina Taft, *Senior Member, IEEE*, and Christophe Diot

***Abstract*—Intradomain routing protocols, such as IS-IS or OSPF, associate a weight (or cost) with each link to compute traffic routes. Proposed methods for selecting link weights largely ignore two practical issues, that of service-level agreement (SLA) requirements and of failures. Optimizing the routing configuration, without bounding the SLA, could severely violate this requirement, which is one of the most important vehicles used by carriers to attract new customers. Since most failures are short-lived, it is much more practical not to have to change weight settings during these episodes. In this paper we propose a Tabu-search heuristic for choosing link weights that takes into account both SLA requirements and link failures. Our algorithm selects link weights that still perform well, without having to be changed, even under failure events. To validate the heuristic, we develop a lower bound based on a formal integer linear program (ILP) model, and show that our heuristic solution is within 10% of the optimal ILP lower bound. We study the performance of the heuristic using two operational Tier-1 backbones. Our results illustrate two tradeoffs, between link utilization and the SLA provided, and between performance under failures versus performance without failures. We find that performance under transient failures can be dramatically improved at the expense of a small degradation during normal network operation (i.e., no failures), while simultaneously satisfying SLA requirements. We use our algorithm inside a prototype tool to conduct a case study and illustrate how systematic link weight selection can facilitate topology planning.**

***Index Terms*—Failures, interior gateway protocol (IGP) routing, intermediate system to intermediate system (IS-IS) protocol, open shortest path first (OSPF) protocol, optimization, robustness, tabu search, traffic engineering.**

I. Introduction

LARGE IP networks use a link-state protocol such as IS-IS or OSPF as their interior gateway protocol (IGP) for intradomain routing. Every link in the network is assigned a weight, and the cost of a path is measured as the sum of the weights of all links along the path. Traffic is routed between any two nodes along the minimum cost path which is computed using Dijkstra's shortest path first (SPF) algorithm. These weights and the paths they result in, fully determine the distribution of load across all links in the domain. Thus setting the link weights is a primary traffic engineering technique for operators whose networks run IS-IS or OSPF.

The traffic engineering objectives are largely determined by the service-level agreements (SLAs) that define the performance level a carrier guarantees its customers. Such customer guarantees thus become performance requirements for the carrier. One of the elements in SLAs today is a guarantee on the average end-to-end delay across all origin-destination (OD) node pairs in the network. Hence one of two main goals in setting link weights is to keep the SLA low. The second main goal is to ensure that no link becomes unacceptably overloaded. Several formalizations of the problem of selecting link weights that are optimal with respect to the link overloading objective have been shown to be NP-hard [2], [5].

A common recommendation of router vendors is to set the weight of a link to the inverse of its capacity [16]. The idea is that this will attract more traffic to high-capacity links and less traffic to low-capacity links, thereby yielding a good distribution of traffic load. Another common recommendation is to assign a link a weight proportional to its physical length in order to minimize propagation delays. In practice, many backbone operators use the ad-hoc approach of observing the flow of traffic through the network, and iteratively adjusting a weight whenever the load on the corresponding link is higher or lower than desired. The problem has been addressed formally in [2], [3], [9], [14] using different techniques from operations research. We discuss related work in Section VIII.

Extensions to Basic Problem: One drawback of current approaches is that SLA bounds are not incorporated into the link weight assignment problem. This is a crucial aspect for ISPs since delay bounds are both a vehicle for attracting customers and a performance requirement carriers are contracted to meet. Each ISP defines their own metrics and the values for those metrics in their SLA. For the North merican Sprint IP backbone the SLA is set to 45 ms, whereas for the European Sprint IP backbone, it is set to 20 ms. The average delay between any OD pair must be below the value in the SLA. As our first goal, we seek to incorporate SLA bounds into our problem definition and solution.

Another and more crucial drawback of most current approaches (with the notable exception of [3]) is that they view the link weight assignment problem as a static problem largely ignoring network dynamics. In practice, one of the main challenges for network operators is to deal with link failures that occur on a daily basis in large IP backbones [10]. When a link fails, IS-IS/OSPF routing diverts the traffic from the broken link to alternate paths, increasing the load on one or more other links. The most obvious way of restoring the network, to meet its original traffic engineering objectives, is to perform a

Manuscript received May 16, 2005; revised April 18, 2006, approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor L. Gao. This work was performed while A. Nucci, N. Taft, and C. Diot were with Sprint.

A. Nucci is with Narus, Inc., Mountain View, CA 94043 USA.

S. Bhattacharya is with Sprint Advanced Technology Laboratories, Burlingame, CA 94131 USA.

N. Taft is with Intel Research, Berkeley, CA 94704 USA.

C. Diot is with Thomsom R&D, Paris Research Laboratory, 92648 Boulogne Cedex, France.

Digital Object Identifier 10.1109/TNET.2007.893878

network-wide recomputation and reassignment of link weights. [3] has shown that changing just a few link weights is usually sufficient to rebalance the traffic.

However changing link weights during a failure may not be practical for two reasons. First, the new weights will have to be flooded to every router in the network, and every router will have to recompute its minimum cost path to every other router. This can lead to considerable instability in the network, aggravating the situation already created by the link failure. The second reason is related to the short-lived nature of most of the link failures. In [10] they examined inter-PoP link failures over a 4 month period and found that 80% of the failures last less than 10 minutes and 50% of the failures last less than a minute. They thus define transient failures as those failures that last less than 10 minutes. Transient failures can create rapid congestion that is harmful to the network [17]. However, they leave a human operator with insufficient time to reassign link weights before the failed link is restored. The study in [10] also examined the frequency of single link and multiple link failures. They observed that more than 70% of the transient failures are single-link failures.

As a second goal, we therefore focus on minimizing the impact of transient link failures on traffic. Accordingly, we propose an approach for assigning link weights in IS-IS/OSPF networks that is robust to all single link failure episodes. The goal is to find *one* set of link weights that works well in *both* the normal operating state (i.e., the absence of failures) and during transient failures. By "perform well" during transient failures we mean that none of the remaining active links should be overloaded, in general, nor in such a way as to lead to an SLA violation. We focus on transient failures because of their frequency and since it is for these episodes that we do not want to have to alter weight settings. We also focus on single-link failures at the IP layer since these are the primary cause of transient failures. Other studies [6] have focused on the problem of finding routes at the IP layer that are robust to long-lived multi-link failures.

Summary of Contributions and Findings: The contributions of this paper are multiple. First, we incorporate a critical practical requirement, namely the SLA bounds, into the traditional problem. Second, we introduce the concept of providing robustness without changing weights during short-lived link failure events, and incorporate this second practical requirement into our solution. Third, we develop an algorithm to solve the extended problem based on the Tabu-Search meta-heuristic; our method performs very well and is scalable (at least) to the size of the Tier-1 networks studied. Fourth, we develop an analytical formulation of the problem with link failures via an integer linear pogram (ILP). We use this to derive a lower bound on the impact of SLA requirements and failures which we use to validate our heuristic algorithm. Fifth, we evaluated our solution using two operational Tier-1 backbones.

Our solution has been prototyped into a tool called METL. Having such a heuristic algorithm as the engine in an automated tool enables operators not only to select link weights but to apply the tool for assistance in other traffic engineering tasks. Our last contribution is a case study illustrating three applications of our tool. We show here, for the first time, the impact of the maximum weight value on routing. We explain how the tool can be used to help operators rapidly understand what is the lowest SLA they could offer, and for rapid evaluation of candidate topology designs. Because the SLA offered is a competitive factor that customers use to compare ISPs, the ISPs strive to offer the lowest SLA possible.

Among our findings, we identify a design tradeoff: in order to reduce the SLA, one incurs an increase in the maximum link load. We quantify this tradeoff, and show that we can reduce the 45-ms SLA provided today, in the North American network, to 40 ms at the cost of increasing the maximum link load by approximately 6%. We also illustrate the tradeoffs of including failures inside the optimization procedure. For example, without considering failures in weight selection, the worst case load can rise to 135% when a failure occurs in the network. Including failures in the optimization limits the worst case load to 90% under network failure events. This enhanced performance under failure events comes at the price of an increased maximum load of 10% under normal operation.

This paper extends our work in [14] by increasing the level of formalization, the extent and variety of validation, and through the addition of an important practical constraint. In [14] the SLA was used as a performance metric, whereas in this work it is incorporated directly into the optimization. We presented a heuristic solution alone in [14]; here we have formalized a general ILP to validate our heuristic. In fact, the validation exceeds the previous work not only due to the ILP comparison, but also because herein we use a second sample ISP network with a *real* traffic matrix, and because we compare our solutions to the weights used in today's currently deployed solutions.

The rest of the paper is organized as follows. Section II formally states the problem addressed in this work. Section III describes our model for the general routing problem that considers SLA bounds and single backbone link failure scnearios. Section IV describes the Tabu search heuristic for the link weight selection problem. We explain the topologies and traffic matrices used for evaluation in Section V. The performance of the heuristic is presented in Section VI along with a discussion of the tradeoffs. Section VIII discusses related work and Section IX concludes the paper.

II. Link Weight Selection Problem

The problem of computing IS-IS link weights can be formally stated as follows. We represent a network by an undirected graph $G(\mathcal{V}, \mathcal{L})$ where $\mathcal{V}$ corresponds to the set of nodes and $\mathcal{L}$ corresponds to the links connecting the nodes. Each edge $l \in \mathcal{L}$ has a bandwidth capacity c_l and integer ISIS weight w_l. Let L be the cardinality of the set $\mathcal{L}$. Let D be a traffic matrix representing the traffic demands such that $d^{sd} \in D$ corresponds to the traffic demand between origin node s and destination node d. The traffic demand between an O-D pair s and d is routed along the minimum cost path from s to d, where the cost of a path is measured as the sum of the weights of the links along the path. If multiple paths exist with the same minimum cost, the traffic demand is split equally among all these paths.[1] The problem is to choose a

[1]Technically ISIS/OSPF do equal splitting per outgoing interface, not per path. During our research efforts, we considered both equal cost splitting per outgoing link and per path (in the problem formulation). We found that incorporating the splitting per outgoing link adds considerable complexity into the problem, but yields little difference in final performance. We thus selected the per-path formulation as a good approximation.

single set of link weights $\mathcal{W} = (w_1, w_2, \ldots, w_L)$ so as to optimize a given objective function. The objective function is typically selected to meet the traffic engineering goals of a network. An important goal for IP networks is to distribute traffic evenly so that no link carries an excessive load. A common approach is to choose routes so as to minimize maximum load over all network links.

At the same time, carriers have to guarantee that the new routing configuration satisfies the SLA requirement, defined as the average end-to-end delays for all OD pairs traversing the backbone. Selecting link weights that minimize the maximum link load yet violate the SLA the carrier offers its customers, yields a solution that cannot be deployed in an operational network. In our work, we assume that only propagation delay contributes to end-to-end delay since queueing delays have been observed to be negligible in a core network [18].

We model the occurrence of isolated transient failures in the network as state transitions. In the absence of any such failure, the state of the network is denoted by S_0. During the transient failure of link i ($i = 1, 2, \ldots, L$), the state of the network is denoted by S_i. We refer to the maximum link load of the network in any given state as the *bottleneck load*. Let $\mu(S_j) \geq 0$ be the bottleneck load in state S_j. S_{wst} is defined as the state with the maximum bottleneck load over all network states, i.e., $\mu(S_{wst}) \geq \mu(S_j), \forall j \in [0, L]$. Henceforth we refer to state S_0 as the no-failure state, and the state S_{wst} as the worst-failure state.

To handle both the short-lived failure events and the SLA requirement in the problem, we propose an objective function that penalizes the SLA violation for the no failure state S_0 and limits the maximum link overload due to *any* single-link failure event. Selecting a weight set that is optimal for the network in a failure state may result in suboptimal performance in the absence of failures. We want to meet three objectives: satisfying the SLA constraint, preventing link overloads during a failure while simultaneously minimizing performance degradation in the absence of failures. We thus choose the objective function

$$\mathcal{F} = (1 - W)\mu(S_0) + W\mu(S_{wst}) + B\Delta(S_0) \qquad (1)$$

where $W \in [0, 1]$ is a tradeoff parameter that helps in balancing the two goals minimizing performance degradation in the absence of failures while maximizing the gain for any transient failure. Setting W to 0 corresponds to finding link weights without considering failures. Setting $W = 1$ results in link weights that minimize the maximum load under any failure, but that completely ignore performance in the absence of failures.

The $\Delta(S_0)$ represents the excess SLA (the amount exceeding the requirement) for the no failure state S_0. To compute this, we linearly sum the excess SLA for all routes. If the routing configuration selected meets the bound, then $\Delta(S_0) = 0$. However, if for example, the SLA is set to be 50 ms for all routes and we generate all routes below 50 ms except for two routes with 53-ms SLA, then $\Delta(S_0) = 6$ ms. Let B denote a large scalar number used to give high priority to the SLA constraint. Only solutions characterized by the same $\Delta(S_0)$ value are further differentiated by to the first two terms of $\mathcal{F}$.

III. ILP for General Routing Problem

In this section, we formulate the *general routing problem* with single-link failures as an integer linear programming problem. The goal is to derive a lower bound on the value of the objective function used for METL [refer to (1)].

Since the IS-IS link weight selection problem is NP-hard, it is computationally intractable to obtain the optimal solution. On the other hand, the general routing problem is computationally tractable. We therefore model the general routing problem to obtain a lower bound on the performance of METL.

Unlike the link weight selection problem, the general routing problem directly selects routes between node pairs, instead of doing it indirectly via link weights. Also route selection is not restricted to minimum cost routes based on link weights. Instead, traffic can be arbitrarily routed along several paths between two nodes, and split in arbitrary ratios across these paths. Therefore an optimal solution to the general routing problem yields a lower bound on the objective function value for the link weight selection problem.

A. Notation

We aggregate all the traffic between a given origin-destination (OD) node pair in the network into a single traffic demand. Let $\mathcal{C} = \{c_i\}$ be the set of all OD pairs whose cardinality is described by $C = |\mathcal{C}|$ and $\mathcal{L} = \{l_i\}$ be the set of bidirectional links belonging to the network topology whose cardinality is represented by $L = |\mathcal{L}|$. The traffic demand associated with OD pair $c \in \mathcal{C}$ is represented by d^c. Let $\mathcal{S}$ be the set of all the network states, with cardinality $S = |\mathcal{S}| = L + 1 : \mathcal{S}_0$ represents the no failure case while $\mathcal{S}_j$ with $j > 0$ represents the failure case when the link l_j is broken. Let $R^c(S_j) = \{r_i^c(S_j)\}$ be the set of all the feasible routes for the OD pair $c \in \mathcal{C}$ in the network state $\mathcal{S}_j$. A route in a failure network state S_l is defined as *feasible* if it does not use the broken link l. Moreover, a route is defined feasible, for the no failure state S_0, if and only if it does not violate the SLA constraint defined as an input of the problem. Let $u_{ij}^c(l)$ be the binary parameter that is equal to 1 if the ith route associated with OD pair c in the network state S_j, i.e. $r_i^c(S_j)$, uses the link l and 0 otherwise. Let $v_{ih}^c(S_0, S_j)$ be a binary parameter equal to 1 if $r_i^c(S_0) = r_h^c(S_j)$, and 0 otherwise.

B. Decision Variables

Let $\alpha_i^c(S_j) \in \{0, 1\}$ represent the set of variables used to define which routes are selected by OD pair c in network state S_j. The variable $\alpha_i^c(S_j) = 1$ if the OD pair c uses the ith route in network state S_j ($r_i^c(S_j)$) and 0 otherwise. Let $d_i^c(S_j) \geq 0$ be the fraction of traffic associated with OD pair c that is sent to the ith route in the network state S_j. The variables $\beta^l(S_j) \geq 0$ represent the aggregated load carried by link $l \in \mathcal{L}$ in network state S_j. Let $\gamma_i^c(S_j) \in \{0, 1\}$ describe the feasibility of each route $r_i^c(S_0)$ in each network state S_j; $\gamma_i^c(S_j) = 1$ if OD pair $c \in \mathcal{C}$ uses route $r_i^c(S_0)$ and this route uses the broken link l_j. When a link fails, all OD pairs going over the link must be rerouted. Several feasible routes can be used to reroute this traffic. For route $r_i^c(S_0)$, let $\theta_i^c(S_j) \geq 0$ be the traffic associated with OD pair c that is routed over failed link l_j. In other

words, $\theta_i^c(S_j)$ is equal to $d_i^c(S_0)$ if $\gamma_i^c(S_j) = 1$ and 0 otherwise. Finally let $\mu(S_j) \geq 0$ be the maximum link load in network state S_j, and let $\mu(S_{wst})$ be the maximum link load over all the network states, i.e. $\mu(S_{wst}) = max_{j\in[0,L]}\mu(S_j)$. The corresponding state is denoted by S_{wst}.

C. Constraints

- The constraints associated with the maximum link load in each network state $S_j \in \mathcal{S}$ are

$$\mu(S_j) \geq \beta^l(S_j) \quad \forall S_j \in \mathcal{S}, \forall l \in \mathcal{L} \tag{2}$$

$$\mu(S_{wst}) \geq \mu(S_j) \quad \forall S_j \in \mathcal{S} \tag{3}$$

  where constraint (2) defines the maximum link load in each network state S_j, and (3) defines the maximum link load over all network states.
  The variable $\beta^l(S_0)$ describes the aggregated traffic flowing on the link $l \in \mathcal{L}$ in the no failure state S_0

$$\beta^l(S_0) = \sum_{c\in\mathcal{C}, i\in|R^c(S_0)|} d_i^c(S_0) u_{i0}^c(l) \quad \forall l \in \mathcal{L}. \tag{4}$$

- The constraints associated with routes used in S_0 and their associated traffic are

$$\sum_{i\in|R^c(S_0)|} \alpha_i^c(S_0) \geq 1 \quad \forall c \in \mathcal{C} \tag{5}$$

$$d_i^c(S_0) \leq d^c \alpha_i^c(S_0) \quad \forall c \in \mathcal{C}, \forall i \in |R^c(S_0)| \tag{6}$$

$$\sum_{i\in|R^c(S_0)|} d_i^c(S_0) = d^c \quad \forall c \in \mathcal{C} \tag{7}$$

  where constraint (5) ensures that each OD pair in S_0 has to select at least one route to send its own traffic, while constraint (6) restricts the traffic carried by each route. If $\alpha_i^c(S_0) = 1$, the total amount of traffic for OD pair c_i on this route is restricted by d^c; if $\alpha_i^c(S_0) = 0$ then no part of the traffic for OD pair c_i is sent over this route. The traffic associated with each OD pair has to be sent using one or more routes (7).
- The constraints associated with routes affected by the failure of link $j \in [1, L]$ are

$$\gamma_i^c(S_j) \leq \alpha_i^c(S_0) u_{i0}^c(l_j) \quad \forall c \in \mathcal{C}, \forall i \in |R^c(S_0)|, \quad \forall l_j \in \mathcal{L}, \forall S_j \in \{\mathcal{S} - \mathcal{S}_0\} \tag{8}$$

$$\gamma_i^c(S_j) \geq 2\alpha_i^c(S_0) u_{i0}^c(l_j) - 1 \quad \forall c \in \mathcal{C}, \forall i \in |R^c(S_0)|, \quad \forall l_j \in \mathcal{L}, \forall S_j \in \{\mathcal{S} - \mathcal{S}_0\} \tag{9}$$

  where constraints (8) and (9) detect which routes used in S_0 are involved in the failure of link $l_j \in \mathcal{L}$. Note that if OD pair $c \in \mathcal{C}$ uses the route $r_i^c(S_0)$ and the link l_j belongs to $r_i^c(S_0)$ ($u_c^{i0}(l_j) = 1$), the route is not feasible for state S_j ($\gamma_i^c(S_j) = 1$). The traffic flowing on this route has to be rerouted using other feasible routes.
- The constraints associated with a single route $r_i^c(S_j)$ affected by the failure of link $j \in [1, L]$ (i.e., $\gamma_i^c(S_j) = 1$) are

$$\theta_i^c(S_j) \leq B\gamma_i^c(S_j) \quad \forall c \in \mathcal{C}, \forall i \in |R^c(S_0)|, \quad \forall S_j \in \{\mathcal{S} - \mathcal{S}_0\} \tag{10}$$

$$\theta_i^c(S_j) \leq d_i^c(S_0) \quad \forall c \in \mathcal{C}, \forall i \in |R^c(S_0)|, \quad \forall S_j \in \{\mathcal{S} - \mathcal{S}_0\} \tag{11}$$

$$\theta_i^c(S_j) \geq d_i^c(S_0) - B\left[1 - \gamma_i^c(S_j)\right] \quad \forall c \in \mathcal{C}, \quad \forall i \in |R^c(S_0)|, \forall S_j \in \{\mathcal{S} - \mathcal{S}_0\}. \tag{12}$$

  Constraints (10)–(12) evaluate the total traffic associated with each OD pair traversing routes affected by the link failure. This amount of traffic must be dropped from the routes in S_0 affected by the failure and rerouted using other *feasible* routes. The model decides whether to keep the routes in S_0 unaffected by the failure or to select new feasible ones in the current failure state. Note that if $\gamma_i^c(S_j) = 0$ then $\theta_i^c(S_j) = 0$, otherwise $\theta_i^c(S_j) = d_i^c(S_0)$.
- The constraints associated with OD pair $c \in \mathcal{C}$ affected by failure of link $j \in [1, L]$ are

$$\sum_{i\in|R^c(S_j)|} d_i^c(S_j) = \sum_{k\in|R^c(S_0)|} \theta_k^c(S_j) \quad \forall c \in \mathcal{C}, \quad \forall S_j \in \{\mathcal{S} - \mathcal{S}_0\} \tag{13}$$

$$\alpha_i^c(S_j) \leq B \sum_{k\in|R^c(S_0)|} \theta_k^c(S_j) \quad \forall c \in \mathcal{C}, \quad \forall i \in |R^c(S_j)|, \forall S_j \in \{\mathcal{S} - \mathcal{S}_0\} \tag{14}$$

$$\sum_{i\in|R^c(S_0)|, h\in|R^c(S_j)|} \left(\alpha_i^c(S_0) + \alpha_h^c(S_j) v_{ih}^c(S_0, S_j)\right) \leq \mathcal{M} \quad \forall c \in \mathcal{C}, \forall S_j \in \{\mathcal{S} - \mathcal{S}_0\} \tag{15}$$

$$d_i^c(S_j) \leq d^c \alpha_i^c(S_j) \quad \forall c \in \mathcal{C}, \forall i \in |R^c(S_j)|, \forall S_j \in \{\mathcal{S} - \mathcal{S}_0\}. \tag{16}$$

  Constraint (13) defines the total amount of traffic associated with OD pair c that is affected by the failure of link l_j. This amount of traffic has to be rerouted using other feasible routes in this failure state. Constraints (13) and (14) define which feasible routes for the failure state S_j are to be used and how much traffic is to be assigned to each. Constraint (15) upper bounds the maximum number of routes ($\mathcal{M}$) that can be used by each OD pair in a given network state. Note that if none of the routes associated with OD pair c, ($r_i^c(S_0)$), are affected by failure S_j, then no traffic has to be rerouted (i.e., $d_i^c(S_j) = 0$) and the OD pair does not need to use new routes for this state (i.e., $\alpha_i^c(S_j) = 0$). Otherwise the traffic affected ($\sum_{i\in|R^c(S_j)|} d_i^c(S_j)$) has to be rerouted and the model has to choose new routes.
- The constraints associated with load on each link in the failure state S_j with $j > 0$ are

$$\beta^l(S_j) = \beta^l(S_0) - \sum_{c\in\mathcal{C}, i\in|R^c(S_0)|} \theta_i^c(S_j) u_{i0}^c(l) + \sum_{c\in\mathcal{C}, i\in|R^c(S_j)|} d_i^c(S_j) u_{ij}^c(l) \quad \forall l \in \mathcal{L}, \forall S_j \in \{\mathcal{S} - \mathcal{S}_0\} \tag{17}$$

  where constraint (17) defines the load on each link in each failure state S_j. The load of each link l is modified starting with the load in S_0. Traffic crossing link l for routes affected by failure S_j is subtracted out, and traffic crossing link l associated with the new routes in failure state S_j is added in.

The general routing problem uses the same objective function as METL [refer to (1)] without the third term $\Delta(S_0)$. Recall that in Section III-A we stated that the only routes considered feasible (to search over in optimization) are those that meet the SLA constraint. Thus we have essentially prefiltered out routes that violate SLA constraints and hence do not need the third term in our objective function. The objective function for this optimization problem is to minimize $\mathcal{F}$, where

$$\mathcal{F} = (1 - W)\mu(S_0) + W\mu(S_{wst}). \tag{18}$$

IV. Search Heuristic for Link Weight Selection

A. Overview of Tabu Search

Our heuristic is based on the Tabu search (TS) [7] methodology. TS is an iterative procedure designed to solve optimization problems. It is based on selected concepts that unite the fields of artificial intelligence and optimization. It has been applied to a wide range of problems such as job scheduling, graph coloring and network planning, and is considered an alternative to techniques such as simulated annealing and genetic algorithms.

TS conducts a guided exploration of the space of admissible solutions, keeping track of all solutions evaluated along the way. The exploration starts from an initial solution that is generally obtained with a greedy algorithm. When a stop criterion is satisfied, the algorithm returns the best visited solution. To move from one solution to the next, TS explores the neighborhood of the last solution visited (referred to as the current solution). It generates a neighbor solution by applying a transformation, called a *move*, on the current solution. The set of all admissible moves (determined by the rules guiding TS) uniquely defines the *neighborhood* of the current solution. At each iteration of the TS algorithm, all solutions in the neighborhood are evaluated, and the best is selected as the new current solution.

TS occasionally moves from a better solution to a poorer one during the search procedure. This is in contrast to local search heuristics which move only towards better solutions but that may get stuck in local minima. Note that, in order to efficiently explore the solution space, the definition of neighborhood may change during the exploration; in this way it is possible to achieve an *intensification* or a *diversification* [7] of the search in different solution regions.

A special rule, the *Tabu list*, is introduced in order to prevent the algorithm from deterministically cycling among solutions already visited. The Tabu list stores the last accepted moves; while a move is stored in the Tabu list, it cannot be used to generate a new move. The choice of the Tabu List size is very important—too small a size may cause a periodic repetition of the same solutions, while too large a size may severely limit the number of applicable moves, thus preventing a good exploration of the solution space.

B. Tabu Search for Link Weight Assignment

The fundamental elements of our link weight assignment heuristic are described below.

Precomputation Step: The time to find the optimal solution grows exponentially with the size of the search space, thereby making the search procedure very slow. Before running the Tabu search, we use an approximation to speed up the search procedure by reducing the size of the search space. Each solution in the search space consists of a set of routes for all traffic demands. We choose a criterion to filter out a subset of possible routes. This filtering is based on a hop-count threshold. Only routes whose hop-count is less than the threshold are deemed admissible and considered during the search procedure. This approximation is based on the intuition that Tabu search will avoid very long routes anyway because they consume network resources unnecessarily and will lead to long end-to-end delays. Long delays for OD flows, will in turn, lead to SLA violations. Moreover, it seems pointless to waste time searching such paths that do not have any serious potential as candidates. The hop-count threshold allows Tabu search to efficiently explore the search space with a reasonable computational overhead.

Care needs to be taken in selecting the hop-count threshold which will be topology dependent. On the one hand, choosing a small value for the threshold will eliminate many routes from consideration, including potentially the optimal solution. On the other hand, a large value of the threshold may result in a very large search space thus slowing down the run-time performance of the algorithm. We explore this tradeoff in Section VI and explain our choice of hop-count threshold.

Initial Solution: We set the initial weight of each link to be the inverse of the link capacity—a common recommendation of router vendors [16].

Moves and Neighborhood Generation: Each move corresponds to perturbing one or more of the weights in the current weight set. The first step in a move is to run Dijkstra's SPF algorithm for the current weight set, generate all minimum-cost routes and compute the traffic load for each link. We then identify two sets of links—those whose loads are within a small percentage of the maximum load (heavily loaded) and links whose loads are within a small percentage of the minimum load (lightly loaded). A link is selected at random from the heavily loaded set and its weight is increased (in order to divert traffic to other paths and reduce its load). Then, a link is selected at random from the lightly loaded link set and its weight is decreased. The goal is to attract traffic towards this link and potentially reduce the load on other, more heavily loaded, links. A new neighborhood is designed by repeating the above procedure.

Evaluation of Each Solution: Each iteration consists of generating a new neighborhood and selecting the best solution in the neighborhood. Every solution in the neigborhood is evaluated as follows. Minimum cost routes are computed using the SPF algorithm and the traffic load on each link is determined for a given traffic matrix. This evaluation is performed when there is no failure in the network *and* for each possible link failure. Then the objective function in (1) is computed.

Diversification: This step is needed to prevent the search procedure from indefinitely exploring a region of the solution space with only poor quality solutions. It is applied when there is no

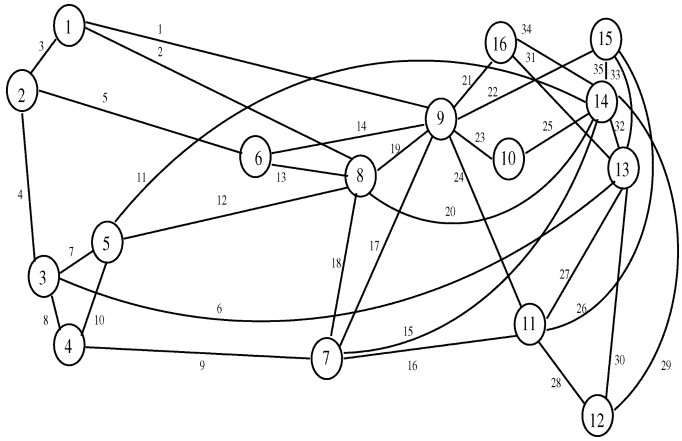

Fig. 1. PoP-level North American Sprint IP backbone.

improvement in the solution after a certain number of iterations. The diversification is a modification of the move described earlier. For a regular move, only one link, from each of the heavily and the lightly loaded link sets, is chosen at random. For a diversification move, several links are picked from each set. The weights of the selected links from the heavily loaded set are increased while the weights of the selected links from the lightly loaded links are decreased. This diverts the search procedure to a rather different region of the solution space where it resumes the process of neighborhood generation and solution evaluation.

Tabu List: The Tabu list serves to remember the most recent moves made, and consists of the links whose weights have been changed as well as the amount of increase/decrease applied to the corresponding link weight.

Stop Criterion: The search procedure stops when a fixed pre-determined number of iterations is reached. The number of iterations is defined based on the size of the network, the computational time needed, and the quality of solution desired.

V. Experimental Configuration

To evaluate our algorithm, and its impact on link loads and SLA guarantees, as well as its performance under failures, we first need to select topologies and traffic matrices on which to assess our heuristic. We describe these now, as well as our choice of hop-count threshold for limiting the search.

A. North American and European Sprint IP Backbones

For IP backbones, the failure of long-haul inter-PoP links between cities is significantly more critical than link failures within a PoP, since every PoP has a highly meshed topology. Accordingly we consider two representative PoP-level topologies based on the North American Sprint IP backbone, consisting of 16 nodes (each corresponding to a PoP) and 68 full-duplex links (Fig. 1) and the European Sprint IP backbone, consisting of 12 PoPs and 18 full-duplex links (Fig. 2). We emphasize that METL does not rely on the properties of any specific network topology and is therefore equally applicable to any topology or graph. According to current practices at Sprint, links are typically assigned integer weights in the range of 5 and 255 for the North American topology and in the range of 5 to 80 for the European topology. We will compute link weights in the same range using our algorithm. For ease of readability, we will sometimes refer to Sprint's North American backbone as the "NA topology," and Sprint's European backbone as the "Europe topology."

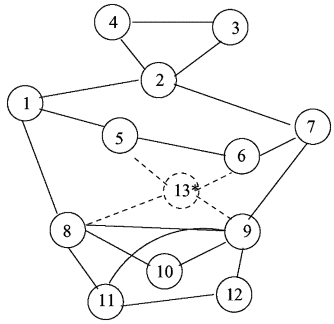

Fig. 2. PoP-level European Sprint IP backbone.

B. Traffic Matrices

The second input to a link weight selection algorithm is the traffic demands between origin-destination node pairs, represented by a traffic matrix. Unfortunately, little measurement data has been collected on PoP-to-PoP traffic matrices for the North American Sprint IP backbone. To evaluate our heuristic on the NA topology, we therefore use synthetic traffic matrices, based on models derived from OD flow characteristics that have been observed [13]. For the Europe topology, we do have an exact traffic matrix at our disposal that is described below. First, we state our two models used for synthetic traffic matrix generation.

- *Gravity Model (GM):* This form of the traffic matrix is based on the findings in [13] about the characteristics of PoP-to-PoP traffic matrices in Sprint's IP backbone. The volume of traffic from node i to node j is selected as follows:

$$d_{i,j} = O_i e^{V_j} / \sum_{j \in [1,V]} e^{V_j} \quad \forall i \in [1, V] \tag{19}$$

where O_i is the total traffic originating at node i, given by

$$O_i = \begin{cases} \text{uniform}(10, 50), & \text{if } prob \in [0, 0.6) \\ \text{uniform}(80, 130), & \text{if } prob \in [0.6, 0.95) \\ \text{uniform}(200, 300), & \text{if } prob \in [0.95, 1] \end{cases}$$

and $prob$ is a uniform random variable between 0 and 1. $V_j / \sum_{j \in [1,V]} e^{V_j}$ is the share of traffic originating at node i that is destined to node j. The V_j is a random number picked according to a uniform distribution between 1 and 1.5. The idea is to create three kinds of traffic demands by volume [1]. The fan-out of traffic originating at a given PoP is then determined by (19) in accordance with the observations in [13].

- *Negative Exponential (NegExp):* Each entry in the matrix is generated according to a negative exponential distribution with mean 40 Mbps. This is one of the several other forms of traffic matrices that we evaluated in order to validate the generality of our results. We choose this type of traffic matrix because it closely approximates the distribution of a subset of traffic demands seen in the Sprint network [1].

For Sprint's European IP backbone, we collected netflow data for roughly a three month period. Netflow was enabled on all the incoming links from gateway routers to backbone routers. The version of Netflow used is called Aggregated Sampled Netflow and deterministically samples 1 out of every 250 packets. Netflow samples fine granularity flows defined by the so called 5-tuple in IP packet headers. Using local BGP tables and topology information we were able to determine the exit link for each incoming flow. The resulting link-by-link traffic matrix is aggregated to form a PoP-to-PoP traffic matrix. Traffic matrices exhibit strong diurnal patterns, and although they vary a lot throughout the day, we are seeking link weights that can be static for long periods of time (i.e., even though short-lived failures). In order to ensure that our link weights can perform well under any traffic conditions, we consider the worst case traffic demand scenario. We thus extracted the peak hour (12–1 pm) of our traffic matrix to be used as input to our problem.

C. Reducing the Size of the Search Space

Recall that in our precomputation step (Section IV) we eliminate candidate routes for each OD pair whose hop-count is above a threshold. We now describe our choice for this threshold parameter that influences the sizes of the search space. The choice of this threshold depends on the characteristics of the network topology. The hop-count threshold T has to be larger than the longest minimum-hop path for any origin-destination pair across all network states $S_j, j \in [0, L]$. Otherwise, there will be no admissible route between some origin-destination pair(s) in one or more network states. By counting the hop-counts for all routes in the Sprint network, we found this value to be 4; hence the hop-count threshold has to be *at least* 4.

Fig. 3 illustrates the tradeoff between the computational overhead of METL and the quality of the solution for the North American network with different values of T. Link weights are selected without considering failures ($W = 0$). We have verified that the behavior is similar for other values of W. The top graph shows the number of iterations needed by METL to reach the final solution. The number of iterations grows exponentially fast—from 2000 for $T = 4$ to 16000 for $T = 12$. This shows that the parameter T is necessary for enabling METL to efficiently explore the search space with a reasonable computational overhead.

The bottom graph of Fig. 3 plots the maximum link load in network state S_0 for the link weight set chosen by METL. As expected, the quality of the solution improves as the threshold grows. The problem with small threshold values is that most routes are eliminated from consideration, and having few routes to choose from, METL yields a solution whose maximum link load is high. By increasing the threshold from $T = 4$ to $T = 8$, we see a large reduction in maximum link load from roughly 80% to 60%. On the one hand, we do not want too small a threshold as this can degrade the quality of the solution found by METL. On the other hand, we see that for values of $T > 8$, there is little further gain from increasing the threshold.

Fig. 4 explains the above observations. It shows the number of admissible routes per O-D pair for different values of the hop-count threshold (T). The number of admissible routes for each O-D pair is roughly 30 for $T = 4$ and roughly 200 for $T = 6$. We know from Fig. 3 that there exist good solutions among the many solutions being eliminated by such low thresholds. For $T = 8$, the number of solutions retained increases tenfold to about 1000. Consequently the maximum link load for the final solution found by METL could be reduced to 62%. For higher values of the threshold, the number of admissible routes increases at a slower rate—from 1000 for $T = 8$ to 3000 for $T = 12$. But at this point, the search space is so big that any further increase in its size causes the computation time of METL to grow exponentially. Based on these results, we pick the hop-count threshold to be 8.

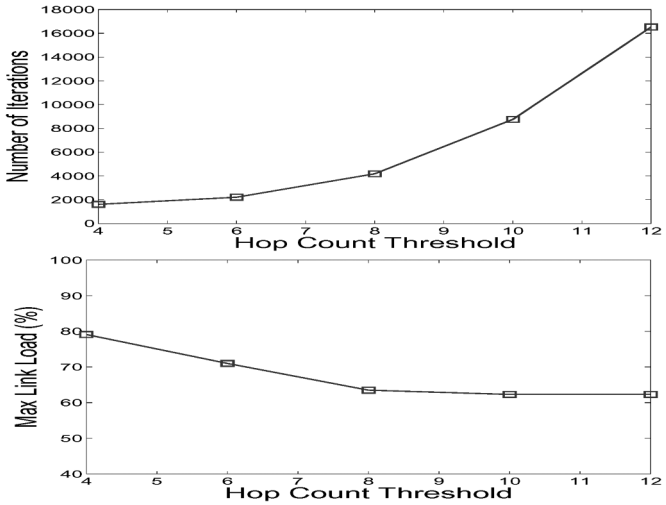

Fig. 3. Effect of hop-count threshold on number of iteration to reach final solution (top), and maximum link load for final solution (bottom). Case $W = 0$.

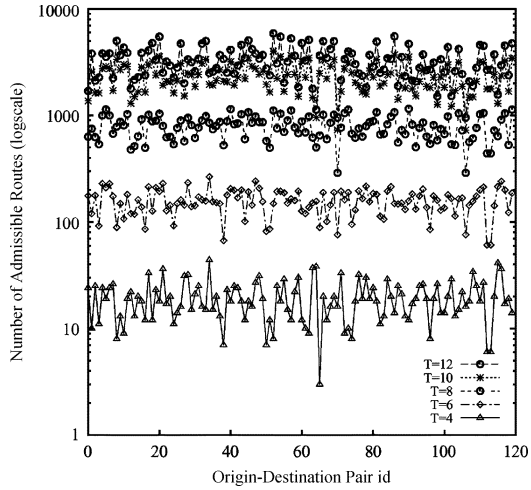

Fig. 4. Number of admissible routes per O-D pair for various T.

VI. Results

We analyze the performance of our link weight assignment heuristic using as follows. For each scenario, we choose a network topology and a traffic matrix as described in the previous section. Link weights are then computed using METL. Based on these weights, the minimum-cost routes for all source-destination traffic demands are determined and the resulting load on

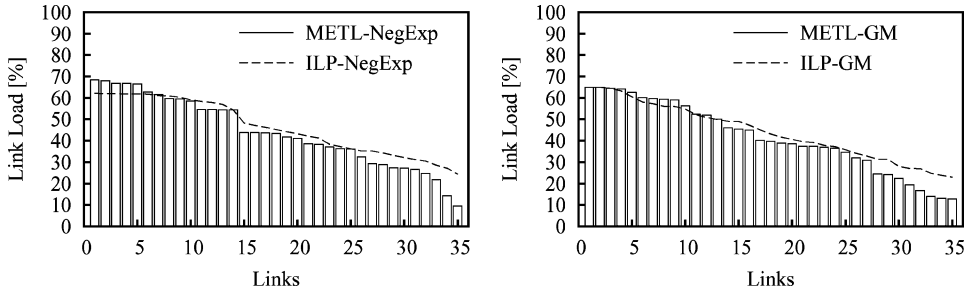

Fig. 5. Comparison of link loads for METL and ILP solutions with $W = 0$ and NA topology. (Left) NegExp traffic matrix. (Right) GM traffic matrix.

every link is calculated. We first validate our heuristic and then illustrate the benefits and tradeoffs of taking into account SLA requirements and single-link failures.

A. Validation of METL

We validate the quality of the METL solution by comparing it with the ILP model. We use the North American (NA) topology with both the NegExp and GM traffic matrices. For METL we find a set of link weights and compute shortest path routes and the load on each link. For the ILP model, we generate the set of optimal routes and then compute the load on each link. The ILP is solved using the CPLEX software package [12]. We consider a 45-ms SLA and $W = 0$, which corresponds to the traditional objective of minimizing maximum load in the absence of failures. Fig. 5 shows the link loads for METL and the ILP model for both types of traffic matrices. METL link loads are shown as bars while the ILP solution link loads are shown as a dashed continuous line. Links are ranked on the x axis by decreasing load under the METL solution.

The SLA bound was met by both the ILP and heuristic solutions, for both the NegExp and GM traffic matrices. In all cases the SLA achieved was 42 ms. For the NegExp traffic matrix, the ILP solution yields a maximum link load of 62% while METL's solution yields a maximum load of 68%—a difference of only 6%. In the case of the GM traffic matrix, the maximum load is the same for METL and the ILP. In fact, we considered 20 instances of each type of traffic matrix. All of the differences between the heuristic solution and ILP solution for a particular instance of traffic matrix, lay between 0–6%. The cases plotted in Fig. 5 are merely two examples chosen to illustrate the two extreme error bounds. METL performs well not only with respect to the maximum load criteria, but also on the distribution of the load across links. The distribution of link loads are quite similar for both cases of traffic inputs.

Recall that our heuristic solution uses the precomputation step of limiting the set of candidate routes by a hop-count threshold. The fact that we achieve near optimal results indicates that even though we filter out routes, we are not filtering out the best routes. We are thus benefiting from fast execution without paying any price in terms of solutions found.

Results for the no-failure case $(W = 0)$ are shown for a specific reason. We will subsequently evaluate the benefits of considering link failures in weight selection by comparing against weight selection with $W = 0$ as a baseline. Therefore, it is important to establish first that METL yields a solution of high quality for $W = 0$. However, we have performed this comparison for all values of W in [0,1] and found that the difference in the value of the objective function in (1) is less than 10% in all cases.

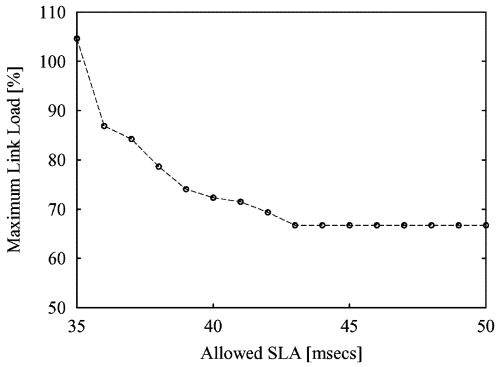

Fig. 6. Maximum link load versus SLA requirement; NA toplogy, $W = 0$, and GM traffic matrix.

We point out that METL is a fast performing algorithm; it can find a solution for a realistic network in less than 2 min on a dual-processor Intel Pentium computer with two 1.6-GHz CPUs and 1-GB RAM. The scalability of METL has been illustrated in [20].

B. Impact of Including the SLA Constraint

In this section we explore the impact of including an SLA requirement in our problem definition. We considered a range of potential SLA values and plotted the resulting maximum link load achieved for each SLA value in Fig. 6. In this example, we considered a GM traffic matrix and no failures $(W = 0)$. The SLA bound is considered *tighter* when it is lowered since lowering this value corresponds to providing improved performance to the customer.

We see that for SLA values above 43 ms, there is no change in the maximum link load of 67%. However, as the SLA is lowered below 43 ms, there is a steady increase in maximum load. This can be understood as follows. As the SLA is tightened, there are fewer and fewer alternate paths available to each OD flow. With less choices, many OD flows are going to get piled on the same link thus increasing its load (this could happen to multiple links). More specifically, we can say that the penalty function Δ does not play any role when the SLA is above 43 ms because there are no SLA violations. In this case the problem collapses into that of minimizing the maximum link load. However, as the

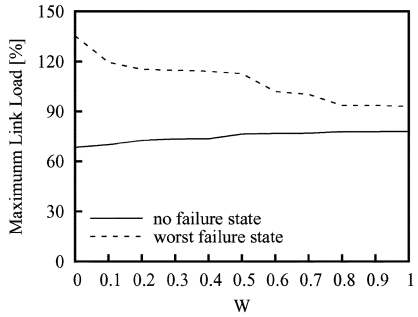

Fig. 7. Variation in maximum link load; NA topology and NegExp traffic matrix.

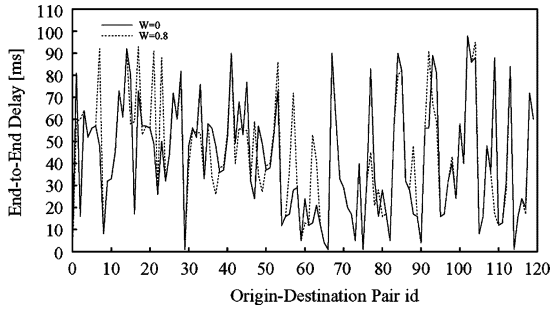

Fig. 8. End-to-end delays per origin-destination pair.

SLA offered is tightened to smaller values, then any routing configurations violating the SLA are penalized by the functional Δ, and the search for the optimal solution is restricted to solutions whose average end-to-end delay is close to the target SLA. As result, the maximum load that was 67% for a 43-ms SLA, increases to 104% if a 35-ms SLA is desired.

We were given the set of ISIS link weights used by Sprint in the NA-backbone and computed the SLA performance and link load conditions under no failures and all single link failures. Using our sample Sprint weights, we saw that the SLA could not be pushed below 43 ms without some SLA violations occurring. We point out that our Tabu search algorithm was always able to find a solution that did not violate the SLA bound when the SLA was in the range or 35–43 ms. Hence METL can be used to find an alternate set of weights, relative to what is used today, that would allow Sprint to offer its customers a lower SLA. This illustrates that today's solutions can be further improved upon. As Fig. 6 shows, however, the price to pay for pushing down the SLA is increased maximum link load.

Today Sprint's NA-backbone is designed to guarantee a lower SLA (43 ms) than what is claimed (45 ms). We see here an application of our algorithm beyond weight setting in which carriers can consider their ability to improve their service by reducing their SLA. Our algorithm serves as a network design aid because it can evaluate the impact of changing the SLA value and the tradeoffs incurred. For example, a carrier can ask: "what is the level of tradeoff I need to incur in order to lower my SLA to say 40 ms?" For the GM traffic matrix, METL was able to find an ISIS weight assignment that guarantees a 40-ms SLA at the cost of a 6% increasing in the maximum link load. For the NegExp traffic matrix, the 40-ms SLA was achieved at the cost of a 7% increase in the maximum link load. We further discuss such applications of a tool based on this algorithm in Section VII.

C. Impact of Considering Link Failures in Weight Selection

The extent to which we include failures in our weight choices is determined by the parameter W. We now consider the performance as W is varied. For the case of the NegExp traffic matrix, Fig. 7 shows the maximum link load for increasing values of W. For each value of W, we find a set of ISIS weights. The solid line shows the maximum load experienced by any link for each solution under normal operation when there are no failures. The dotted line shows the performance for the "worst failure state." By *worst failure state* we mean the following. Each point on the dotted line corresponds to one value of W. For each W, we consider all possible link failures. For each individual link failure, we examine the load on all links and record the worst one. Hence the worst failure state occurs when the link whose failure causes the maximum link load, over all possible link failures. A point on the dotted line corresponds to the maximum load over all failure states for a given W. Note that each value of W can have a different failure state that constitutes the worst one.

If we consider failure events within our optimization (i.e., $W = 1$), then the load performance bounds we achieve are given by the pair (*maximum load of 78% under normal, maximum load of 90% under failure*). If we ignore failures during weight selection the load performance bounds achieved are given by the pair (*maximum load of 68% under normal, maximum load of 135% under failure*). This indicates that performance during failures can be significantly improved by considering failures in link weight selection. This is achieved at the cost of a reasonable performance degradation in the no-failure state. Thus the gain in a failure mode is a 40% reduction in worst case load, while the disadvantage in normal mode is a 10% increase in worst case load. This illustrates another tradeoff in that improving performance under failure leads to a small degradation in performance when there are no failures.

We now consider the issue of how to select a value for W. We see in Fig. 7 that the maximum link load in the worst-failure state stays the same between $W = 0.8$ and $W = 1$, while the maximum link load for the no-failure state increases only slightly for $0.8 \leq W \leq 1$. Thus there is no benefit to increasing W beyond $W = 0.8$. Clearly we do not want link loads to exceed 100% because that is when loss starts occuring. Moreover, carriers also do not want link loads to exceed 90% or 95% because for such load levels, queues build-up and delay increases can become significant. We therefore select $W = 0.8$ for the remainder of our study since for the NA network this produces a good performance tradeoff between the no-failure state and the worst-failure state (when the traffic inputs are according to the NegExp model). In general, the most suitable value of W will depend on the particular combination of topology and traffic matrix. It is therefore important to do the above analysis, when designing a network, instead of picking a random value in [0,1] or just defaulting to $W = 0$ or $W = 1$.

It is important to ensure that our approach of considering failures in weight selection does not degrade the SLA in the network. Fig. 8 shows the end-to-end delay for each origin-destination node pair for two sets of link weights—one chosen without

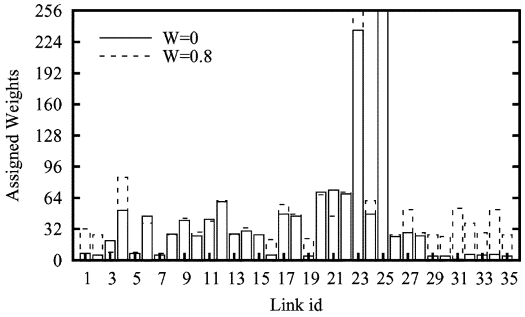

Fig. 9. Selected Link weights; NA topology, NegExp traffic matrix.

considering link failures (solid line) and the other chosen by considering link failures (dotted line). These delay values were computed using knowledge of link propagation delays in the NA network. We observe that for many of the OD pairs, there is no change in their end-to-end delay. We computed the mean and standard deviation of the delays for both $W = 0$ and $W = 0.8$. When failures are not considered, the mean delay was 42.22 with a standard deviation of 26.23; when failures were included, the mean delay was 42.93 with a standard deviation of 26.02. The maximum delay over all O-D pairs is the same in both cases (93 ms). Note that although these summary statistics for both solutions are very similar, some individual O-D flows do suffer when failures are included. For example, for O-D pair 7, the delay increased from 48 to 92 ms when link failures are considered in weight selection. Roughly 7% of our O-D flows experienced a significant increase in their end-to-end delay. However since the *average* end-to-end delay was not impacted, we can say that including failures in our problem did not degrade the SLA.

In order to gain some insight into how our algorithm works, we look at two examples to see how the weights for critical links are being manoeuvred. We start by looking at Fig. 9 that shows the link weights for $W = 0$ in solid bars, and $W = 0.8$ in dashed bars. Links 23 and 25 are assigned very large weights for both $W = 0$ and $W = 0.8$. By looking at Fig. 1, we see that link 23 connects nodes 9 and 10 while link 25 connects nodes 10 and 14. These two links are the only ones connecting node 10 to the rest of the network. When one fails, the other has to absorb all the load exchanged between node 10 and the rest of the network. For $W = 0$, METL does not consider this failover scenario when selecting weights. But it assigns large weights to these links in order to ensure that transit traffic (i.e., traffic not destined to node 10) is diverted away from these links. All traffic that either ingresses or egresses the network at node 10, only has two (fully disjoint) paths to chose from, and so these links must be used for that traffic, while transit traffic is discouraged. With $W = 0.8$, METL considers the failover scenario and further increases the weight of link 23 by another 30 units in order to mitigate overload in the event of failure of either link 23 or 25. Note that the weight for link 25 cannot be further increased because it is already at the maximum allowed value.

Consider a second example. Node 1 is connected to the rest of the network via three links: 1, 2, and 3. For $W = 0$, METL assigns weights 7, 5, and 20 respectively to links 1, 2, and 3. In the absence of failures, the loads on these links are well-balanced—67%, 68% and 43% respectively. When link 1 fails, traffic between node 1, and 13 of the other 15 nodes, fails over to link 2 resulting in a load of 135%. Link 2 is thus the link that experiences the maximum load for any single link failure.

When METL considers link failures ($W = 0.8$), it actually plans ahead for the possible failure of link 2 and assigns weights 32, 26, and 8 to links 1, 2, and 3 respectively. When link 1 fails, traffic between node 1, and 8 of the 15 other network nodes, fails over to link 2, while the traffic between node 1 and the rest of the nodes fails over to link 3. This balances the failover traffic on the two alternate paths and reduces the impact of failure on any single link.

In summary, our findings in this section are that there is a tradeoff in considering failures, namely that the maximum link load during normal operations increases in order to reduce the maximum load during failure episodes. We showed that this has no impact to the SLA as it is currently defined (as an average), and only a few OD pairs suffer in terms of delays when failures are considered. Despite these tradeoffs incurred by incorporating failure scenarios into the optimization problem for IGP link weight selection, we believe that the disadvantages are small. More importantly, the frequency with which failures occur (as discussed in Section I) simply mandates a solution that is robust to frequent short-lived failures.

D. Performance in Sprint's European IP Backbone

We now consider the European Sprint IP backbone. Conversely to the NA-backbone, this topology is characterized by a large diversity of link speeds. Most of the links are high-speed (OC192 and OC48), but a few low-speed links (OC12) play a crucial role in case of transient link failure scenarios. We remind the reader that we use a real peak-time traffic matrix (from this backbone) for our evaluation.

We compare the weight assignment found by METL against those used today in the Sprint-Europe network. We found that the set of link weights used today work quite well. They satisfy the 20-ms SLA, achieve a maximum link load of 49.4% under normal operation and spread the load rather evenly across links. When there are transient failures, the worst case load experienced is 70%. The solution found by METL obtained similar performance in terms of the SLA bound and the maximum load under failure. The fact that we used a real traffic matrix on a real topology and arrived at an answer close to the one currently being used in today's Internet backbone is a good confirmation that our method: 1) produces practical and feasible solutions in the right range and 2) contains the right set of constraints reflecting a carrier's needs.

It is interesting to ask whether METL can provide any further assistance to a network seemingly well designed by hand. Of the three metrics, SLA offered, maximum load under no-failure conditions and maximum load under failure conditions, we found that METL was able to improve upon one of them, namely the maximum load under no failure conditions. By using the set of weights obtained from METL, this maximum load can be reduced from 49% to 35%, while maintaining the same SLA and maximum load under failure as the deployed solution. This is a substantial gain. It is clear by looking at the ensemble of load levels in the left plot of Fig. 10 that METL

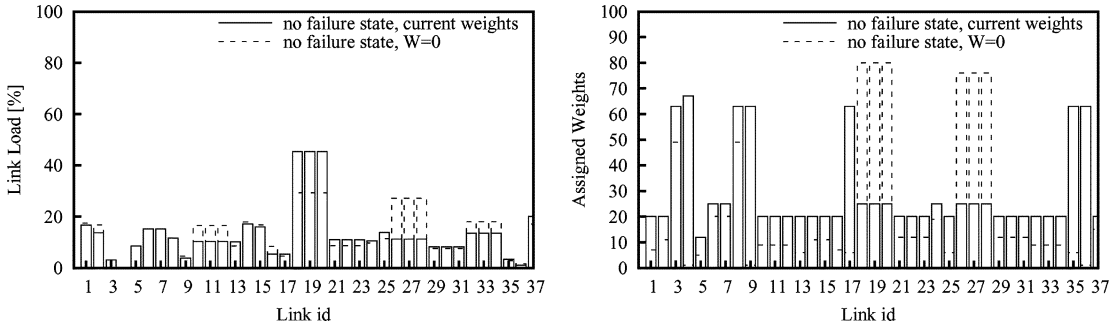

Fig. 10. Deployed weight settings versus METL weight selection. Europe topology (20-ms SLA), measured traffic matrix.

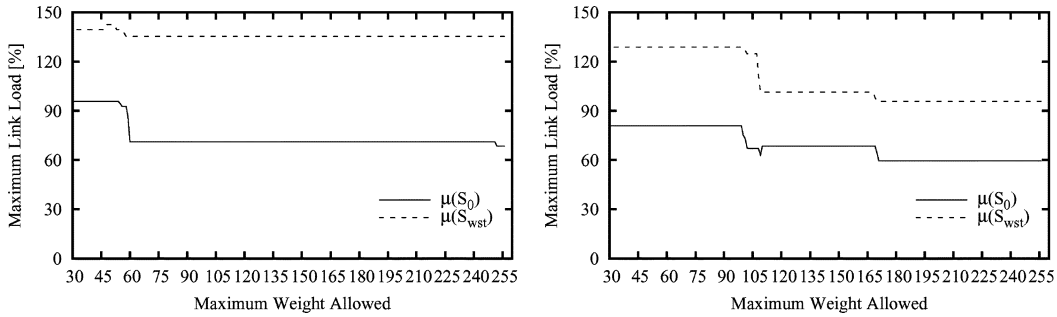

Fig. 11. Variation in maximum link load with maximum link weight allowed; NA topology. (Left) NegExp traffic matrix. (Right) GM traffic matrix.

can also improve upon the current load distribution in today's network. This comes from the fact that METL assigns higher weights to links 18, 19, and 20 (that are low-speed links) than the current solution (see right plot in Fig. 10).

VII. Applications

An algorithm that selects link weights can also be used for other traffic engineering tasks. We now show how an automated tool, such as METL, can be used by operations personnel to assess the impact of design choices (such as maximum allowed link weight) and for tasks such as topology planning and service design. Topology changes lead to changes in routing which in turn lead to change in network performance metrics. An application tool using METL as its engine, can rapidly asses the impact on performance metrics of a topology change through its impact on routing. We use Sprint's networks in a case study to illustrate these applications.

A. Selecting the Range of Allowable Link Weights

The original specifications of IS-IS allowed for link weights in the range of [0,63] [16]. Recent modifications to the protocol have increased this range to $[0, 2^{24}-1]$. Network operators typically choose weights in [0, 255]. But the tradeoffs of increasing or decreasing the range of link weights are poorly understood. For example, a carrier might want to know by how much they have to increase the maximum link weight before any gain is realized from the broader range. By allowing an operator to change the maximum weight value as an input parameter to a tool, they can explore this question.

We examine the case when the maximum weight allowed is successively increased. Fig. 11 shows how the maximum link load varies with the allowable range of link weights in the Sprint network for the NegExp and GM traffic matrices. In each graph, the solid line shows the maximum link load in the absence of failures and the dashed line shows the maximum load under any single link failure. Interestingly, the maximum link load does not decrease uniformly with an increase in the maximum weight value allowed. Increasing the maximum weight has no impact over a large range of values and then suddenly we see a sharp change. It is surprising that this curve behaves close to a decreasing step function. For example, in the case of the NegExp traffic matrix, the sharpest reduction happens around a maximum weight of 60. After that, there is no reduction until a value of 254 is reached, implying that if a carrier wants to further reduce the maximum link load beyond what is achievable with a maximum weight value of 60, they will need to increase the weights all the way up to 254. We suspect that, in general, the particular value (e.g,. 60) at which these sharp transitions occur will depend upon the topology and the traffic mix. We limit our conclusions here to the observation that the impact of the maximum weight value on maximum link load is that the maximum load value decreases in broad discrete steps for increasing maximum weight value. This insight can be useful for operators when they decide to alter the range of link weight values used by their routing protocols.

B. Aiding Topology Design

In this section we show how a network operator can use the output of our weight selection algorithm to flag potential troubles or limitations in the topology design of a network. By examining the set of weights selected, as in Fig. 9, an operator can observe that two links have weight settings that are more than three times the value of any other link and are also near the maximum allowed. This suggests that those links are protecting

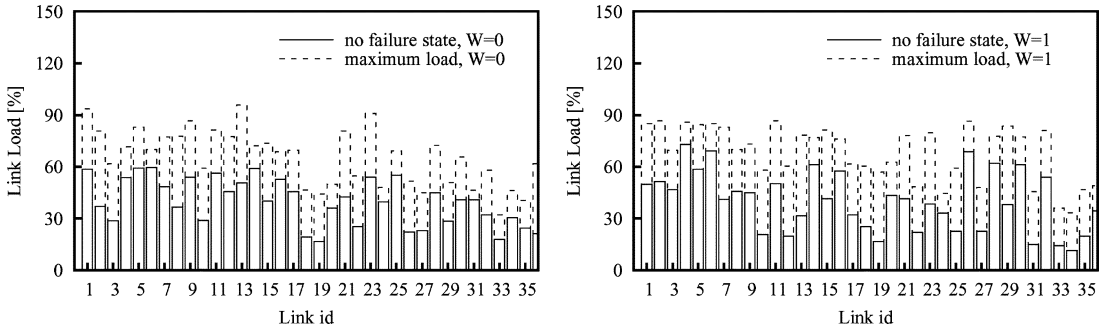

Fig. 12. Impact of adding a new link in NA toplogy; GM traffic matrix. (Left) $W = 0$. (Right) $W = 1$. The solid bars show the loads in the absence of failures. The dashed bar for each link shows the maximum load on that link for any link failure.

a path for one of their end-nodes. This is because by setting the weights so large, most nodes (except the immediate neighbors of those links) are discouraged from using the link since using it would yield a high-cost path. Some paths may be "protected" or "saved" for particular users in this way if there are few (or no) alternate paths. Viewing the ensemble of all link weight settings can thus serve as a warning that there may be limitations due to the topology in terms of how many paths are available to a given O-D pair, either during normal operation or during failure. Such a topology limitation can affect the maximum load level(s) and even the best of routing schemes cannot overcome a topology limitation.

This warning is indeed true in our topology. We can see in Fig. 1 that links 23 and 25 are node 10's only connections to the rest of the network. When one fails, the other must absorb all of the fail-over traffic. Let's revisit the righthand of Fig. 11. At the maximum weight value of 255, the maximum load was 60% under normal conditions and 92% under failures. Looking inside our computations, we saw that the maximum link load over all failures occurs when link 25 fails. Indeed, when this happens, link 23 becomes the most heavily loaded link. It is impossible to do anything about this because there are no alternate paths. Hence, even though the results in Fig. 11 were computed for $W = 0$, in this case considering $W > 0$ would not help and thus would not alter these maximum loads. This is an important point that illustrates that considering failures does not help when there are no alternate paths.

A tool such as ours, allows an operator to rapidly assess the hypothesis that there is a topology limitation. We can do this simply by adding a link into our topology, near the problematic portion of the graph, running the tool to compute the new set of link weights, and to compute the resulting worst case loads during failure and normal operation. A tool that is automated and quick, allows a network designer to consider numerous topology changes and receive fast feedback, along with specific performance metrics quantified, regarding the impact of such a topology change.

As an illustration, we did this by adding a link between nodes 10 and 16 in the NA topology. The link weights that METL assigns in this new topology are shown in Fig. 13. We see that a second set of links (26, 29,30,31,32, and 34 for $W = 1$) are now being given large weights as well. Fig. 12 shows the distribution of link loads, for this modified topology with the GM traffic matrix, for two sets of link weights (corresponding to $W = 1$ and $W = 0$). Note that if we do not consider failures ($W = 0$), the maximum load under normal conditions (60%) and the maximum load under the worst failure (92%) are the same as in the original topology. However if we do consider the failure states ($W = 1$), then the maximum load under failures can be reduced to 87%. This is possible since there are now two alternate links in the event that link 25 fails, and METL is able to select weights that balance the failover traffic between these two paths.

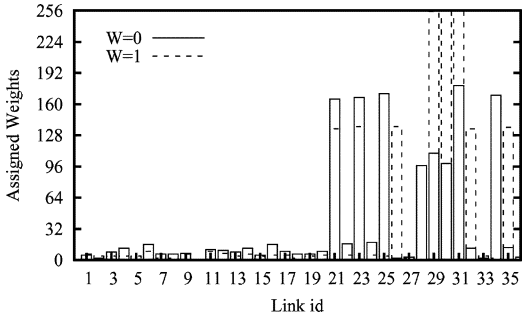

Fig. 13. Link weights selected for the GM traffic matrix.

Hence adding a link did not impact the performance in the no-failure mode, but instead it helped under failure episodes This implies that the topology limitation we found was one that had its impact not during normal operation but only during failures. That our tool helped reveal this illustrates a nice application of our tool.

C. Designing the SLA Offered

One of the key questions a carrier asks when designing a service is "what is the best SLA I can offer?"; Using the European topology and traffic matrix, we show how our tool can be used to answer this question.

First we consider the impact of adding a new PoP in the topology.[2] We added a new PoP (PoP id 13$*$ in Fig. 2) that is attached with four high-speed links to PoPs 2, 5, 6, and 7. The impact of this topology change on our three performance metrics is given in Table I. With this additional PoP we were able to reduce the SLA by 5 ms, from 20 to 15 ms. The link utilization

[2]It is possible to formalize another optimization problem to identify the optimal placement strategy. Our intent here is not tackle this problem, but rather to illustrate how our can aid operators in rapid assessment of the impact of topology design choices.

TABLE I
COMPARING TOPOLOGY DESIGNS THROUGH THEIR IMPACT ON ROUTING

Metrics	Original Topology	with extra PoP	extra PoP and upgraded links
SLA	20 ms	15 ms	15 ms
worst case load no-failure mode	49.4%	36%	38%
worst case load failure mode	70%	70%	42.1%

is reduced from 49.4% (Section VI-D) to 36% for the no failure scenario while a 15-ms SLA is simultaneously satisfied. The addition of a new PoP thus improved both the SLA and maximum load under normal operation. We can also see, however, that it did not improve the maximum load under failures.

This is most likely due to the heterogeneity of link types, and in particular the low-speed links. To test this hypothesis, we considered a second topology change, namely that of upgrading the two low-speed links by doubling their capacity. The maximum link load for any failure case dropped dramatically from 70% to 42.1%. This new routing configuration incurred a minor 2% degradation in maximum load for the no failure scenario. Using METL we are able to realize when optimization limits have been reached (e.g., due to current topology limitations) and evaluate "what if" scenarios (e.g., alternate topologies) quickly.

VIII. RELATED WORK

Despite the obvious importance of the IS-IS weight selection problem, a formal approach to the problem has been undertaken only recently. The first work to address this problem was [2], which showed that the problem of finding *optimal* IS-IS link weights is NP-hard with respect to many objectives. [2] proposes a local search heuristic to find weights for two objective functions. The first objective function is to minimize the maximum utilization across all links and the second one is to minimize a cost function that assigns every link a cost depending on its utilization. They show for both objective functions that their heuristic can compute weight sets that lead to solutions within a few percent of a theoretical lower bound. This bound is computed using linear programming much in the same way as we have done in this paper.

Some other authors have evaluated the use of other methods from operations research for the link weight assignment problem. Pioro *et al.* [5] and Harmatos [9] implement heuristics based on local search, simulated annealing, Lagrangean relaxation, and evolutionary algorithms. Furthermore, they propose a new method that uses a combination of simulated annealing and linear programming. This method is not guaranteed to find a weight set for every problem instance, but is shown to work in 95% of the cases. The authors find that in terms of the number of overloaded links and the degree of overload all methods perform similarly well. For larger topologies the new combined method is much faster than any of the other methods.

The use of linear programming duality was proposed in [19] to compute link weights. However, their method is limited to the case where unequal splitting is feasible. Neither OSPF nor IS-IS currently support unequal splitting, but only even splitting on equal cost paths.

All the above methods view the link weight assignment problem as a static problem ignoring the case of link failures. The first work to look at the problem of assigning link weights in the context of traffic matrix changes and link failures was [3]. This effort was mainly concerned with computing link weights that are robust to predicted periodic changes in the traffic demands. Instead of a single demand matrix, their heuristic picks weights based on several demand matrices. They show that it is possible to find a weight setting covering demand matrices dominated by convex combinations of the given matrices. They also briefly address the issue of link failure. They conclude that there are only a few links whose failure has a significant impact on the network. They show that changing even a single link weight in the event of a link failure can sometimes yield substantial performance improvements. Our target in this work was to avoid changing link weights for isolated failures that are transient (of which there are many).

The work in [15] addresses the problem of link weight selection with the goal of selecting weights that are robust to a wide variety of traffic matrices. This is important because traffic matrices are dynamic and evolve over time. They show that it is possible to find a single set of link weights that work nearly optimally under a wide variety of dynamic traffic demands. This result is encouraging in the same way that ours is, namely that it is possible to find a single set of weights that are robust to a variety of link failures. Another paper [4] addresses the problem of how to precompute backup routes around failures after a failure has been detected. This problem is unrelated to the problem we address in this paper.

IX. CONCLUSION

We extended the problem of link weight assignment to include the important practical requirement coming from SLA bounds that ISPs guarantee their customers. Our solution also addresses a second practical constraint, that of finding weights such that the network will perform well during transient failures. In this way, operations personnel avoid having to change weights for short-lived failure events, that are known to occur frequently. To the best of our knowledge this is the first work to consider transient link failures and SLA constraints for link weight selection.

We carried out validation from many angles. In comparing the performance of our heuristic to an optimal lower bound computed using an ILP formulation, we found that our solution was within 0–10% of optimal, over a variety of scenarios considered. We compared our solution to the link weights currently deployed today and found that we could meet the same performance metrics achieved today (delay and load under no failure conditions). This shows that our method yields practical solutions. We showed that we can improve upon today's solutions that were not designed to work well under failure. The maximum load on a link during any single-link failure can be reduced by as much as 50% at the cost of a 10% degradation in maximum load during no-failure modes. We illustrated that the SLA for the NA network, can be reduced from 45 to 40 ms at the cost of a small increase (6%) in the maximum link load during no-failure modes. In our case study, we used our tool to surmise that the inability to further reduce the SLA bound was not

due to a limitation of optimization but rather to a limitation in the topology. With the addition of a single extra node, we were then able to reduce the SLA from 20 to 15 ms for the European network.

In this work, we have focused on point-to-point traffic matrices for illustrative purposes and because this covers a large portion of an ISP's traffic. Since the time this work was completed, we have extended this work for the case of point-to-multipoint traffic matrices [20], since many destinations will have multiple egress points in a carrier's network.

We hope this paper highlights the importance of considering the interaction of routing with link failures and SLA requirements when considering the link weight selection problem. In the future, we would like to extend our work by considering multiple link failures. Finally, we hope that both the performance of our method and our case study will serve to motivate operations personnel to consider more seriously the use of automated tools for network design.

REFERENCES

[1] S. Bhattacharyya, N. Taft, J. Jetcheva, and C. Diot, "POP-level and access-link-level traffic dynamics in a Tier-1 POP," in *Proc. 1st ACM SIGCOMM Internet Measurement Workshop (IMW)*, San Francisco, CA, Nov. 2001.

[2] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. EEE INFOCOM*, Tel-Aviv, Israel, Mar. 2000.

[3] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing World," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 4, pp. 765–767, 2001.

[4] B. Fortz and M. Thorup, Fortifying OSPF/IS-IS against link failures. [Online]. Available: http://www.research.att.com/mthorup/PAPERS/papers.html Tech. Rep.

[5] M. Pioro, A. Szentesi, J. Harmatos, A. Juttner, P. Gajowniczek, and S. Kozdrowski, "On OSPF related network optimisation problems," in *Proc. IFIP ATM IP*, Ilkley, U.K., Jul. 2000.

[6] F. Giroire, A. Nucci, N. Taft, and C. Diot, "Increasing the robustness of IP backbones in the absence of optical level protection," in *Proc. IEEE INFOCOM*, Apr. 2003.

[7] F. Glover and M. Laguna, *Tabu Search*. Amsterdam, The Netherlands: Kluwer.

[8] S. Halabi, *Internet Routing Architectures*, 2nd ed. Indianapolis, IN: Cisco, 2001.

[9] J. Harmatos, "A heuristic algorithm for solving the static weight assignment optimisation problem in OSPF networks," in *Proc. Global Internet Conf.*, Dallas, TX, Nov. 2001.

[10] G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of link failures in a large IP backbone," in *Proce. 2nd ACM SIGCOMM Internet Measurement Workshop (IMW)*, San Francisco, CA, Nov. 2002.

[11] M. Krunz, B. Zhang, and C. Chen, "Stateless QOS routing in IP networks," in *Proce. Global Internet Workshop*, Dallas, TX, Nov. 2001.

[12] ILOG CPLEX. [Online]. Available: http://www.ilog.com/products/cplex/

[13] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: Comparisons and new directions," in *Proc. ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002.

[14] A. Nucci, B. Schroeder, S. Bhattacharyya, N. Taft, and C. Diot, "Link weight assignment for transient link failures," in *18th Int. Teletraffic Congr. (ITC)*, Sep. 2003.

[15] D. Applegate and E. Cohen, "Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs," in *ACM SIGCOMM*, Aug. 2003.

[16] D. Oran, "OSI IS-IS Intradomain Routing Protocol," IETF Request For Comments 1142.

[17] K. Papagiannaki, R. Cruz, and C. Diot, "Network performance monitoring at small time scales," in *ACM Internet Measurement Conf.*, Miami, FL, Oct. 2003.

[18] K. Papagiannaki, S. Moon, C. Fraleigh, P. Thiran, F. Tobagi, and C. Diot, "Analysis of measured single-hop delay from an operational backbone network," in *Proc. IEEE INFOCOM*, New York, Jun. 2002.

[19] Y. Wang, Z. Wang, and L. Zhang, "Internet traffic engineering without full mesh overlaying," in *Proc. IEEE INFOCOM*, Anchorage, AK, Apr. 2001.

[20] S. Agarwal, A. Nucci, and S. Bhattacharyya, "Measuring the shared fate of IGP engineering and interdomain traffic," in *IEEE Int. Conf. NP*, Nov. 2005.

Antonio Nucci (M'99–SM'05) received the Dr.Ing. degree in electronics engineering and the Ph.D. degree in telecommunications engineering from the Politecnico di Torino, Turin, Italy, in 1998 and 2002, respectively.

From 2001 to 2005 he worked in Sprint's Advanced Technology Laboratories in the IP research group. He is currently Chief Technology Officer at Narus Inc., Mountain View, CA. His research interests include traffic measurement, characterization, analysis and modeling, performance evaluation, security, and network design.

Supratik Bhattacharyya received the Ph.D. degree from the University of Massachusetts, Amherst.

He is a Distinguished Member of Technical Staff at Sprint Advanced Technology Laboratories, Burlingame, CA. He is broadly interested in Internet and wireless protocols and systems. His past work at Sprint has covered several aspects of core IP networks such as performance monitoring, routing, traffic engineering, and fault tolerance. His current interests are in mobile communication and services and in mining network traffic data.

Nina Taft (M'94–SM'06) received the B.S. degree fom the University of Pennsylvania, Philadelphia, and the M.S. and Ph.D. degrees from the University of California, Berkeley, in 1985, 1990, and 1994, respectively.

She is currently a Senior Researcher at Intel Research, Berkeley, CA. From 1995 to 1999, she worked at SRI International; and from 1999–2003, she was a member of the IP Group at Sprint Advanced Technology Laboratories. Her research interests lie in traffic characterization and modeling, performance evaluation, network design, and enterprise network security.

Christophe Diot received the Ph.D. degree in computer science from INP Grenoble, Grenoble, France, in 1991.

He was with INRIA Sophia-Antipolis from October 1993 to September 1998, Sprint (Burlingame, CA) from October 1998 to April 2003, and Intel Research (Cambridge, U.K.) from May 2003 to September 2005. He joined Thomson R&D, Boulogne, France, in October 2005 to start and manage the Paris Research Laboratory. His research activities focus on communication services and platforms of the future.

Dr. Diot is a Fellow of the Association for Computing Machinery.