

# Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks

Ashwin Sridharan<sup>†</sup>, Roch Guérin<sup>†</sup>, Christophe Diot<sup>††</sup>

{ashwin,guerin}@ee.upenn.edu , christophe.diot@intel.com

**Abstract**—Traffic engineering aims to distribute traffic so as to “optimize” some performance criterion. This optimal distribution of traffic depends on both the routing protocol and the forwarding mechanisms in use in the network. In IP networks running the OSPF or IS-IS protocols, routing is over shortest paths, and forwarding mechanisms distribute traffic “uniformly” over equal cost shortest paths. These constraints often make achieving an optimal distribution of traffic impossible. In this paper, we propose and evaluate an approach that can realize near optimal traffic distribution without changes to routing protocols and forwarding mechanisms. In addition, we explore the trade-off that exists between performance and the configuration overhead that our solution requires. The paper’s contributions are in formulating and evaluating an approach to traffic engineering in IP networks that achieves near-optimal performance while preserving the existing infrastructure.

**Index Terms**—Routing, Networks, Traffic Engineering, Aggregation.

## I. INTRODUCTION

As the amount and criticality of data being carried on IP networks grows, managing network resources to ensure reliable and acceptable performance becomes increasingly important. Furthermore, this should be accomplished while minimizing or deferring costly upgrades. One of the techniques that is being evaluated by many Internet Service Providers to achieve this goal is traffic engineering. Traffic engineering uses information about the traffic entering and leaving the network to generate a routing scheme that optimizes network performance. Most often the output of traffic engineering is an “optimal” set of paths and link loads that produce the best possible performance given the available resources. However, explicitly setting up such paths

and (optimally) assigning traffic to them, typically calls for changes to both the routing protocols and the forwarding mechanism they rely on, e.g., through the introduction of new technology such as MPLS [15].

Currently, two of the most widely used Interior Gateway routing protocols are OSPF [14] and IS-IS [4], and devising solutions that allow these protocols to emulate “optimal routing,” is therefore desirable. There are two main difficulties in doing so. The first is that these protocols use shortest path routing with destination based forwarding. The second is that when the protocols generate multiple equal cost paths or next hops for a given destination routing prefix, the forwarding mechanism equally splits the corresponding traffic across them to balance the load. This “equal” splitting is an approximation that depends on the selection of which next hop to use for a given packet. This selection is based either on the value (one for each possible next hop) of a hash function applied to the packet header (source and destination addresses and port numbers), or on the state of a simple round-robin scheme that cycles through the possible next hops. Although the latter option is occasionally used and provides for a more even distribution of load across possible next hops, the former option is the more commonly used in practice. This is because it preserves packet ordering within a flow, and because the large number of flows that modern routers handle results in a good approximation of equal splitting ([5]). For simplicity, in the paper we assume that traffic is split in exactly equal fractions across equal cost next hops.

The constraints imposed by the use of both shortest path routing and equal splitting across equal cost paths make it difficult or even impossible to achieve optimal traffic engineering link loads. One of the first works to explore this issue was [7], where a local search heuristic was proposed for optimizing

<sup>†</sup> Dept. of Elec. Eng., Univ. of Pennsylvania, Philadelphia, PA 19104.

<sup>††</sup> Intel Research, Cambridge, UK

The work of A. Sridharan and R. Guérin was supported in part by NSF Grant ANI-9902943

OSPF weights assuming knowledge of the traffic matrix. [7] showed that in spite of these constraints, properly selecting OSPF weights could yield significant performance improvements. However, the paper also showed that for some topologies, performance could still differ substantially from the optimal solution. Subsequently, a result from linear programming ([2][Chap. 17, Sec. 17.2]) was used in [21] to prove that *any* set of routes can be converted into a set of shortest paths based on some link weights that matches or improves upon the performance of the original set of routes. This establishes that the shortest path limitation is in itself not a major hurdle. However, the result of [21] assumes forwarding decisions that are specific to each *ingress-egress* pair, and more importantly, the ability to split traffic in an arbitrary ratio over different shortest paths. Both of these assumptions are at odds with current IP forwarding mechanisms.

Compatibility with destination based forwarding can be achieved through a simple extension to the result of [21], simply by taking advantage of a property of shortest paths ([17]), or by readjusting the program formulation itself ([1])<sup>1</sup>. Accommodating the constraint of uneven splitting of traffic across multiple shortest paths is a more challenging task. It is typically not supported with the forwarding path implementations that accompany most routing protocols (some limited support is available with Cisco’s proprietary routing protocol EIGRP<sup>2</sup>). A new protocol, OSPF-OMP, was proposed in [19] that circumvents this constraint by modifying the hash function in the router to allow unequal load balancing. However, this will require significant changes to the forwarding mechanisms in the router’s data path, which is typically not something that can be accomplished through a simple software upgrade.

The solution we propose leverages the fact that present day routers have thousands of route entries (destination routing prefixes) in their routing table. Instead of changing the forwarding mechanism responsible for distributing traffic across equal cost paths, we plan to control the actual (sub)set of shortest paths (next hops) assigned to routing prefix entries in the forwarding table(s) of a router. *In*

*other words, for each prefix we define a (sub)set of allowable next hops by carefully selecting this subset from the set of next hops corresponding to the shortest paths computed by the routing algorithm.* This allows us to control how traffic is distributed *without* modifying routing protocols such as OSPF or IS-IS, and without requiring changes to the data path of current routers, i.e., their forwarding mechanism. It does, however, require some changes to the control path of routers in order to allow the selective installment of next hops in the forwarding table.

Our initial focus is on gaining a better understanding of how well the selective installment of next hops for different routing prefixes can approximate an optimal traffic allocation (set of link loads). We show that this problem is NP-complete and hence one must resort to heuristics. Toward this end, we propose a simple local search heuristic for implementing the solution. We obtain a bound on the gap between the heuristic and optimal allocation, and also demonstrate its efficacy through several experiments. Even though we study the heuristic in the context of a routing problem, we believe that it is generic enough to be of potential use in other load balancing scenarios. The main finding from our investigation is that the performance achieved by this approach is essentially indistinguishable from the optimal.

This being said, an obvious drawback of “hand-crafting” the set of next hops that are to be installed for each routing prefix in a router’s forwarding table, is the configuration overhead it introduces. The potential magnitude (proportional to the size of the routing/forwarding table) of this overhead could make this approach impractical. As a result, our next step is to investigate a solution that can help mitigate this overhead, albeit at the cost of a possible degradation in performance. Specifically, we limit the number of routing prefixes for which we perform the proposed selective installment of next hops. Our results indicate that a significant reduction in configuration overhead can be achieved without a major loss of performance.

The rest of the paper is structured as follows. Section II introduces the linear program formulation used in [21] to generate an “optimal” set of shortest paths, and introduces the proposed modifications to make it compatible with existing IP routers. Section III presents a heuristic for approximating

<sup>1</sup>This is explained in detail in Section II-A

<sup>2</sup>Cisco’s specification of EIGRP allows unequal load balancing across shortest paths and non-shortest paths.

an optimal traffic distribution by manipulating the set of next hops assigned to each routing prefix. A performance bound is also derived (see Appendix). Section IV presents several experiments that first establish the efficacy of the heuristic of Section III, and then explore the impact on performance of lowering configuration overhead. Section V provides a brief summary of the paper's contributions and outline directions for future work.

## II. FROM OPTIMAL ROUTING TO SHORTEST PATH ROUTING

In this section, we first briefly review the classic result from linear programming [2][Chap. 17, Sec. 17.2] that was cast in the context of routing in communication networks in [21] to show how optimal routing can be achieved using only shortest paths. We then discuss why this result is not directly usable in current IP networks, and finally propose solutions that allow us to implement the result under the existing paradigm.

The network is modeled as a directed graph  $G = (V, E)$  with  $v = \|V\|$  routers and  $e = \|E\|$  directed links. We assume the existence of a traffic matrix  $T$  where entry  $T(s_r, t_r) = d_r$  denotes the average intensity of traffic entering the network at ingress router  $s_r$  and exiting at egress router  $t_r$  for commodity  $r \in \mathcal{R}$ . A good reference on how to construct such a traffic matrix can be found in [6]. Assume that an optimal allocation based on some network wide cost function yields a set of paths  $\mathcal{P}_r$  for each commodity  $r$  (ingress-egress router pair), so that the total bandwidth consumed by these paths<sup>3</sup> on link  $(i, j)$  is  $\tilde{c}_{ij}$ . It can be shown that the same performance, in terms of the bandwidth consumed on each link, can be achieved with a set of shortest paths by formulating and solving a linear program and its dual. The linear program can be formulated as ([21]):

$$\min \sum_{(i,j) \in E} \sum_{r \in \mathcal{R}} d_r X_{ij}^r$$

subject to

$$\sum_{j:(j,i) \in E} X_{ji}^r - \sum_{j:(i,j) \in E} X_{ij}^r = \begin{cases} 0, & i \neq s_r, t_r \\ 1, & i = t_r, \\ & r \in \mathcal{R} \end{cases}$$

<sup>3</sup>The bandwidth consumed on a link is assumed to be the sum of the traffic on all paths that use the link

$$\begin{aligned} \sum_{r \in \mathcal{R}} d_r X_{ij}^r &\leq \tilde{c}_{ij}, \quad (i, j) \in E \\ 0 \leq X_{ij}^r &\leq 1 \quad (i, j) \in E, \quad r \in \mathcal{R}, \end{aligned} \quad (1)$$

where  $X_{ij}^r$  is the fraction of traffic for commodity  $r$  that flows through link  $(i, j)$ . Solving the linear program gives a traffic allocation  $\{\tilde{X}_{i,j}^r\}$  that consumes no more than  $\tilde{c}_{i,j}$  amount of bandwidth on any link  $(i, j)$ . Note that the formulation is not dependent on the original network cost function used to obtain the paths  $\{\mathcal{P}_r\}$ . Instead it achieves the same performance by matching the desired bandwidths  $\tilde{c}_{i,j}$ . In order to obtain link weights for shortest path routing, the dual of the linear program as formulated in [21] needs to be solved:

$$\begin{aligned} \max \quad & \sum_{r \in \mathcal{R}, t \in V} d_r U_{tr}^r - \sum_{(i,j) \in E} \tilde{c}_{ij} W_{ij} \\ \text{subject to} \quad & U_j^r - U_i^r \leq W_{ij} + 1, \quad \forall r \in \mathcal{R}, (i, j) \in E \\ & W_{ij} \geq 0 \\ & U_{s_r}^r = 0. \end{aligned} \quad (2)$$

The dual gives a set of link weights  $\{1 + \tilde{W}_{i,j}\}$ , from which a set of shortest paths can be constructed.

It is however important to understand that although routing can now be done over shortest paths, this is still quite different from the *forwarding* paradigm currently deployed in existing OSPF and IS-IS networks. The reasons are two-fold and both can be traced to the output of the primal LP, Eqn. (1), namely, the traffic allocation  $\{\tilde{X}_{i,j}^r\}$ . They govern the fraction of traffic forwarded on each next-hop.

Firstly, observe that the traffic allocation is for each *commodity* or ingress-egress router *pair*. This means that the routing protocol could possibly generate different set of next hops for each ingress-egress pair *on which traffic is to be forwarded*. This would impact the forwarding mechanism on the data path, as the router would need to make decisions on the basis of both ingress and egress routers. Clearly, this is at odds with the current paradigm of destination-based forwarding.

The second problem relates to the fact that current forwarding mechanisms support only equal splitting of traffic on the set of equal cost next hops. The linear program yields a traffic allocation that is not guaranteed to obey this constraint. Modifying the

forwarding engine to support unequal splitting (see, for example, [19]) of traffic would involve changes to the data path, namely modification of the hash function to allow for controllable hash boundaries. In the next two sub-sections we suggest methods that overcome both these problems.

### A. Destination Based Aggregation of Traffic

The first problem of translating a traffic allocation that distinguishes between ingress-egress router pairs into one that only depends on egress point is relatively straightforward. It can be achieved simply by transforming the individual splitting ratios of ingress-egress pairs that share a common egress router into a possibly different splitting ratio for the aggregate traffic associated with the common egress router. The reason this is possible is because all chosen routes are shortest paths. Shortest paths have the property that segments of shortest paths are also shortest paths, so that once two flows headed to the same egress point meet at a common node they will subsequently follow the same set of shortest paths. This means that we need not distinguish between these packets based on their source address, and can make splitting and forwarding decisions simply based on their destination address.

The aggregation of flows headed to a common egress could either be done *after* the optimal traffic allocation for each commodity has been computed (for example by the LP in Eqn. (1)) or in the formulation of the linear program itself. The first method is presented in [17]. The second scheme was presented in [1] and illustrated below. Since traffic allocation is done based only on the egress router, we can aggregate all commodity flow variables headed toward a common egress point and then suitably modify the conservation constraints. A re-formulation of Eqn. (1) and Eqn. (2) in the destination aggregated form as presented in [1] is reproduced below.

The primal Linear Program is given by :

$$\min \sum_{(i,j) \in E} \sum_{t \in V} Y_{ij}^t$$

subject to

$$\sum_{j:(j,i) \in E} Y_{ji}^t - \sum_{j:(i,j) \in E} Y_{ij}^t = \begin{cases} -D^t, & i = t \\ d_r, & i = s_r, t = t_r \\ 0, & \text{otherwise} \end{cases}$$

$$\forall i \in V, t \in V \\ \sum_{t \in N} Y_{ij}^t \leq \tilde{c}_{ij}, \forall (i,j) \in E$$

where

$$D^t = \sum_{r:r \in \mathcal{R}, t_r = t} d_r.$$

(3)

Note that the variables of the Primal,  $Y_{i,j}^t$ , represent traffic on link  $i,j$  headed toward egress router  $t$ . Also, the flow conservation constraints have been modified to accommodate the aggregation, where  $D^t$  represents all traffic headed toward destination  $t$ .

The dual is given by :

$$\max \sum_{t \in N} \mathbf{P}^t \mathbf{U}^t - \sum_{(i,j) \in E} \tilde{c}_{ij} W_{ij}$$

subject to

$$U_i^t - U_j^t \leq W_{ij} + 1, \forall t \in N, (i,j) \in E \\ W_{ij} \geq 0, \quad \forall (i,j) \in E \\ U_t^t = 0, \quad \forall t \in N$$

(4)

where  $\mathbf{P}^t$  represents the right hand side array in the flow conservation constraints of the primal, Eqn. (3), that is :

$$P^t(i) = \begin{cases} -D^t, & i = t \\ d_r, & r : i = s_r, t = t_r \\ 0, & \text{otherwise} \end{cases}$$

and as before,  $\{1 + \tilde{W}_{i,j}\}$  still represent link weights for shortest path computation. The above formulation is not only conformal to the paradigm of destination based forwarding, but also reduces the number of variables by a factor of  $\|V\|$  through removal of information regarding ingress-egress router pairs. This greatly improves the computation complexity involved in obtaining an optimal solution and will be used henceforth throughout the remainder of the paper to compute the optimal traffic allocation and link weights.

### B. Approximating Unequal Split of Traffic

In the previous sub-section, we saw that solving the problem of source-destination based forwarding decisions was relatively straightforward. Unfortunately, the same does not hold for the uneven

splitting issue, and as mentioned earlier providing such a capability is a significant departure from current operations. Our proposal to overcome this problem is to take advantage of the fact that today’s routing tables are relatively large, with multiple routing prefixes associated with the same egress router. By controlling the (sub)set of next hops that each routing prefix is allowed to use, we can control the traffic headed toward a particular egress router(destination). In other words, instead of the current operation that has all routing prefixes use the full set of next hops, we propose to selectively control this choice based on the amount of traffic associated with each routing prefix and the desired link loads for an optimal traffic allocation.

The following example illustrates the idea behind the approach. Assume that at some node, there are four routing prefixes,  $r_1, r_2, r_3$  and  $r_4$  that map to a common destination  $d$  and have traffic intensities  $t(r_1) = 2, t(r_2) = 5, t(r_3) = 8$  and  $t(r_4) = 4$ . Let there be three shortest paths associated with destination  $d$ , so that the routing table has 3 possible next hops to  $d$ , and assume that the *optimal* distribution of traffic to the three next hops is  $f_1 = 6, f_2 = 4$  and  $f_3 = 9$ . We can then intuitively match this traffic distribution by the following next hop assignment:

$r_1 \rightarrow$  Hops 1, 3  
 $r_2 \rightarrow$  Hop 1  
 $r_3 \rightarrow$  Hop 3  
 $r_4 \rightarrow$  Hop 2

The resulting traffic distribution is  $f'_1 = (2/2 + 5/1) = 6, f'_2 = (4/1) = 4, f'_3 = (2/2 + 8/1) = 9$ , which matches the optimal allocation.

The advantage of the above approach is that the forwarding mechanism on the data path remains unchanged, as packets are still distributed evenly over the set of next hops assigned to a routing prefix. This means that a close approximation of an optimal traffic engineering solution might be feasible even in the context of existing routing and forwarding technologies. There are, however, a number of challenges that first need to be addressed. The first is the need for traffic information at the granularity of a routing prefix entry instead of a destination (egress router). This in itself is not an insurmountable task as most of the techniques currently used to gather traffic data, e.g., router mechanisms’ like Cisco’s Netflow or Juniper’s cflowd, can be readily adapted

to yield information at the granularity of a routing prefix.

The second issue concerns the configuration overhead involved in communicating to each router the subset of next hops to be used for each routing prefix. This can clearly represent a substantial amount of configuration data, as routing tables are large and the information that needs to be conveyed is typically different for each router. The approach we propose and study is to identify a small set of prefixes for which careful allocation of next hops is done and rely on default behavior for the remaining prefixes. The trade-off will then be in terms of how close one can get to an optimal traffic distribution, while configuring the smallest possible number of routing prefixes. We investigate this trade-off in Section IV, where we find that near optimum performance can often be achieved by configuring only a small number of routes (routing prefixes).

The third and last challenge is to actually formulate a method for determining which subset of next hops to choose for each routing prefix in order to approximate an optimal allocation. The goal of any solution will be to minimize some metric that measures discrepancy between the optimal traffic allocation and the one achieved under equal-splitting constraints on any hop. In this work, we use the maximum *load* on any hop as a measure of performance, where the *load* on a hop is the ratio of the allocated traffic to the optimal traffic. Details regarding the use of another metric, the *gap* between optimal traffic and allocated traffic can be found in [17].

### III. HEURISTIC FOR TRAFFIC SPLITTING

Ideally, one should consider the problem of selective next-hop allocation at the global level, that is, do a *concurrent* optimal assignment of next hops for *each* routing prefix at *each* node. However, even the single node allocation problem is NP-complete<sup>4</sup>, and hence may not be computationally tractable. Consequently, we propose heuristics that perform independent computations for each routing prefix at each node. These computations are based only on the incoming traffic at the node and the desired outgoing traffic profile. A potential problem with this approach is that the traffic arriving at a node

<sup>4</sup>See Appendix for proof.

may not match the optimal profile due to the heuristic decisions at some upstream node. Consequently, the profile of the outgoing traffic from the node in question, could further deviate from the desired one. However, in our experiments we observe that usually the heuristic is able to track the optimal load profile and hence incoming traffic seen at any node and the resultant outgoing traffic have a near-optimal profile. We have proposed three heuristics that are greedy in nature and try to minimize one of the two metrics mentioned in Section II-B. Since the heuristics are similar in nature, we shall limit our discussion to only one of them, namely the MIN-MAX LOAD heuristic, because in all our experiments, this heuristic performed the best. Details regarding the other two heuristics can be obtained from [17].

The heuristic we propose works broadly in the following fashion. When performing computation at an arbitrary node

- 1) Sort routing prefixes destined to a particular egress router in decreasing order of traffic intensity,
- 2) Sequentially assign each routing prefix to a subset of next hops so as to minimize the given metric.

For clarity we use the following notation in our subsequent discussion:

At an arbitrary node  $n$ , when assigning routing prefixes associated with an arbitrary egress router (destination)  $m$  to next hops:

- 1) Denote the set of next hops to egress router  $m$  by  $\mathcal{K}_{n,m} = \{1, 2, \dots, K_{n,m}\}$ .
- 2) Denote the desired (optimal) traffic load (for egress router  $m$ ) on hop  $k \in \mathcal{K}_{n,m}$  by  $f_k$ .
- 3) Denote the traffic intensity of routing prefix  $i$  by  $x_i$ . Denote the collective set of the routing prefixes (at  $n$  for  $m$ ) that need to be assigned to next hops by  $\mathcal{X}_{n,m}$ . Let  $N_{n,m} = \|\mathcal{X}_{n,m}\|$ .
- 4) Denote the traffic load on hop  $k$  after heuristic  $H$  has assigned  $i$  routing prefixes by  $l_k^i$ . Assume  $l_k^i = 0$  for  $i \leq 0$ .

#### A. The MIN-MAX LOAD heuristic

The Min-Max Load heuristic is similar to a work conserving scheduling algorithm which tries to minimize the maximum load on any processor (the maximum makespan, [10]). Heuristic MIN-MAX LOAD tries to minimize the maximum ratio

of assigned traffic to the optimal traffic load over all hops. The difference now is that each task (stream) can be split equally among multiple processors (next hops) and the processors (next hops) can have different speeds (optimal traffic loads).

Algorithm MIN-MAX LOAD:

- 1) Sort the set of prefixes  $\mathcal{X}_{n,m}$  in descending order of traffic intensity.
- 2) For each prefix  $i \in \mathcal{X}_{n,m}$  choose a subset of next hops  $\tilde{\mathcal{M}} \in \mathcal{K}_{n,m}$ , with cardinality  $\|\tilde{\mathcal{M}}\|$  which minimizes

$$\max_{k \in \mathcal{K}_{n,m}} \left( \frac{l_k^{i-1} + \frac{x_i}{\|\tilde{\mathcal{M}}\|}}{f_k} \right)$$

Step 2) can be achieved in two stages. First, for each index  $p = 1, 2, \dots, K_{n,m}$ , do a *virtual* assignment of routing prefix  $i$  to a set of  $p$  hops which yields the smallest maximum. This can be done by simply sorting the set  $\left\{ \frac{l_k^{i-1} + x_i/p}{f_k} \right\}$ ,  $k \in \mathcal{K}_{n,m}$  in increasing order, re-indexing them and *virtually* assigning  $i$  only to the first  $p$  hops.

Second, from all the  $K_{n,m}$  such possible assignments, choose the one with the smallest maximum for an *actual* assignment. In case of a tie, choose a lexicographically smaller assignment. The complexity of the algorithm is  $O(N \log N + NK^2)$ , where we have set  $N = N_{n,m}$ ,  $K = K_{n,m}$  for simplicity.

We outline our implementation of the heuristics in the pseudo-code below :

#### procedure **Selective Hop Allocation**

Input  $\leftarrow$  (Link Weights  $\{1 + \tilde{W}_{ij}\}$ , optimal traffic allocation  $\{\tilde{Y}_{ij}^t\}$ , Traffic Matrix  $T$ )

For each destination node  $m$  do

Run Dijkstra's algorithm with weights  $\{1 + \tilde{W}_{ij}\}$

For each node  $n \neq m$  in order of decreasing distance from  $m$  do

Apply the heuristic to the set of routing prefixes  $\mathcal{X}_{n,m}$  to determine, for each routing prefix  $i$ , the set of next hops  $\mathcal{K}_{n,m}^i \subset \mathcal{K}_{n,m}$

For each routing prefix  $i \in \mathcal{X}_{n,m}$  do

Update the intensity of the corresponding routing prefix at each node  $j \in \mathcal{K}_{n,m}^i$

done

done

done

For the sake of continuity, analysis of the heuristic is postponed till the appendix. There we show that the MIN-MAX LOAD heuristic achieves a load

ratio that is within a factor of  $(1 + \ln K/2)$  of the ratio achieved by an optimum allocation (under the equal splitting constraint).

#### IV. EXPERIMENTS

In order to evaluate the effectiveness of our approach, we conducted two sets of experiments on artificially generated topologies as well as on an actual ISP topology, namely the Sprint Backbone. In the first set we studied the performance of our heuristic when compared against optimal routing. In the second set of experiments we studied the trade-off between performance and configuration overhead by varying the number of routing prefixes for which we controlled the set of next hops they were assigned.

For purposes of comparison, we solved a linear multi-commodity flow routing problem with the same piecewise linear cost function used in [7]. The only constraint in the routing problem is flow conservation and consequently it provides a lower bound on the performance of any routing scheme, for the same metric. Hence forth, we shall refer to this problem as the “*optimal routing problem*” and its solution as the “*optimal routing solution*”. The solution to this problem is a set of paths (traffic allocation) for each commodity (or destination node) which yields  $\tilde{c}_{i,j}$ , the bandwidth consumed on each link  $(i, j)$ .

We reproduce the optimal allocation problem with regard to this cost function below for completeness. Let the flow to destination node  $t$  on link  $(i, j)$  be denoted by  $y_{i,j}^t$ . Let the total flow on link  $(i, j)$  be  $f_{i,j} = \sum_{t \in V} y_{i,j}^t$  and the capacity is  $C_{i,j}$ . Denote the cost of link  $(i, j)$  by  $\Phi_{i,j}(f_{i,j}, C_{i,j})$ , which is a piecewise linear function that approximates an exponentially growing curve (the exact pieces of the function are presented in Eqn. (6) - Eqn. (11)). The cost grows as the traffic on the link increases and the rate of growth accelerates with increasing utilization. Evolution of the cost function with link load is shown in Figure 1. The *optimal routing problem* may then be formulated as:

$$\begin{aligned} \min \quad & \sum_{(i,j) \in E} \Phi_{i,j}(f_{i,j}, C_{i,j}) \\ \text{subject to} \quad & \end{aligned}$$

$$\sum_{j:(j,i) \in E} Y_{j,i}^t - \sum_{j:(i,j) \in E} Y_{i,j}^t = \begin{cases} -D_t & i = t \\ d_r & i = s_r \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

$$\forall i \in V, t \in V$$

$$f_{i,j} = \sum_{t \in V} y_{i,j}^t, \quad \forall (i, j) \in E$$

$$u_{i,j} = f_{i,j}/C_{i,j}, \quad \forall (i, j) \in E$$

$$\Phi_{i,j} = f_{i,j}, \quad u_{i,j} \leq 1/3 \quad (6)$$

$$\Phi_{i,j} = 3f_{i,j} - \frac{2}{3}C_{i,j}, \quad 1/3 \leq u_{i,j} \leq 2/3 \quad (7)$$

$$\Phi_{i,j} = 10f_{i,j} - \frac{16}{3}C_{i,j}, \quad 2/3 \leq u_{i,j} \leq 9/10 \quad (8)$$

$$\Phi_{i,j} = 70f_{i,j} - \frac{178}{3}C_{i,j}, \quad 9/10 \leq u_{i,j} \leq 1 \quad (9)$$

$$\Phi_{i,j} = 500f_{i,j} - \frac{1468}{3}C_{i,j}, \quad 1 \leq u_{i,j} \leq \frac{11}{10} \quad (10)$$

$$\Phi_{i,j} = 5000f_{i,j} - \frac{16318}{3}C_{i,j}, \quad 11/10 \leq u_{i,j} \quad (11)$$

where, as before

$$D^t = \sum_{r:r \in \mathcal{R}, t_r=t} d_r \quad (12)$$

Note that the approaches presented in [21] and [7] are not limited to any particular cost function. We simply chose this cost function as an example. Also note that the above formulation is based on the idea of destination based aggregation ([1]) presented in Section II-A and was used for computation because of its lower complexity. In the rest of the section, we explain our experimental set up and discuss our observations regarding performance and complexity trade-off.

##### A. Experimental Set Up

For our experiments, the artificial topologies were generated using the Georgia Tech [22] and BRITE [13] topology generators<sup>5</sup>. The topologies constructed using the generators were random graphs chosen from a grid using either a uniform or waxman distribution. The link capacities were either set uniformly to 500 Mbps in some cases, or chosen randomly from a uniform distribution in other cases. Actual physical link capacities were used for the topology based on the Sprint backbone.

For the artificially generated topologies, we present results for random traffic matrices that were

<sup>5</sup>BRITE allows several options for generating topologies: AS Level, Hierarchical and router level. We chose the router level option.

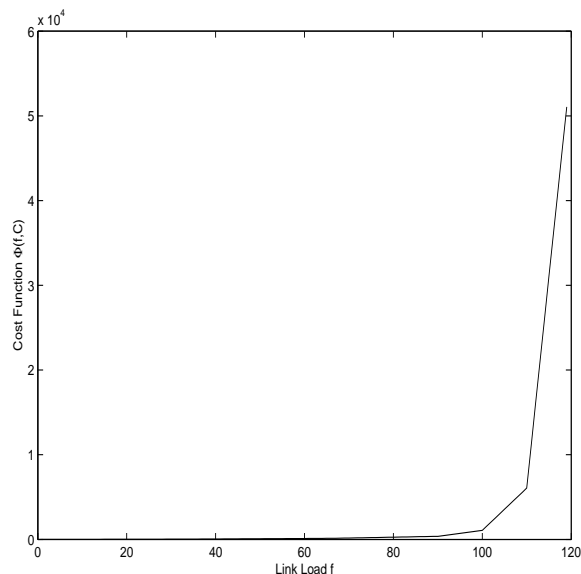


Fig. 1. Evolution of cost function  $\Phi_{i,j}(f_{i,j}, C_{i,j})$  as a function of link load  $f_{i,j}$

generated by picking the traffic intensity of each routing prefix from a pareto or uniform distribution. The choice of a pareto distribution was motivated by measurements taken from several routers on the Sprint backbone. We also experimented with other distributions, eg. uniform, bimodal, exponential and gaussian. Of these, we present results for the uniform distribution since it is representative of the others. The other parameter of importance is the number of routing prefix associated with each egress router, that is, the granularity of the matrix. For this, we again used both uniform and pareto distributions, as it gives a reasonable coverage for the possible difference in the number of available routing prefixes to a given egress router. The Sprint traffic matrix was based on actual traffic traces downloaded from access links to two of the Sprint backbone routers. The traces was measured at the granularity of the routing table entries<sup>6</sup> and gives us two rows of the traffic matrix. The routing prefix intensities in the remaining rows were generated artificially using a pareto distribution for both intensity and granularity. Details of how the entire Sprint traffic matrix was constructed can be found in [18].

Each experiment was conducted in the following fashion :

- 1) For each network topology, we generated random traffic matrices, varying both the total

number of routing prefixes and distribution<sup>7</sup> from which the ingress traffic intensity of each routing prefix was picked.

- 2) Hot spots were introduced in the traffic matrix by randomly selecting elements from the traffic matrix and scaling them to create several instances of the traffic matrix. We tested cases where only some of the traffic elements were chosen and also cases where all entries were chosen. In the latter case, this involves scaling the entire traffic matrix.
- 3) The “*optimal routing problem*”, Eqn. (12), was then solved for each such instance (topology and traffic matrix).
- 4) The linear program, Eqn. (3), with the optimal link bandwidths from the “*optimal routing solution*” as input, was solved to obtain the traffic allocation (which was aggregated based on destination, ref. Section II-A) and the set of link weights.
- 5) Finally, our heuristic was run over the network with the link weights and traffic flows from the previous step (*please refer to pseudo-code*) in order to approximate the optimal load profile of Eqn. (3).

We used ILOG CPLEX to solve the *optimal routing problem* and the linear program. On a Dell 1500 1 Ghz machine, it took about 2 hours to solve the *optimal routing problem* and 10 minutes for the linear program and our heuristic on the largest networks.

### B. Performance Comparison against Optimal Routing

We now present and discuss the results of our experiments. In Figure 2 we plot Cost vs Total Traffic for the MIN-MAX LOAD heuristic and the optimal routing solution for the Sprint topology. The horizontal lines represent various levels of maximum average link utilization over all links for optimal routing. The entire traffic matrix was scaled for the experiments involving the Sprint backbone. From the figure, we see that in all the cases, the heuristic is very near the optimal solution indicating that it is able to match the optimal traffic split very closely.

For comparison, we have also shown the performance of standard OSPF routing with weights

<sup>6</sup>The routing prefix intensities are averages over 10 hrs.

<sup>7</sup>Except in the case of the Sprint traffic matrix.

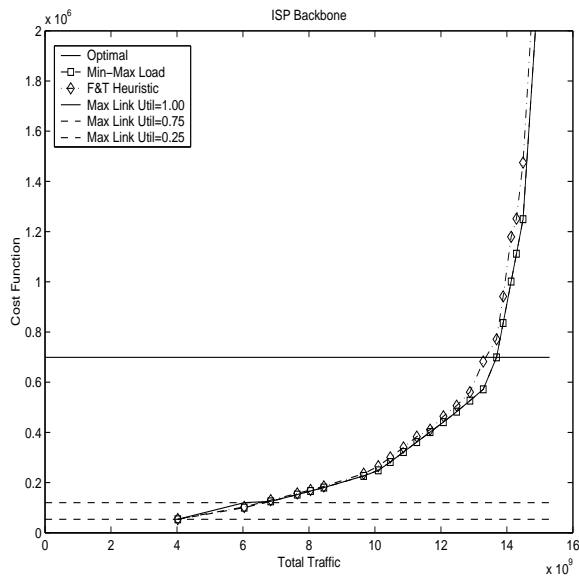


Fig. 2. Sprint Backbone: Performance of heuristic vs. optimal routing.

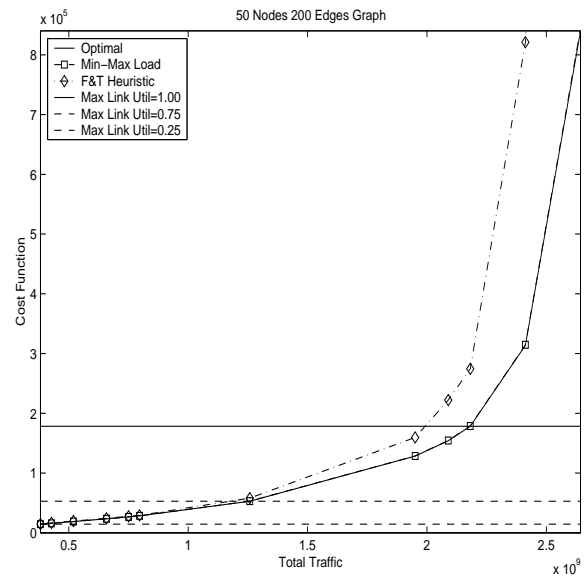


Fig. 4. 50 Node 200 Edge graph (BRITE Generated): Performance of heuristic vs. optimal routing

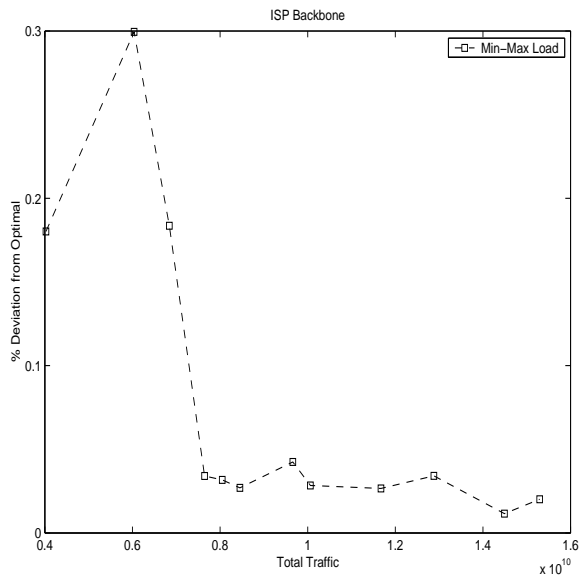


Fig. 3. Sprint Backbone: % Deviation of the heuristic from optimal routing.

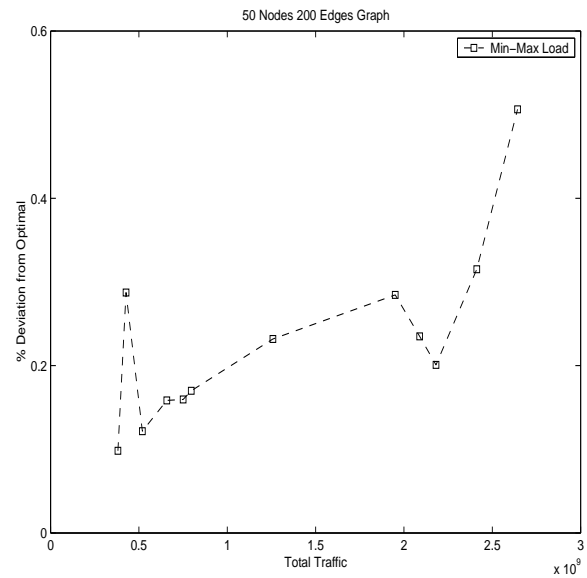


Fig. 5. 50 Node 200 Edge graph (BRITE Generated): % Deviation of the heuristic from optimal routing.

computed using our implementation of the heuristic proposed in [7] (denoted by “F&T Heuristic” in the graph). The heuristic uses local search techniques to determine the best OSPF weight setting. Specifically, it searches the neighbourhood of a given weight setting by performing random link weight changes in order to determine a better weight setting. It then repeats the search in the neighbourhood of the new weight setting. A unique feature of the heuristic is its use of hash tables to avoid cycling as well as re-computation of same

routings under different weight settings. In order to avoid getting trapped in local minima valleys, the heuristic performs diversification by randomly selecting a new neighbourhood, again by weight perturbation. Similar to [7] we run the heuristic for a fixed number of iterations (5000) and retain the best weight setting. In our experiments, we found the heuristic to perform quite well, closely tracking the optimal cost for utilizations below 75%. However, instances in [7] show that the heuristic can deviate significantly from the optimal routing even at low

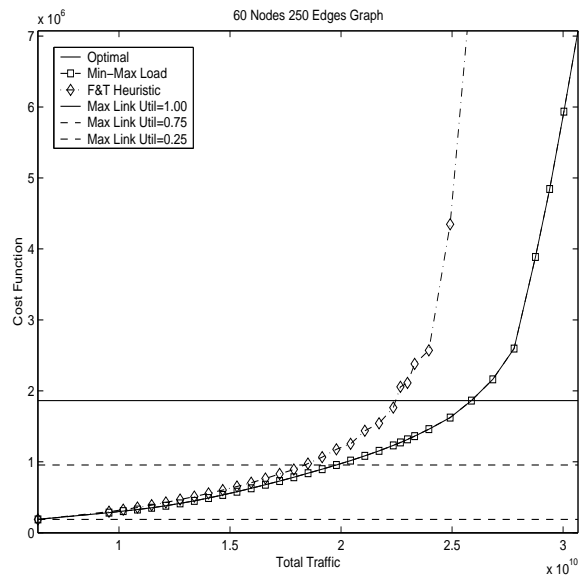


Fig. 6. 60 Node 250 Edge graph (GT Topology Generator): Performance of the heuristic vs. optimal routing

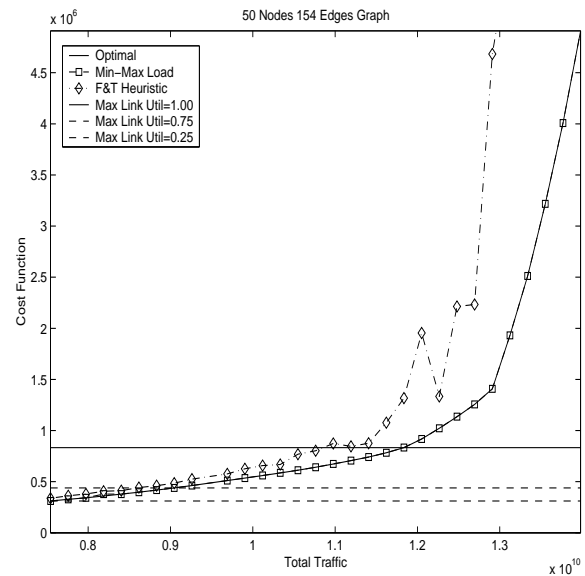


Fig. 8. 50 Node 154 Edge Graph (GT Topology Generator): Performance of the heuristic vs optimal routing.

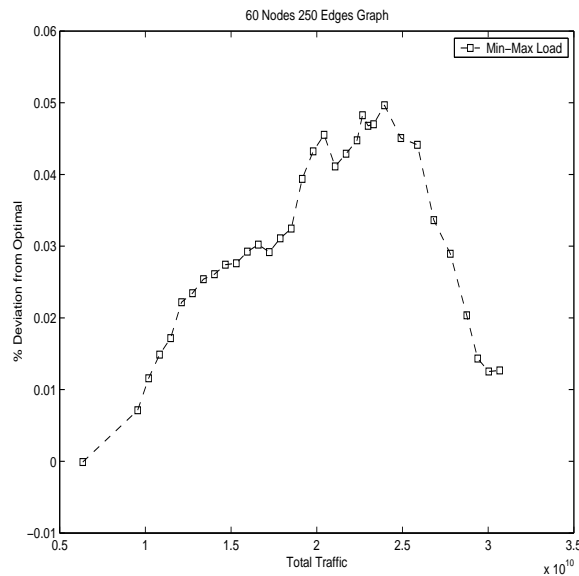


Fig. 7. 60 Node 250 Edge graph (GT Topology Generator): % Deviation of the heuristic from optimal routing

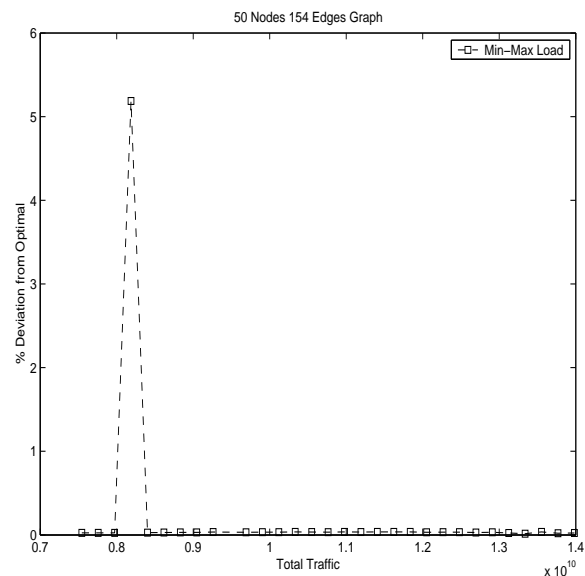


Fig. 9. 50 Node 154 Edge Graph (GT Topology Generator): % Deviation of the heuristic from optimal routing.

utilizations. Furthermore, we note our heuristics can be used to improve the performance of the Fortz and Thorup heuristic (see Section IV-D).

In Figure 3, we plot the % deviation of the MIN-MAX LOAD heuristic from the optimal. The maximum deviation of the heuristic is well within 1% of the optimal, highlighting the ability of the heuristic to approximate the desired (optimum) load very closely.

We now look at 4 artificially generated topologies that we used in our experiments. In Figure 4, we

plot Cost vs Total Traffic for the heuristic, optimal routing as well as standard OSPF routing, on a 50 Node 200 Edge topology with a granularity of 26500 routing prefixes per node. This number was chosen simply as an approximation of the number of routing prefixes in a backbone router. We have conducted experiments with up to 100,000 routing prefixes and as few as 500 routing prefixes without any significant change in performance of the heuristic. The topology was generated using the BRITE generator and all links were set to a capacity

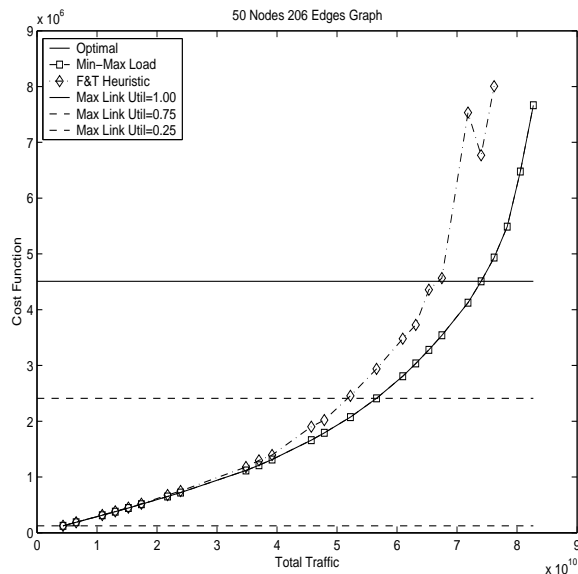


Fig. 10. 50 Node 206 Edge Waxman Graph (GT Topology Generator): Performance of the heuristic vs. optimal routing.

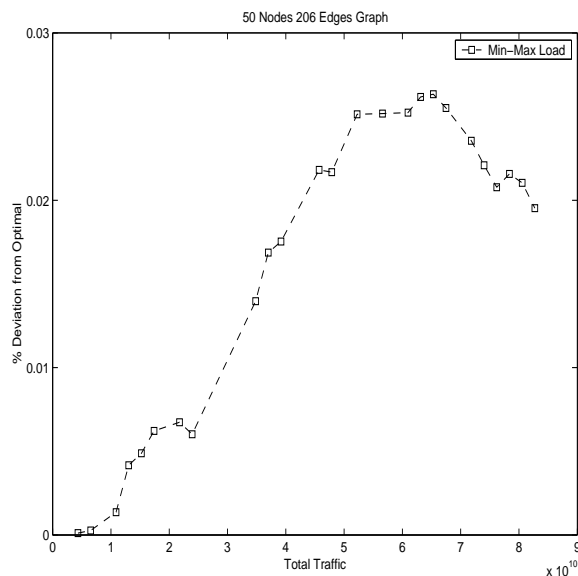


Fig. 11. 50 Node 206 Edge Waxman Graph (GT Topology Generator): % Deviation of the heuristic from optimal routing.

of 500 Mbps. The number of prefixes for each node-pair were chosen from a uniform distribution and the intensity of the entries in the traffic matrix for this experiment were generated from a pareto distribution. Hot-spots were generated by scaling 70% of the traffic elements. We note from Figure 4, that the heuristic closely tracks the optimal. An alternate view is provided in Figure 5 where we plot the % deviation of the heuristic from the optimal. The low percentage deviation (0.2%–1%) from the optimal value again highlights the effectiveness of

the heuristic.

For all the 3 remaining topologies, the traffic matrix granularity as well as intensities were chosen from a uniform distribution. Hot spots were created by scaling 50% of the node-pairs in the traffic matrix. Cost and % deviation curves for a 60 node 250 edge topology are shown in Figures 6 and 7 respectively. Results for a 50 node 154 edge random topology are presented in Figures 8 and 9. Both topologies were generated by the Georgia Tech topology generator using the uniform distribution and their link capacities were set to 500 Mbps. Finally, the cost and % deviation graphs for a 50 node 206 edge random topology generated using the waxman-1 distribution are shown in Figures 10 and 11. The link capacities for this topology were randomly chosen from a uniform distribution. We note that for all three topologies, the heuristic performs very well, closely tracking the optimal cost.

### C. Non-Integer Link Weights

We should point out that the dual of the linear program, Eqn. (4), is not guaranteed to yield exact integer solutions for link weights. In practice, routing protocols like OSPF and IS-IS have a finite field width for link weight information ([11],[16]). If the link weights obtained from the linear program are not exact integers, fitting them within the provided field length can affect performance in the form of routings that are different from the optimal routing. Hence it is important to study how errors in link weights influence performance.

There are two factors which can introduce inaccuracies in link weights. First of course is the loss of precision in rounding off link weights, especially in the presence of recurring fractions, e.g.  $2/3$ . The second factor is the non-zero tolerance required by optimizers to converge to feasible solutions. This implies that the optimal link weights (and path costs) are accurate only within a certain tolerance. For example, two path costs are treated to be equal by the optimizer if the difference in costs is less than the specified tolerance. In our experiments, the tolerance limit was set to  $10^{-5}$ . The presence of recurring fractions and non-zero tolerance limits means that even with large precision integer representations, these errors still persist and in fact are

exacerbated as the integer precision decreases<sup>8</sup>

Figure 12 shows the impact of the length of the link weight field on % deviation of the heuristic from optimal cost for the 50 Node 206 Waxman Graph. The link weights are treated as exact integers with precision governed by the field length. We plot curves with field lengths of 8, 16 and 24 bits. Observe that the field-length has little impact on performance till link utilizations start increasing. This is because, in all our experiments, at low utilizations ( $< 75\%$ ), the link weights are almost always exact integers<sup>9</sup>. At higher utilizations, errors due to non-zero tolerances and non-integer link weights come into play so that the performance starts to deviate significantly. Note that even with a 24-bit field length, performance is off by at least 50%, indicating that precision is not the dominant issue. Instead the errors are caused by the inherent inaccuracy in link weights. When treated as exact integers, regardless of the precision used, this produces different routings.

In order to avoid problems due to such errors, we use an approach similar to that used by the optimizer. We treat path costs to be equal, whenever they differ by less than the specified tolerance thus allowing for inaccuracies. This explains the much better performance (within 3%) seen for the same instance in Figure 11.

#### D. Applications of the Next Hop Allocation Technique

Although we have presented our technique for next hop allocation within the framework of shortest path routing as computed by the Linear Program, Eq. (3), it is also applicable to paths computed through other methods. Given a set of paths computed using any other technique, we can find the optimal distribution of traffic over the set of pre-computed paths and then use our next hop allocation technique to match this profile. If the constraining factor is a poor set of paths, then modifying the traffic profile may not offer much improvement. However, if the deciding factor is coarse granularity of load balancing over the paths, significant

<sup>8</sup>Note that since the computer is a finite precision machine, these errors are already present in the link weights obtained from the solution of Eq. (4).

<sup>9</sup>At low utilizations, the optimizer can easily find feasible solutions and hence the tolerance limits are not enforced.

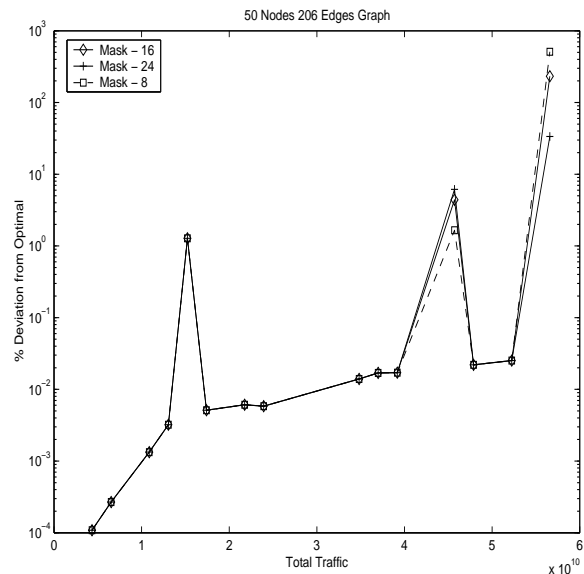


Fig. 12. 50 Node 206 Waxman Topology: Impact of Integer weights on performance

improvement may be obtained with our heuristic. As an example, one could compute shortest paths using the heuristic proposed by Fortz et. al. [7]. However, instead of splitting traffic equally over *all* equal cost next hops, it could be distributed in an “optimal fashion” over the computed paths using the next hop allocation technique presented in this paper. In order to achieve this, we first solve the optimal routing problem but with traffic now constrained to flow *only* on paths computed using the link weights obtained from the ‘F & T’ heuristic. The MIN-MAX LOAD heuristic is then run to shape the allocated traffic to match the optimal load. We present an example of this procedure for the 60 Node 250 Graph in Figure 13, where we show the performance improvement of traffic allocation with the MIN-MAX LOAD heuristic over equal splitting. Note that for both the curves, the same set of paths were used, but the MIN-MAX Load heuristic allows for finer load balancing. In other words, the improvement afforded over the standard ‘F & T’ heuristic by the use of the MIN-MAX Load heuristic can be solely attributed to its ability to better match optimal traffic loads by distributing traffic unevenly. We revisit this issue in the next section, where we explore the impact of the number of equal cost paths (next hops) generated by the routing algorithm.

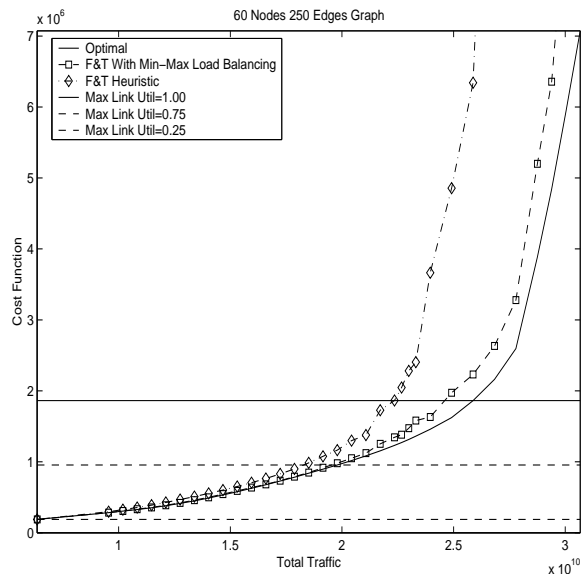


Fig. 13. 60 Node 250 Edge Topology: Impact of MIN-MAX LOAD Heuristic on load balancing with weights computed using Fortz and Thorup Heuristic

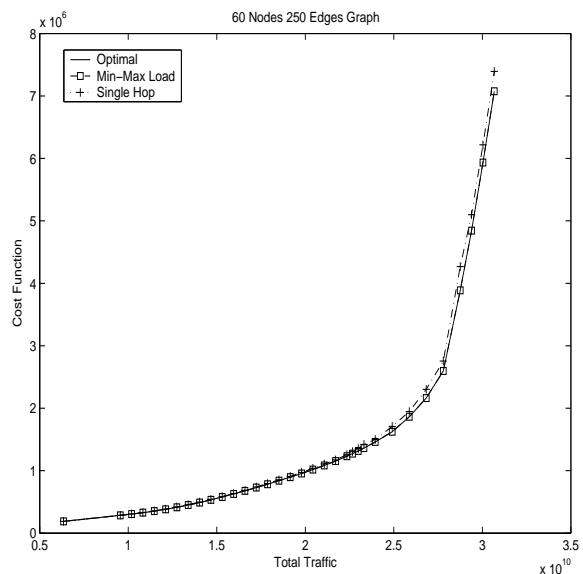


Fig. 14. Impact of Equal Cost Paths on performance for the 60 Node 250 Edge Graph.

| Network                 | Number of Hops |         |
|-------------------------|----------------|---------|
|                         | Average        | Maximum |
| Sprint N/W              | 1.69           | 5       |
| 50 Node, 200 Edge Graph | 1.31           | 4       |
| 50 Node, 154 Edge Graph | 1.20           | 4       |
| 50 Node, 206 Edge Graph | 1.26           | 4       |
| 60 Node, 250 Edge Graph | 1.05           | 5       |

TABLE I  
NUMBER OF EQUAL COST NEXT HOPS FOR EACH  
INGRESS-EGRESS PAIR

### E. Equal Cost Paths

One of the key features of OSPF/ISIS discussed in this paper is the ability to balance traffic across multiple equal cost paths. In this section, we examine the effectiveness of this feature in improving performance. In other words, how is routing performance dependent on its ability to send traffic on more than one path. Intuitively, optimal routing is more likely to use multiple paths to balance traffic at higher loads rather than when capacity is plentiful. Hence, we focus on reasonably high utilization scenarios, albeit less than 100% link loads, for which we compute statistics regarding the number of equal cost next hops used by optimal routing. Table I shows the average and maximum number of equal cost next hops computed by the optimal formulation across all nodes and for all destinations for several network configurations,

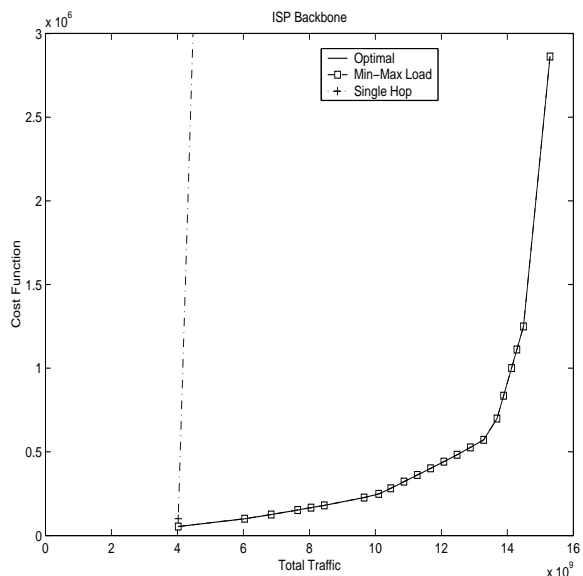


Fig. 15. Impact of Equal Cost Paths on performance for the Sprint Backbone.

We observe that the number of equal cost next hops used by the optimal formulation in order to balance load (although the distribution of traffic across them need not be uniform) varies with the topology. In other words, the benefit of routing over multiple equal cost paths is a function of how the logical topology is designed and also how well the traffic is matched to the topology. To emphasize the dependence of benefits of using equal cost paths on the topology, we evaluate performance when no splitting of traffic is allowed. The “SINGLE HOP”

heuristic chooses exactly one next hop, the one with largest “optimal traffic,  $f_k$ ”, for each destination at each node. In all other aspects it functions exactly like the MIN-MAX Load heuristic.

Figure 14 shows performance curves for the “SINGLE HOP” heuristic the MIN-MAX LOAD heuristic on the 60 Node, 250 edge graph. Observe the near-optimal performance of the “SINGLE HOP” heuristic, indicating that multiple equal cost paths have little impact on performance. The average number of next hops for the graph in Table I is close to 1, which confirms our observation. Note that this configuration is the same as that of Figure 13 which illustrated the improvement that could be achieved by the MIN-MAX Load heuristic over the standard ‘F & T’ heuristic. This is because whenever the ‘F & T’ heuristic computes multiple paths, it ends-up allocating traffic equally across them, even when an optimal solution calls for a very uneven traffic allocation. The MIN-MAX Load heuristic, even if it is constrained to using the same paths, is able to generate the uneven loads the the optimal routing solution calls for.

In contrast, performance on the Sprint network (Figure 15) is significantly enhanced by distributing traffic across multiple next hops, and as a result the “SINGLE HOP” heuristic performs quite poorly. The average number of equal cost next hops for the Sprint network from Table I is close to 2, reflecting this behaviour. Indeed, some major ISPs specifically design their logical topologies to increase the number of equal cost paths, especially between geographical hubs, for purposes of both load balancing (as explained in Section I) as well as robustness. We expect load balancing over equal cost multiple paths to markedly improve performance in these networks.

#### F. Lowering Configuration Overhead

Our other goal was to investigate the trade-off between configuration overhead and performance. Recall that in the original approach the heuristic decides the *subset* of next hops assigned to *every* routing prefix. However, it has been observed that in practice ([3]), a large fraction of the traffic is distributed over a relatively small number of routing prefixes. Our analysis of the backbone traces obtained from the Sprint router show that 95% of the total traffic was accounted for by only 10% of

the routing prefixes, confirming the results reported in [3]. Figure 16 highlights this observation, where we have plotted the cumulative traffic intensity as a function of the number of routing prefixes sorted in decreasing order of their traffic intensities. We can potentially exploit such a phenomenon by configuring the set of next hops for only a few selective routing prefixes that carry most of the traffic and allowing the default assignment of all next hops for the remaining routing prefixes. This has the advantage of lowering configuration overhead, but raises the question of how it impacts performance.

We carried out a systematic study of such a trade-off on all the previous topologies. In each instance, we configured the set of next hops at each node for only a certain set of routing prefixes that were selected based on the amount of traffic they carried. The remaining routing prefixes were split equally over the entire set of next hops as would happen with default OSPF/IS-IS behavior. The set of configured routing prefixes was then progressively increased in each experiment to determine the evolution of the impact on performance. In all cases, the MIN-MAX LOAD heuristic was used when configuring the set of next hops.

The resulting performance curves for the 50 Node 200 Edge topology are shown in Figure 17 and the number of configured routing prefixes are shown in Table II. Each curve on the plot is referenced by the amount of traffic that was accounted for by the configured routing prefixes. This can be cross-referenced from the table against the number of routing prefixes that were configured. We observe that on an average, by configuring about 165 routing prefixes per router (which accounts for about 20% of the traffic), we get good performance till about 50% maximum link utilization. If we configure next hops for about 17 % of all routing prefixes, or 4500 entries, at a router, we account for approximately 75% of the traffic and the resulting performance is quite close to that of optimal routing.

Experiments conducted on the Sprint Backbone (Figure 18, Table III) yield similarly encouraging results. We get good performance up to approximately 50-60% maximum link utilization, by configuring only 200 routing prefixes per router and up to more than 70% link utilization if we configure 600 routing prefixes per router.

The 60 node 250 edge topology (Figure 19, Table IV) and the 50 node 154 edge topology (Figure

20, Table V) present an interesting case in that by carefully configuring only 3 – 5% of the prefixes we get close to optimal performance over a very wide range of link utilizations. We do not get quite such dramatic results for the 50 node 206 edge Waxman topology, (Table VI, Figure 21), but even in this case, we get good performance by configuring around a quarter of the prefixes (50% of the traffic).

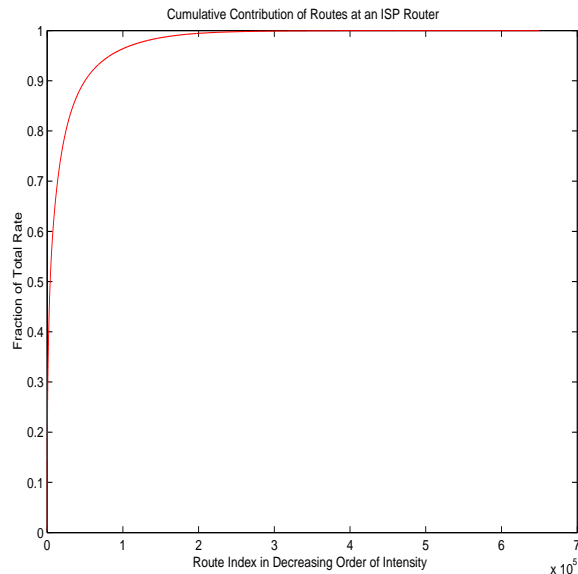


Fig. 16. Cumulative contribution of routing prefixes at a Sprint Router sorted in decreasing order of intensity.

| Prefixes Configured | Total No. of Prefixes | % Allocated Fraction | % of Traffic |
|---------------------|-----------------------|----------------------|--------------|
| 75                  | 26500                 | 0.3%                 | 10 %         |
| 165                 | 26500                 | 0.62%                | 20 %         |
| 1252                | 26500                 | 4.7 %                | 50 %         |
| 4500                | 26500                 | 17.0 %               | 75 %         |
| 11747               | 26500                 | 44.32 %              | 90 %         |

TABLE II

CONFIGURATION OVERHEAD: 50 NODE 200 EDGE TOPOLOGY, ALL ENTRIES ARE PER NODE

### V. CONCLUSION

In this paper, we have described and evaluated an approach that offers the benefits of traffic engineering to IP networks without requiring changes to either the routing protocols or the forwarding mechanisms.

Our contribution is three-fold. First, we devised a solution for closely approximating optimal link

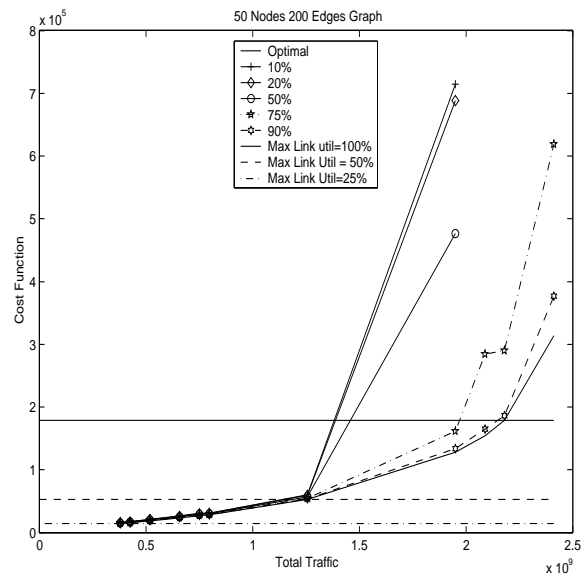


Fig. 17. BRITE 50 Node 200 Edge topology : Performance as a function of configuration overhead

| Prefixes configured | Total No. of Prefixes | % Allocated Fraction | % of Traffic |
|---------------------|-----------------------|----------------------|--------------|
| 160                 | 30700                 | 0.5%                 | 10 %         |
| 200                 | 30700                 | 0.6%                 | 20 %         |
| 620                 | 30700                 | 2 %                  | 50 %         |
| 1750                | 30700                 | 6 %                  | 75 %         |
| 4180                | 30700                 | 14 %                 | 90 %         |

TABLE III

CONFIGURATION OVERHEAD: SPRINT BACKBONE, ALL ENTRIES ARE PER NODE

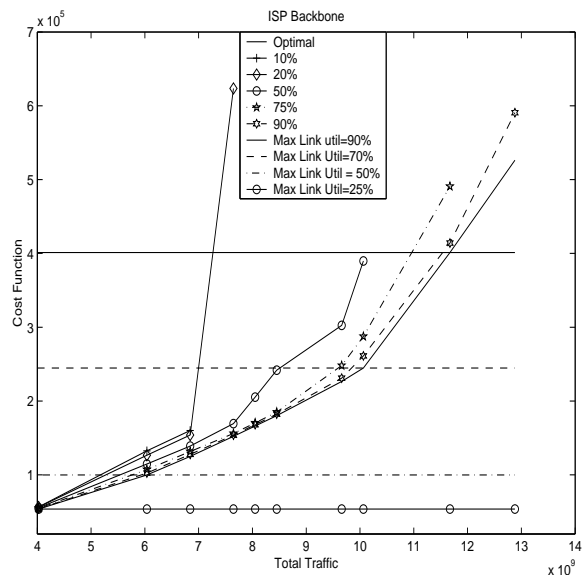


Fig. 18. Sprint Backbone: Performance as a function of configuration overhead

| Avg No. of Routes configured | Total No. of Routes | % Allocated Fraction | % of Traffic |
|------------------------------|---------------------|----------------------|--------------|
| 5125                         | 117890              | 4.3%                 | 10 %         |
| 14183                        | 117890              | 12.0%                | 20 %         |
| 32754                        | 117890              | 27.5 %               | 50 %         |
| 57940                        | 117890              | 49.1 %               | 75 %         |
| 80500                        | 117890              | 68.2%                | 90 %         |

TABLE IV

CONFIGURATION OVERHEAD: 60 NODE 250 EDGE TOPOLOGY (GT TOPOLOGY GENERATOR), ALL ENTRIES ARE PER NODE

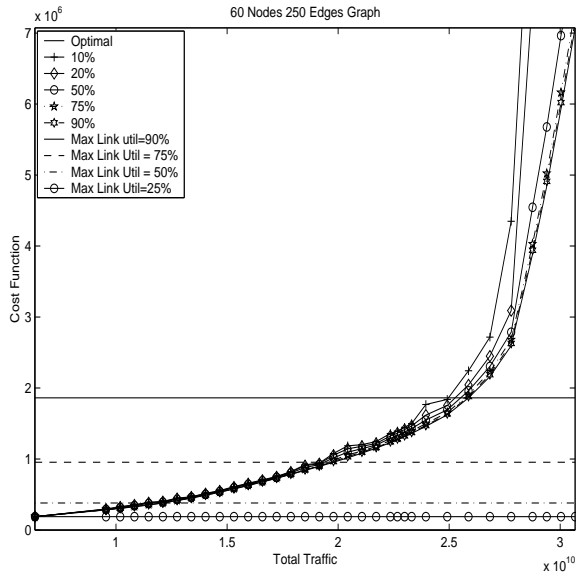


Fig. 19. GT-ITM 60 Node 250 Edge Topology: Performance as a function of configuration overhead

| Prefixes configured | Total No. of Prefixes | % Allocated Fraction | % of Traffic |
|---------------------|-----------------------|----------------------|--------------|
| 5640                | 145827                | 3.8%                 | 10 %         |
| 15805               | 145827                | 10.8%                | 20 %         |
| 37108               | 145827                | 25.4 %               | 50 %         |
| 67132               | 145827                | 46 %                 | 75 %         |
| 95420               | 145827                | 65.4 %               | 90 %         |

TABLE V

CONFIGURATION OVERHEAD: 50 NODE 154 EDGE TOPOLOGY, ALL ENTRIES ARE PER NODE

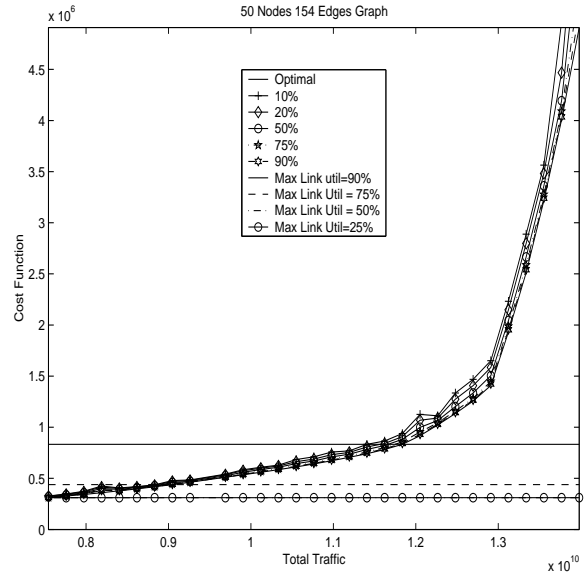


Fig. 20. 50 Node 154 Edge Topology : Performance as a function of configuration overhead

| Prefixes configured | Total No. of Prefixes | % Allocated Fraction | % of Traffic |
|---------------------|-----------------------|----------------------|--------------|
| 4221                | 97772                 | 4.3%                 | 10 %         |
| 11706               | 97772                 | 11.9%                | 20 %         |
| 27131               | 97772                 | 27.5 %               | 50 %         |
| 48098               | 97772                 | 49.8 %               | 75 %         |
| 55754               | 97772                 | 68.5 %               | 90 %         |

TABLE VI

CONFIGURATION OVERHEAD: 50 NODE 206 EDGE WAXMAN TOPOLOGY, ALL ENTRIES ARE PER NODE

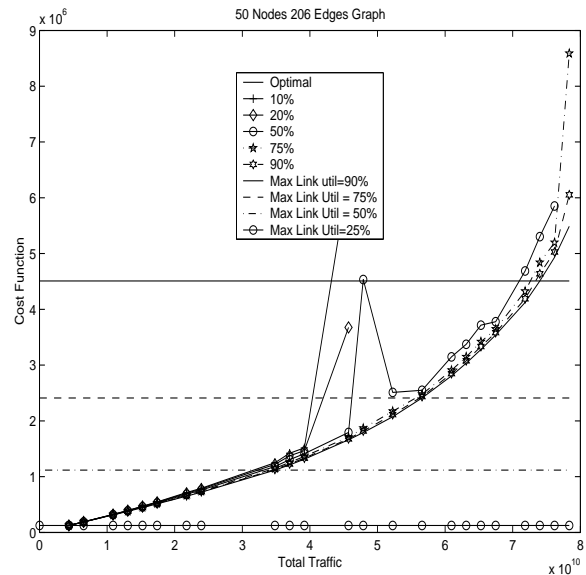


Fig. 21. 50 Node 206 Edge Waxman Topology: Performance as function of configuration Overhead

loads using routing protocols and packet forwarding mechanisms, as they exist today. Second, we proposed a simple heuristic with a provable performance bound (see Appendix) to implement our solution. We performed several experiments in which the heuristic consistently matched the optimal load profile. We believe the heuristic and those presented in [17] to be general enough to be potentially useful in their own right. Finally, we showed, using actual traffic traces, that configuration overhead can be vastly reduced without significant loss of performance. Specifically, by only configuring next hops for a small set of prefixes, we were able to obtain near-optimal performance for link loads of up to 70%. This is obviously an important aspect for the practical deployment of our traffic engineering solution.

Overall, we believe that the paper provides initial arguments in favor of evolving the current infrastructure to support traffic engineering, if and when needed, rather than embark on a migration to a rather different technology. There may be justifications for such a migration, e.g., better support for policies or VPNs, but traffic engineering does not appear to be one of them, and we hope that the results of this paper can help clarify this issue.

There are several directions in which this work can be extended and further improved. One of them is in dealing with non-integer link weights as discussed in Section IV-C. Although we have been able to successfully deal with this issue by coupling the computational tolerance of the optimizer with the available precision of link weights, we are investigating more general solutions to the problem. Another direction we are currently pursuing is that of making our traffic engineering solution robust to unexpected changes in network topology, e.g., link or router failures. This is obviously an important aspect. One that has been considered by both traffic engineering solutions that rely on new forwarding technologies such as MPLS, e.g., [12], and solutions targeting current IP networks, e.g., [8].

## REFERENCES

- [1] H. Abrahamsson, B. Ahlgren, J. Alonso, A. Andersson, and P. Kreuger. A multi path routing algorithm for IP networks based on flow optimization. In *International Workshop on Quality of future Internet Services*, Zurich, Switzerland, October 2002.
- [2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows, Chapter 17, Section 17.2*. Prentice-Hall Inc., 1990.
- [3] S. Bhattacharya, C. Diot, J. Jetcheva, and N. Taft. Pop-level and access-link level traffic dynamics in a tier-1 pop. *Proceedings of ACM SIGCOMM Internet Measurement Workshop (IMW 2001)*, November 2001.
- [4] R. Callon. Use of OSI IS-IS for routing in TCP/IP and dual environments. Request For Comments (Standard) RFC 1195, Internet Engineering Task Force, December 1990.
- [5] Z. Cao, Z. Wang, and E. Zegura. Performance of hashing-based schemes for internet load balancing. In *Proceedings of INFOCOM*, Tel Aviv, Israel, March 2000.
- [6] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. *IEEE/ACM Transactions on Networking*, 9, 2001.
- [7] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings of INFOCOM'2000*, Tel Aviv, Israel, March 2000.
- [8] B. Fortz and M. Thorup. OSPF/IS-IS weights in a changing world. *Journal on Selected Areas in Communication*, February 2002.
- [9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
- [10] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17, March 1969.
- [11] D. Katz, D. Yeung, and K. Kompella. Traffic Engineering Extensions to OSPF Version 2. INTERNET-DRAFT, draft-katz-yeung-ospf-traffic-09.txt, October 2002. work in progress.
- [12] M. Kodialam and T.V. Lakshman. Dynamic routing of bandwidth guaranteed tunnels with restoration. In *PROCEEDINGS of INFOCOM*, Tel Aviv, Israel, March 2000.
- [13] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: Boston university representative internet topology generator. <http://cs-www.bu.edu/brite>, Boston University, April 2001.
- [14] J. Moy. OSPF Version 2. Request For Comments (Standard) RFC 2328, Internet Engineering Task Force, April 1998.
- [15] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. Request For Comments (Standards Track) RFC 3031, Internet Engineering Task Force, January 2001.
- [16] H. Smit. IS-IS Extensions for Traffic Engineering. INTERNET-DRAFT, draft-ietf-isis-traffic-04.txt, December 2002. work in progress.
- [17] A. Sridharan, R. Guérin, and C. Diot. Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks. Technical report, University of Pennsylvania, May 2002. Available at <http://einstein.seas.upenn.edu/publications.html>.
- [18] Ashwin Sridharan, J. Jetcheva, S. Bhattacharyya, C. Diot, R. Guérin, and N. Taft. On the impact of traffic aggregation on traffic aware routing. *Proceedings of the 18th International, COM-31(10)*, October 1983.
- [19] C. Villamizar. OSPF optimized multipath (OSPF-OMP). Available at <http://www.fictitious.org/omp>, 1997. Unpublished Work.
- [20] C. Villamizar. OSPF optimized multipath OSPF-OMP. INTERNET-DRAFT, draft-villamizar-ospf-omp-01.txt, February 1999. (work in progress).
- [21] Z. Wang, Y. Wang, and L. Zhang. Internet traffic engineering without full mesh overlaying. In *Proceedings of INFOCOM'2001*, Anchorage, Alaska, April 2001.
- [22] E. W. Zegura. GT-ITM: Georgia Tech Internetwork Topology Models (software). <http://www.cc.gatech.edu/fac/ellen.zegura/gt-itm/gt-itm/tar.gz>, Georgia Tech, 1996.

## APPENDIX

We first show that problem of splitting prefixes equally over a subset of next hops is NP-complete even at a single node. The problem can be stated as follows.

INSTANCE: A set of flows  $\mathcal{S} = \{s_i\}$  with intensity of flow  $s_i$  denoted by  $x(s_i)$ , set of next hops  $\mathcal{K} = \{k_1, k_2, \dots, k_K\}$  with capacities of hop  $k_i$  denoted by  $f(k_i)$  and an integer  $D$ .

QUESTION: Is there an allocation  $\sigma$  of each flow  $s_i$  to a set of next hops, ie.,  $\sigma : s_i \rightarrow \{k_1, k_2, \dots, k_K\}$  and a split  $p(s_i) = |\sigma(s_i)|$ , such that

$$\max_{j \in \mathcal{K}} \left\{ \frac{l_j}{f(k_j)} \right\} \leq D$$

where  $l_j = \sum_{s_i: j \in \sigma(s_i)} \frac{x(s_i)}{p(s_i)}$ .

We restrict the above problem by setting  $K = 2$ . This implies that now each flow has three choices, either hop 1, or hop 2, or be equally split on both hops. Observe that this is equivalent to *a priori* splitting a flow  $s_i$  into two equal flows with half the intensity and then making *independent* decisions for each half, but now without splitting the new flows. This new problem can be recast into a modified form of the partition problem, which we call MODIFIED-PARTITION, defined as follows.

INSTANCE: An integer  $\alpha$  and a set of elements  $A = \{a_i\} \cup \{a'_i\}$  with size of element  $a_i$  given by  $s(a_i)$ . Further,  $s(a_i) = s(a'_i)$ , that is, for each element  $a_i$  in  $A$ , there is an element  $a'_i$  with identical size. We shall henceforth refer to this as the mirroring property.

QUESTION: Is there a partition of  $A$  into disjoint sets  $A'$  and  $A - A'$  such that

$$\alpha \sum_{a \in A'} s(a_i) = \sum_{a \in A - A'} s(a).$$

**Theorem:** MODIFIED-PARTITION is NP-Complete.

*Proof:* It is easy to see that MODIFIED-PARTITION  $\in$  NP, since one only has to guess a partition and check if it satisfies the relation.

We transform 3 DIMENSIONAL MATCHING (3DM) to MODIFIED-PARTITION. The proof closely follows the outline of reduction from 3DM

to PARTITION presented in [9]. Let the sets  $W, X, Y$ , with  $|W| = |X| = |Y| = q$ , and  $M \subseteq W \times X \times Y$ , the candidate matching set, be an arbitrary instance of 3DM. Let the elements of the sets be denoted by

$$\begin{aligned} W &= \{w_1, w_2, \dots, w_q\} \\ X &= \{x_1, x_2, \dots, x_q\} \\ Y &= \{y_1, y_2, \dots, y_q\} \end{aligned}$$

and

$$M = \{m_1, m_2, \dots, m_k\}$$

where  $k = |M|$ . We must construct a set of elements  $A$  with size  $s(a) \in Z^+$ , that satisfies the mirroring property and a number  $\alpha$  such that,  $A$  contains a subset  $A'$  satisfying

$$\alpha \sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a)$$

if and only if  $M$  contains a matching. The set  $A$  will contain  $2k + 2$  elements. The first  $k$  are constructed from  $M$ , where each element  $a_i$  is associated with a triple  $m_i \in M$ . The size  $s(a_i)$  of  $a_i$  will be specified in its binary representation, in terms of a string of 0's and 1's divided into  $3q$  zones of  $p = \lceil \log_2(2k + 1) \rceil$  bits each. Each of these zones is labeled with an element from  $W \cup X \cup Y$  as shown in Figure 22.

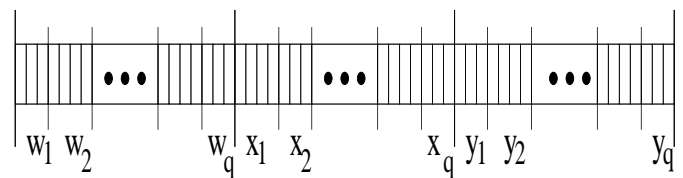


Fig. 22. Labeling of the  $3q$  zones, each containing  $p = \lceil \log_2(2k + 1) \rceil$  bits of the binary representation of  $s(a)$ .

The representation for  $s(a_i)$  depends on the corresponding triple  $m_i = (w_{f(i)}, x_{g(i)}, y_{h(i)}) \in M$ . It has a 1 in the rightmost bit position of the zones labeled  $w_{f(i)}$ ,  $x_{g(i)}$  and  $y_{h(i)}$ . Hence, the size of  $a_i$  is given by

$$s(a_i) = 2^{p(3q-f(i))} + 2^{p(2q-g(i))} + 2^{p(q-h(i))}$$

The next  $k$  elements of  $A$  are just replicas of the first  $k$ , that is  $s(a_{i+k}) = s(a_i)$ . Note that if we sum up all entries in a zone over all elements  $\{a_i : 1 \leq i \leq 2k\}$ , it never exceeds  $2k = 2^p - 1$ . Hence in adding  $\sum_{a \in A'} s(a)$  for any subset  $A' \subseteq \{a_i : 1 \leq i \leq 2k\}$ , there will never be any carries from one zone to the next. Now if we let

$$B = \sum_{j=0}^{3q-1} 2^{pj}$$

which is the number whose binary representation has a 1 in the rightmost position of every zone, then any subset  $A' \subseteq \{a_i : 1 \leq i \leq 2k\}$  will satisfy

$$\sum_{a \in A'} s(a) = B$$

if and only if  $M' = \{m_i : a_i \in A'\}$  is a matching for  $M$ . Also note that by virtue of construction  $A'$  can contain only distinct elements and hence no duplicates, In other words,  $A' \subseteq \{a_i : 1 \leq i \leq k\}$ .

We now define the remaining 2 elements of the set  $A$ ,  $b$  and  $b'$  with sizes

$$s(b) = s(b') = \beta B - \sum_{i=1}^k s(a_i)$$

where  $\beta$  is a number such that  $s(b) > B$ . The final step is to choose  $\alpha$  which we set to be equal to  $2\beta - 1$ .

Now if,  $M'$  is a matching, then we must have for the corresponding set  $A'$  that  $\sum_{a \in A'} s(a) = B$ . The remaining elements of  $A$  sum up to

$$\begin{aligned} & \sum_{i=1}^{2k} s(a_i) - B + 2s(b) \\ &= 2 \sum_{i=1}^k s(a_i) - B + 2\beta B - 2 \sum_{i=1}^k s(a_i) \\ &= (2\beta - 1)B \end{aligned}$$

which satisfies the requirement, that is

$$(2\beta - 1) \sum_{a \in A'} s(a) = \sum_{a \in A-A'} s(a).$$

Conversely, if there exists a subset  $A' \subseteq A$ , such that

$$(2\beta - 1) \sum_{a \in A'} s(a) = \sum_{a \in A-A'} s(a)$$

then by virtue of the property that

$$\sum_{a \in A} s(a) = 2\beta B,$$

the elements of  $A'$  must sum up to  $B$ . Moreover, neither  $b$  nor  $b'$  can be in  $A'$  since they are both larger than  $B$ . Hence only elements  $\{a_i : 1 \leq i \leq k\}$  can be in  $A'$  which results in a matching. Thus,  $3DM \propto \text{MODIFIED-PARTITION}$  and the theorem is proved.

### A. Analysis of the MIN-MAX LOAD Heuristic

Before we analyze the MIN-MAX LOAD heuristic, we define some notation. All analysis is specific to a particular node  $n$  and some egress router  $m$ . Consequently, for ease of exposition, we shall dispense with suffixes with the implication that all variables are specific to a given pair  $(n, m)$ .

- 1) Let  $\mathcal{X}$  denote the set of streams (prefixes) at node  $n$  destined to egress router  $m$  and let  $x_i$  denote the intensity of stream  $i$ . Also, let  $N = \|\mathcal{X}\|$ .
- 2) Let  $\mathcal{K}$  denote the set of next-hops at  $n$  for  $m$ . Let  $f_k$  denote the capacity of hop  $k$  and  $K = \|\mathcal{K}\|$ .
- 3) For simplicity we refer to the load on hop  $k$  as  $l_k$ . The context will make it clear if this is the load *before* or *after* assignment of any particular prefix.

Let

$$\begin{aligned} L &= \sum_{i=1}^N x_i \\ F &= \sum_{k \in \mathcal{K}} f_k. \end{aligned}$$

Our analysis of the MIN-MAX LOAD heuristic consists of two steps. We first give a performance bound when the set of routes is unordered. We then demonstrate an improved bound when the set of routes is ordered according to their traffic intensity.

*Proposition 1:* Heuristic MIN-MAX LOAD achieves a load that is no more than  $(1 + \ln K)$  times that of the maximum load with an optimal algorithm, where  $K$  is the number of next hops (for a given destination).

**Proof:** The proof proceeds in two steps. First we identify a key property of the MAX-MIN LOAD heuristic, namely that for each *virtual* assignment, we can associate a distinct hop. Next we use this property to establish the main result.

Denote the maximum load achieved by Heuristic MIN-MAX LOAD as  $\gamma(\mathcal{X}, \mathcal{K})$ . Let hop  $t \in \mathcal{K}$  achieve this load and the last prefix assigned to hop  $t$  be prefix  $j$ . By definition, we have

$$\gamma(\mathcal{X}, \mathcal{K}) \geq \frac{l_k}{f_k} \quad \forall k \in \mathcal{K}.$$

Let  $\gamma_o(\mathcal{X}, \mathcal{K})$  denote the maximum load achieved by an optimum algorithm that satisfies the equal subset splitting constraint, i.e., splits traffic equally across the subset of next hops that have been assigned to a route. Note that  $\gamma_o(\mathcal{X}, \mathcal{K}) \geq \frac{L}{F}$  where  $\frac{L}{F}$  is the optimum attained if arbitrary splitting of routes was allowed at the node.

First recall how the second step of heuristic MIN-MAX LOAD works. The heuristic does a *virtual* assignment of the streams over an increasing sequence of hops,  $\mathcal{M} = \{d \leq k : d \in \mathcal{K}\}$ ,  $k = 1, 2, \dots, K$  as described in the algorithm and chooses the arrangement with the smallest maximum for an *actual* assignment. We make the following observations:

- 1) The smallest maximum among all *virtual* assignments of prefix  $j$  must be  $\gamma(\mathcal{X}, \mathcal{K})$ . If there were a smaller maximum, it would contradict our assumption that  $j$  is the last prefix assigned to hop  $t$ .
- 2) In a *virtual* assignment of prefix  $j$  over  $p$  hops, let  $r_p$  denote the index of the hop with the maximum load *among* the *virtually* assigned hops (to which a portion  $x_j/p$  of the route was *virtually assigned*). Then  $r_p$  must be maximal over *all* hops, for that *virtual* assignment. If it were not, let there exists a hop  $k$  (which must belong to the set of *virtually unassigned* hops) such that

$$\frac{l_k}{f_k} > \frac{l_{r_p} + x_j/p}{f_{r_p}}. \quad (13)$$

Since all hops satisfy the property  $\frac{l_k}{f_k} \leq \gamma(\mathcal{X}, \mathcal{K})$ , we have an assignment of prefix  $j$  which yields a lower load ratio than  $\gamma(\mathcal{X}, \mathcal{K})$ . This contradicts our initial assumption that  $j$  was the last prefix assigned to hop  $t$  such that its load was  $\gamma(\mathcal{X}, \mathcal{K})$ . Hence  $r_p$  must be maximal.

From Observation 1, we have

$$\frac{l_{r_p} + x_j/p}{f_{r_p}} \geq \gamma(\mathcal{X}, \mathcal{K}). \quad (14)$$

Clearly, the following relation holds for all the  $K-p$  *virtually unassigned* hops

$$\frac{l_k + x_j/p}{f_k} \geq \frac{l_{r_p} + x_j/p}{f_{r_p}} \quad (15)$$

since only the  $p$  smallest hops are chosen in the *virtual* assignment. We then have from Eqn. (14) and Eqn. (15) that the following relation must hold for  $r_p$  and the remaining (if any)  $K-p$  *virtually unassigned* hops :

$$\frac{l_k + x_j/p}{f_k} \geq \gamma(\mathcal{X}, \mathcal{K}). \quad (16)$$

Using these 2 observations we make the following claim.

**Claim :** For every *virtual* assignment over  $p = 1 \dots K$  hops of prefix  $j$ , we can identify a distinct hop  $k(p)$  that satisfies

$$\frac{l_{k(p)} + x_j/p}{f_{k(p)}} \geq \gamma(\mathcal{X}, \mathcal{K}). \quad (17)$$

**Proof:** The proof is by induction. For a *virtual* assignment by the heuristic over  $p$  hops, let  $\mathcal{A}_p$  denote the set of hops such that:

$$\mathcal{A}_p = \left\{ k : \frac{l_k + x_j/p}{f_k} \geq \gamma(\mathcal{X}, \mathcal{K}) \right\}.$$

Note from Observation 2 and Eqn. (16) that  $\mathcal{A}_p$  comprises of at least  $r_p$  and the  $K-p$  unassigned hops. Hence

$$\|\mathcal{A}_p\| \geq K - p + 1.$$

Also note that  $\mathcal{A}_p$  is a non-decreasing sequence for  $p = K, K-1, \dots, 1$ , because if some hop  $k \in \mathcal{A}_n$ , then  $k \in \mathcal{A}_{n-1}$ ,  $n \geq 1$ . The claim certainly holds for  $p = K$  since by Observation 1, there is at least 1 hop which has a load of  $\gamma(\mathcal{X}, \mathcal{K})$ . Let the claim hold for all  $p = K, K-1, \dots, n$ . Then by the non-decreasing property,  $\mathcal{A}_n$  contains all the  $K-n+1$  distinct hops. Let us look at a *virtual* assignment over  $n-1$  hops. We know that

$$\|\mathcal{A}_{n-1}\| \geq K - n + 2.$$

Since only  $K-n+1$  hops have been associated (with *virtual* assignments  $p = K$  to  $p = n$ ) and by the non-decreasing property,  $\mathcal{A}_{n-1}$ , contains all these hops, there is at least 1 hop which has not yet been used and hence can be associated with a *virtual* assignment over  $n-1$  hops. This completes the proof.

We are now in a position to prove **Proposition 1**. By our previous result, for each possible virtual assignment of prefix  $j$  over  $p = 1, 2, \dots, K$  hops, we have a distinct hop  $k(p)$  which satisfies Eqn. (17). Summing Eqn. (17) over this set of  $K$  distinct hops, we have

$$\begin{aligned} \sum_{p=1}^K l_{k(p)} &\geq \sum_{p=1}^K f_k \gamma(\mathcal{X}, \mathcal{K}) - \sum_{p=1}^K \frac{x_j}{p}, \\ \sum_{p=1}^K l_{k(p)} &\geq F \gamma(\mathcal{X}, \mathcal{K}) - \sum_{p=1}^K \frac{x_j}{p}, \\ L - x_j &\geq F \gamma(\mathcal{X}, \mathcal{K}) - x_j(\ln K + 1) \quad (18) \\ \gamma(\mathcal{X}, \mathcal{K}) &\leq \frac{L}{F} + \frac{x_j}{F}(\ln K) \leq \frac{L}{F}(1 + \ln K), \\ \gamma(\mathcal{X}, \mathcal{K}) &\leq \gamma_o(\mathcal{X}, \mathcal{K})(1 + \ln K), \quad (19) \end{aligned}$$

where the LHS of Eqn. (18) follows from the fact that the total assigned load is not more than  $L - x_j$ . This proves **Proposition 1**.

The above analysis holds for an arbitrary ordering of the prefixes. If the prefixes are ordered in decreasing order as is the case in MAX-MIN LOAD, the bound can be improved as can be the performance of the algorithm. Our proof for this result draws on the method used in [10].

*Proposition 2:* If the prefixes are assigned in decreasing order of their traffic intensities, then,

$$\frac{\gamma(\mathcal{X}, \mathcal{K})}{\gamma_o(\mathcal{X}, \mathcal{K})} \leq \left(1 + \frac{\ln K}{2}\right).$$

**Proof:** The proof is by contradiction. Assume that the above result does not hold for some ordered set of prefixes with intensities  $\mathcal{I} = \{x_1, x_2, \dots, x_N\}$ , where

$$x_1 \geq x_2 \geq \dots \geq x_N.$$

Without loss of generality assume  $x_N$  is the intensity of the last prefix( $N$ ) assigned to the hop, which achieves the maximum load under heuristic MIN-MAX LOAD. If it is not, we can truncate the sequence up to the prefix last assigned to a hop which achieves the maximum load without affecting the maximum achieved by MIN-MAX LOAD. If the optimum for the truncated sequence is  $\gamma'_o(\mathcal{X}, \mathcal{K})$ , then  $\gamma'_o(\mathcal{X}, \mathcal{K}) \leq \gamma_o(\mathcal{X}, \mathcal{K})$  and our assumption still holds. Let as before,

$$L = \sum_{i=1}^N x_i \quad \text{and} \quad F = \sum_{k=1}^K f_k.$$

Following the exact same analysis as for the arbitrary ordering we have

$$\begin{aligned} \gamma(\mathcal{X}, \mathcal{K}) &\leq \frac{L}{F} + \frac{x_N}{F} \ln K, \\ &\leq \gamma_o(\mathcal{X}, \mathcal{K}) + \frac{x_N}{F} \ln K, \\ \frac{\gamma(\mathcal{X}, \mathcal{K})}{\gamma_o(\mathcal{X}, \mathcal{K})} &\leq 1 + \frac{x_N}{\gamma_o(x, f) \cdot F} \ln K. \end{aligned}$$

By our assumption, we have

$$\begin{aligned} 1 + \frac{x_N}{\gamma_o(\mathcal{X}, \mathcal{K}) \cdot F} \ln K &> 1 + \frac{\ln K}{2}, \\ \text{or } \frac{x_N}{F} &> \frac{\gamma_o(\mathcal{X}, \mathcal{K})}{2}. \end{aligned}$$

Note that  $\frac{x_N}{F}$  is the smallest achievable load under any split. Hence, if the above inequality, and our assumption, is to hold, we can have only one route (destination prefix) in  $\mathcal{X}$ . However it is clear from the algorithm itself that MIN-MAX LOAD achieves an optimal allocation when there is only one stream(prefix). This proves **Proposition 2**.