

# Link-State Routing with Hop-by-Hop Forwarding Can Achieve Optimal Traffic Engineering

Dahai Xu, AT&T Labs -Research

dahaixu@research.att.com

Mung Chiang

Dept. of EE, Princeton University

chiangm@princeton.edu

and

Jennifer Rexford

Dept. of CS, Princeton University

jrex@cs.princeton.edu

This paper settles an open question with a positive answer: optimal traffic engineering (or optimal multi-commodity flow) can be realized using just link-state routing protocols with hop-by-hop forwarding. Today's typical versions of these protocols, OSPF and IS-IS, split traffic evenly over shortest paths based on link weights. However, optimizing the link weights for OSPF/IS-IS to the offered traffic is a well-known NP-hard problem, and even the best setting of the weights can deviate significantly from an optimal distribution of the traffic. In this paper, we propose a new link-state routing protocol, PEFT, that splits traffic over multiple paths with an exponential penalty on longer paths. Unlike its predecessor, DEFT [Xu et al. 2007], our new protocol *provably* achieves *optimal* traffic engineering while retaining the *simplicity* of hop-by-hop forwarding. The new protocol also leads to significant reduction in the time needed to compute the best link weights. Both the protocol and the computational methods are developed in a new conceptual framework, called Network Entropy Maximization, which is used to identify the traffic distributions that are not only optimal but also realizable by link-state routing.

Categories and Subject Descriptors: C.2.2 [Network Protocols]: Routing protocols; G.1.6 [Optimization]: Convex programming

General Terms: Algorithms, Management

Additional Key Words and Phrases: Interior gateway protocol, traffic engineering, routing, OSPF, optimization, network entropy maximization

## 1. INTRODUCTION

A link-state routing protocol has three components. First is *weight computation*: the network-management system computes a set of link weights through a periodic and centralized optimization. Second is *traffic splitting*: each router uses the link weights to decide traffic splitting ratios among its outgoing links for every destination. Third is *packet forwarding*: each router independently decides which outgoing link to forward a packet based only on its destination prefix, in order to realize the desired traffic splitting. The popularity of link-state protocols can be attributed to their ease of management; in particular, each router's traffic-splitting decision is

---

A preliminary short version of this paper was presented under the same title in Proc. IEEE INFOCOM 2008 - The Conference on Computer Communications. The work was mainly done when Xu was in Dept. of EE, Princeton University.

Table I. Comparison of various TE schemes (new contributions in *italics*).

	Commodity Routing	Link-State Routing	
		OSPF	PEFT
Traffic Splitting	Arbitrary	Even among shortest paths	Exponential
Scalability	Low	High	High
Optimal TE	Yes	No	<i>Yes</i>
Complexity Class	Convex Optimization	NP Hard	<i>Convex Optimization</i>

made autonomously based only on the link weights without further assistance from the network-management system, and each packet’s forwarding decision is made in a hop-by-hop fashion without end-to-end tunneling.

Such simplicity is often believed to come at the expense of optimality. In a procedure known as Traffic Engineering (TE), network operators minimize a convex cost function of the link loads, by tuning the link weights to be used by the routers. With Open Shortest Path First (OSPF) or Intermediate System-Intermediate System (IS-IS), the major variants of link-state protocol in use today, computing the right link weights is NP-hard and even the best setting of the weights can deviate significantly from optimal TE [Fortz and Thorup 2004]. Contrary to some popular belief, the optimal TE method in [Wang et al. 2001] is *not* distributed link-state routing with hop-by-hop forwarding, and the following question remains open: can a link-state protocol with hop-by-hop forwarding achieve optimal TE? This paper shows that the answer is in fact positive, by developing a new link-state protocol, Penalizing Exponential Flow-splitting (PEFT), proving that it achieves optimal TE, and demonstrating that link weight computation for PEFT is highly efficient in theory and in practice.

In PEFT, packet forwarding is just the same as OSPF: destination-based and hop-by-hop. The key difference is in traffic splitting. OSPF splits traffic evenly among the shortest paths, and PEFT splits traffic along all paths but penalizes longer paths (i.e., paths with higher sums of link weights) exponentially. While this is a difference in how link weights are *used* in the routers, it also enables a change in how link weights are *computed* by the operator. It turns out that using link weights in the PEFT way enables optimal traffic engineering. Using the Abilene topology and traffic traces, we observe a 15% increase in the efficiency of capacity utilization by PEFT over OSPF. Furthermore, exponential penalty in traffic splitting is the *only* penalty that can lead to this optimality result. The corresponding best link weights for PEFT can be efficiently computed: as efficiently as solving a linearly constrained concave maximization and much faster than the existing weight computation heuristics for OSPF.

Clearly, if the complexity of managing a routing protocol were not a concern, other approaches could be used to achieve optimal TE. One possibility is multi-commodity-flow type of routing, where an optimal traffic distribution is realized by dividing an arbitrary fraction of traffic over many paths. This can be supported by the forwarding mechanism in Multi-Protocol Label Switching (MPLS) [Awduche 1999]. However, optimality then comes with a cost for establishing many end-to-end *tunnels* to forward packets. Second, other studies explored more flexible ways to split traffic over shortest paths [Wang et al. 2001; Sridharan et al. 2005; Srivastava et al. 2005], but these solutions do not enable routers to *independently* compute

the flow-splitting ratios from link weights. Instead, a central management system must compute and configure the traffic-splitting ratios, and update them when the topology changes, sacrificing the main benefit of running a distributed link-state routing protocol. Clearly, there is a tension between optimal but complex routing or forwarding methods and the simple but to-date suboptimal link-state routing with hop-by-hop forwarding. Recent works [Fong et al. 2005; Xu et al. 2007] attempted to attain optimality and simplicity simultaneously, but in contrast to the current paper, neither proved optimality for TE nor developed sufficiently fast methods for computing link weights. A summary is provided in Table I.

There are several new ideas in this paper that enable a proof of optimality and a much faster computation beyond, for example, the theory and algorithm in DEFT [Xu et al. 2007]. One of these ideas is to develop the traffic splitting as well as the weight computation methods from the conceptual framework of Network Entropy Maximization (NEM). As a proof technique and intermediate step of protocol development, we will construct an optimization called NEM that is solved neither by the operator nor by the routers, but by us, the protocol developers. The optimality condition of NEM reveals the structure of hop-by-hop forwarding and is later used to guide both the router’s traffic splitting and the operator’s weight computation. In short, it turns out that a certain notion of entropy can identify those optimal traffic distributions that can be realized by link-state protocols.

The general principle of entropy maximization has been used to solve other networking problems, e.g., [Blunden 1967; Tomlin and Tomlin 1968; Tomlin 2003; Agrawal et al. 2005]. This is the first work connecting entropy with IP routing. In addition, as we summarize later in Table V, our NEM framework for routing has interesting parallels with recent work relating TCP congestion control to Network Utility Maximization (NUM) [Kelly et al. 1998; Yäiche et al. 2000; Low 2003; Srikant 2004].

The rest of the paper is organized as follows. Background on optimal traffic engineering is introduced in Sec. 2. The theory of Network Entropy Maximization in Sec. 3 leads to the routing protocol PEFT in Sec. 4 and the associated link weight computation algorithm in Sec. 5. Extensive numerical experiments are then summarized in Sec. 6. The interesting and general framework of Network Entropy Maximization is further discussed in Sec. 7. Differences from our previous results [Xu et al. 2007] are summarized in Sec. 8, before we conclude with further observations and extensions in Sec. 9. The key notation used in this paper is shown in Table II.

## 2. BACKGROUND ON OPTIMAL TE

### 2.1 Definitions of Optimality

Consider a wireline network as a directed graph  $G = (\mathbb{V}, \mathbb{E})$ , where  $\mathbb{V}$  is the set of nodes (where  $N = |\mathbb{V}|$ ),  $\mathbb{E}$  is the set of links (where  $E = |\mathbb{E}|$ ), and link  $(u, v)$  has capacity  $c_{u,v}$ . The offered traffic is represented by a traffic matrix  $D(s, t)$  for source-destination pairs indexed by  $(s, t)$ .

The load  $f_{u,v}$  on each link  $(u, v)$  depends on how the network decides to route the traffic. An objective function enables quantitative comparisons between different routing solutions in terms of the load on the links. Traffic engineering usually

Table II. Summary of Key Notation

Notation	Meaning
$D(s, t)$	Traffic demand from source $s$ to destination $t$
$c_{u,v}$	Capacity of link $(u, v)$
$f_{u,v}$	Flow on link $(u, v)$
$\tilde{c}_{u,v}$	Necessary capacity of link $(u, v)$
$f_{u,v}^t$	Flow on link $(u, v)$ destined to node $t$
$f_u^t$	Total incoming flow (destined to $t$ ) at $u$
$w_{u,v}$	Weight assigned to link $(u, v)$
$w_{min}$	Lower bound of all link weights
$d_u^t$	The shortest distance from node $u$ to node $t$ . $d_t^t = 0$
$h_{u,v}^t$	Gap of shortest distance, $h_{u,v}^t \triangleq d_v^t + w_{u,v} - d_u^t$
$\Gamma(h_{u,v}^t)$	Traffic splitting function

considers a link-cost function  $\Phi(f_{u,v}, c_{u,v})$  that is a increasing function of  $f_{u,v}$ .

For example,  $\Phi(f_{u,v}, c_{u,v})$  can be the link utilization  $f_{u,v}/c_{u,v}$ , and the objective of traffic engineering can be to minimize  $\max_{(u,v) \in \mathbb{E}} \Phi(f_{u,v}, c_{u,v})$ .

As another example, let  $\Phi(f_{u,v}, c_{u,v})$  be a piecewise-linear approximation of the M/M/1 delay formula [Fortz and Thorup 2000], e.g.,

$$\Phi(f_{u,v}, c_{u,v}) = \begin{cases} f_{u,v} & f_{u,v}/c_{u,v} \leq 1/3 \\ 3f_{u,v} - 2/3 c_{u,v} & 1/3 \leq f_{u,v}/c_{u,v} \leq 2/3 \\ 10f_{u,v} - 16/3 c_{u,v} & 2/3 \leq f_{u,v}/c_{u,v} \leq 9/10 \\ 70f_{u,v} - 178/3 c_{u,v} & 9/10 \leq f_{u,v}/c_{u,v} \leq 1 \\ 500f_{u,v} - 1468/3 c_{u,v} & 1 \leq f_{u,v}/c_{u,v} \leq 11/10 \\ 5000f_{u,v} - 16318/3 c_{u,v} & 11/10 \leq f_{u,v}/c_{u,v}, \end{cases} \quad (1)$$

and the objective is to minimize  $\sum_{(u,v)} \Phi(f_{u,v}, c_{u,v})$ .

More generally, we use “ $\Phi(\{f_{u,v}, c_{u,v}\})$ ” to represent any increasing and convex objective function. The optimality of traffic engineering is with respect to this objective function.

At this point we can already observe that there is a “gap” between the objective of TE and the mechanism of link-state routing. Optimality is defined directly in terms of the traffic flows, whereas link-state protocols represent the paths indirectly in terms of link weights. Bridging this gap is one of the challenges that have prevented researchers from achieving optimal traffic engineering using link-state routing thus far.

## 2.2 Optimal TE Via Multi-Commodity Flow

Consider the following convex optimization problem: minimizing the TE cost function over flow conservation and link capacity constraints:

COMMODITY:

$$\min \Phi(\{f_{u,v}, c_{u,v}\}) \quad (2a)$$

$$\text{s.t.} \quad \sum_{v:(s,v) \in \mathbb{E}} f_{s,v}^t - \sum_{u:(u,s) \in \mathbb{E}} f_{u,s}^t = D(s,t), \forall s \neq t \quad (2b)$$

$$f_{u,v} \triangleq \sum_{t \in \mathbb{V}} f_{u,v}^t \leq c_{u,v}, \forall (u,v) \quad (2c)$$

$$\text{vars.} \quad f_{u,v}^t, f_{u,v} \geq 0. \quad (2d)$$

The above multi-commodity problem <sup>1</sup> can be readily solved efficiently, where the flow destined to a single destination is treated as a commodity, and  $f_{u,v}^t$  is amount of flow on link  $(u,v)$  destined to node  $t$  <sup>2</sup>.

The resulting solution, however, may not be realizable through link-state routing and hop-by-hop forwarding. Indeed, for a network with  $N$  nodes and  $E$  links, the multi-commodity-flow solution may require up to  $O(N^2E)$  tunnels, i.e., explicit routing (see Appendix E), making it difficult to scale. In contrast, link-state routing is much simpler, requiring only  $O(E)$  parameters (i.e., one per link).

Furthermore, while it is true that, from the solution of the COMMODITY problem, a set of link weights can be computed such that all the commodity flow will be forwarded along the shortest paths [Sridharan et al. 2005; Wang et al. 2001], the flow-splitting ratios among these shortest paths are *not* related to the *link weights*, forcing the operator to specify up to  $O(NE)$  *additional* parameters (one parameter on each link for each destination) as the flow-splitting ratios for all the routers.

Henceforth, we use the following phrases: optimal traffic engineering, optimal multi-commodity flow (2) and optimal distribution of traffic, interchangeably. The rest of this paper shows that optimal traffic engineering can, in fact, be achieved using only  $E$  link weights.

### 3. THEORETICAL FOUNDATION: NEM

In this section, we present the theory of realizing optimal TE with link-state protocols. We first compute the minimal load that each link must carry to achieve optimal traffic distribution, then examine all the traffic splitting choices subject to necessary (minimal) link capacities. The traffic splitting configuration that is realizable with hop-by-hop forwarding can be picked out by exploiting a property it has: maximizing a weighted sum of the entropies of traffic splitting vectors. In addition, the corresponding link weights can be found by solving the new optimization problem using the gradient descent algorithm. It is important to realize

<sup>1</sup>We first remark that solving this COMMODITY problem is only an intermediate step in the proof, the actual PEFT protocol in Section 4 will not be implementing a multicommodity flow based routing with end-to-end tunneling. Another clarifying remark is that while we will later show that PEFT link weight computation is as easy as solving a convex optimization, that optimization is not this well-known COMMODITY problem.

<sup>2</sup>If the objective  $\Phi(\{f_{u,v}, c_{u,v}\})$  is not a strictly increasing function of link flow  $f_{u,v}$  (like minimizing the maximum link utilization), the optimal solution of COMMODITY problem (2) may contain flow cycles. To prevent bandwidth waste, we can eliminate flow cycles in the optimal routing with a  $O(E \log N)$ -time algorithm for each commodity [Sleator and Tarjan 1983]. The cycle-free property is important in designing link-state routing as demonstrated later in Sec. 5.

that the proposed NEM framework developed in this section is used to *design* the protocol—the NEM problem itself is *not* solved by the operator or routers. It is constructed as a proof technique and an intermediate step towards the results in Sections 4 and 5.

### 3.1 Necessary Capacity

Given the traffic matrix and the objective function, the solution to the COMMODITY problem (2) provides the optimal distribution of traffic. We represent the resulting flow on each link  $(u, v)$  as the *necessary capacity*  $\tilde{c}_{u,v} \triangleq f_{u,v}$  (or  $\tilde{\mathbf{c}}$  as a vector). The necessary capacity is a minimal<sup>3</sup> set of link capacities to realize optimal traffic engineering.

There could be numerous ways of traffic splitting that realize optimal TE. If we replace link capacity  $c_{u,v}$  in COMMODITY (2) with the necessary capacity  $\tilde{c}_{u,v}$ <sup>4</sup>, we are free to impose another objective function to pick out a particular optimal solution to the original problem. A key challenge here is to design a new objective function, purely for the purpose of protocol development, such that the resulting routing of flow can be realized *distributively with link-state routing protocols and hop-by-hop forwarding*.

### 3.2 Network Entropy Maximization

Denote  $P_{s,t}$  as the set of paths from  $s$  to  $t$  (repeated nodes are allowed), and  $x_{s,t}^i$  as the probability (fraction) of forwarding a packet of demand  $D(s, t)$  to the  $i$ -th path ( $P_{s,t}^i$ ). Obviously,  $\sum_i x_{s,t}^i = 1$ . If we require the probabilities of using two paths to be same as long as they are of the same length, (see Appendix C for details), to be realized with hop-by-hop forwarding, the values of  $x_{s,t}^i$  should satisfy (3) below where  $w_{u,v}$  is the weight assigned to link  $(u, v)$ ,  $K_{P_{s,t}^i}^{(u,v)}$  is the number of times  $P_{s,t}^i$  passes through link  $(u, v)$  ( $P_{s,t}^i$  can contain cycles) and  $g(\cdot)$  is a known function for all the routers.

$$\frac{x_{s,t}^i}{x_{s,t}^j} = \frac{g\left(\sum_{(u,v) \in \mathbb{E}} K_{P_{s,t}^i}^{(u,v)} w_{u,v}\right)}{g\left(\sum_{(u,v) \in \mathbb{E}} K_{P_{s,t}^j}^{(u,v)} w_{u,v}\right)}. \quad (3)$$

We find that the set of values of  $x_{s,t}^i$  satisfying (3) maximizes a “network entropy” defined as follows. Consider the entropy function  $z(x_{s,t}^i) = -x_{s,t}^i \log x_{s,t}^i$  for source-destination pair  $(s, t)$ . The weighted sum,  $\sum_{s,t} \left( D(s, t) \sum_{i \in P_{s,t}} z(x_{s,t}^i) \right)$ , is defined as the network entropy.<sup>5</sup>

<sup>3</sup>But may not be the minimum capacity.  $\tilde{\mathbf{c}}$  is minimal if  $\nexists \tilde{\mathbf{c}}' : \tilde{\mathbf{c}}' \neq \tilde{\mathbf{c}} \wedge \tilde{\mathbf{c}}' \preceq \tilde{\mathbf{c}}$  whereas  $\tilde{\mathbf{c}}$  is the minimum if  $\forall \tilde{\mathbf{c}}' : \tilde{\mathbf{c}} \preceq \tilde{\mathbf{c}}'$ .

<sup>4</sup>The link cost is still defined in terms of the original link capacity, i.e., link utilization or cost will not be changed due to the use of necessary capacity.

<sup>5</sup>The physical interpretation of entropy for IP routing and the uniqueness of choosing the entropy function to pick out the right flow distributions are presented in Appendix B and C, respectively.

Now we define the Network Entropy Maximization (**NEM**) problem under the necessary capacity constraints as follows:

*NEM*:

$$\max \sum_{s,t} \left( D(s,t) \sum_{i \in P_{s,t}} z(x_{s,t}^i) \right) \quad (4a)$$

$$\text{s.t.} \quad \sum_{s,t,i} D(s,t) K_{P_{s,t}^i}^{(u,v)} x_{s,t}^i \leq \tilde{c}_{u,v}, \forall (u,v) \quad (4b)$$

$$\sum_i x_{s,t}^i = 1, \forall s,t \quad (4c)$$

$$\text{vars.} \quad x_{s,t}^i \geq 0. \quad (4d)$$

From the optimal solution of the COMMODITY problem, we know the feasibility set of NEM is non-empty. For a concave maximization over a non-empty, compact constraint set, there exist globally optimal solutions to NEM.

### 3.3 Solve NEM by Dual Decomposition

We will connect the characterization of optimal solutions to NEM that are realizable with hop-by-hop forwarding to exponential penalty. Towards that end, and to provide a foundation for link weight computation in Sec. 5, we first investigate the Lagrange dual problem of NEM and a dual-gradient-based solution.

Denote dual variables for constraints (4b) as  $\lambda_{u,v}$  for link  $(u,v)$  (or  $\boldsymbol{\lambda}$  as a vector). We first write the Lagrangian  $L(\boldsymbol{x}, \boldsymbol{\lambda})$  associated with the NEM problem

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{\lambda}) &= \sum_{s,t} \left( D(s,t) \sum_{i \in P_{s,t}} z(x_{s,t}^i) \right) \\ &\quad - \sum_{(u,v) \in \mathbb{E}} \lambda_{u,v} \left( \sum_{s,t,i} D(s,t) K_{P_{s,t}^i}^{(u,v)} x_{s,t}^i - \tilde{c}_{u,v} \right). \end{aligned} \quad (5)$$

The Lagrange dual function is

$$\begin{aligned} Q(\boldsymbol{\lambda}) = \max_{\substack{\mathbf{1} \succeq \boldsymbol{x} \succeq \mathbf{0} \\ \|\boldsymbol{x}_{s,t}\| = \mathbf{1}}} L(\boldsymbol{x}, \boldsymbol{\lambda}), \end{aligned} \quad (6)$$

where  $\mathbf{0}$  and  $\mathbf{1}$  are the vectors whose elements are all zeros and ones respectively and  $\boldsymbol{x}_{s,t}$  is the vector of  $x_{s,t}^i$ .

The dual problem is formulated as

$$\begin{aligned} \min Q(\boldsymbol{\lambda}) \\ \text{s.t.} \quad \boldsymbol{\lambda} \succeq \mathbf{0}. \end{aligned} \quad (7)$$

To solve the dual problem, we first consider problem (6). The maximization of the Lagrangian over  $\boldsymbol{x}$  can be solved as a TRAFFIC-DISTRIBUTION problem (8):

TRAFFIC-DISTRIBUTION:

$$\max \sum_{(u,v) \in \mathbb{E}} \lambda_{u,v} \tilde{c}_{u,v} + \sum_{s,t} \left( D(s,t) \sum_{i \in P_{s,t}} z(x_{s,t}^i) \right) \quad (8a)$$

$$- \sum_{(u,v) \in \mathbb{E}} \lambda_{u,v} \left( \sum_{s,t,i} D(s,t) K_{P_{s,t}^i}^{(u,v)} x_{s,t}^i \right)$$

$$\text{s.t. } \sum_i x_{s,t}^i = 1. \quad (8b)$$

Then, the dual problem (7) can be solved by using the gradient descent algorithm as follows for iterations indexed by  $q$ ,

$$\begin{aligned} & \lambda_{u,v}(q+1) \\ &= \left[ \lambda_{u,v}(q) - \alpha(q) \left( \tilde{c}_{u,v} - \sum_{s,t,i} D(s,t) K_{P_{s,t}^i}^{(u,v)} x_{s,t}^i(q) \right) \right]^+ \\ &= [\lambda_{u,v}(q) - \alpha(q) (\tilde{c}_{u,v} - f_{u,v}(q))]^+, \quad \forall (u,v) \in \mathbb{E}. \end{aligned} \quad (9)$$

where  $\alpha(q) > 0$  is the step size,  $x_{s,t}^i(q)$  are solutions of the TRAFFIC-DISTRIBUTION problem (8) for a given  $\lambda(q)$ , and  $f_{u,v}(q)$  is the total flow on link  $(u,v)$ .

After the above dual decomposition, the following result can be proved with standard convergence analysis for gradient algorithms [Bertsekas 1999]:

**Lemma 1.** *By solving the TRAFFIC-DISTRIBUTION problem for the NEM problem and the dual variable update (9),  $\lambda(q)$  converge to the optimal dual solutions  $\lambda^*$  and the corresponding primal variables  $\mathbf{x}^*$  are the globally optimal primal solutions of (4).*

PROOF. See Appendix D.  $\square$

### 3.4 Solve TRAFFIC-DISTRIBUTION Problem

Note that, the TRAFFIC-DISTRIBUTION problem is also separable, i.e., the traffic splitting for each demand across its paths is independent of the others since they are not coupled together with link capacity constraint (4b). So we can solve a subproblem (10) below for each demand  $D(s,t)$  separately:

DEMAND-DISTRIBUTION for  $D(s,t)$ :

$$\max D(s,t) \sum_{i \in P_{s,t}} z(x_{s,t}^i) \quad (10a)$$

$$- \sum_{(u,v) \in \mathbb{E}} \lambda_{u,v} \left( \sum_i D(s,t) K_{P_{s,t}^i}^{(u,v)} x_{s,t}^i \right)$$

$$\text{s.t. } \sum_i x_{s,t}^i = 1. \quad (10b)$$

We write the Lagrangian associated with the DEMAND-DISTRIBUTION subproblem in (11).

$$\begin{aligned}
 & L^r(\mathbf{x}_{s,t}, \mu_{s,t}) \\
 = & \left( D(s,t) \sum_{i \in P_{s,t}} z(x_{s,t}^i) \right) - \mu_{s,t} (\sum_i x_{s,t}^i - 1) \\
 & - \sum_{(u,v) \in \mathbb{E}} \lambda_{u,v} (\sum_i D(s,t) K_{P_{s,t}^i}^{(u,v)} x_{s,t}^i)
 \end{aligned} \tag{11}$$

where  $\mu_{s,t}$  is the Lagrangian variable associated with (10b).

According to Karush-Kuhn-Tucker (KKT) conditions<sup>6</sup> [Boyd and Vandenberghe 2004], at the optimal solution of the DEMAND-DISTRIBUTION subproblem, we have

$$z'(x_{s,t}^{i*}) - \sum_{(u,v)} K_{P_{s,t}^i}^{(u,v)} \lambda_{u,v} - \frac{\mu_{s,t}^*}{D(s,t)} = 0. \tag{12}$$

For the entropy function,  $z(x) = -x \log x$ ,  $z'(x) = -1 - \log x$ , we have

$$x_{s,t}^{i*} = e^{-\left(\sum_{(u,v)} K_{P_{s,t}^i}^{(u,v)} \lambda_{u,v} + \frac{\mu_{s,t}^*}{D(s,t)} + 1\right)}. \tag{13}$$

where  $x_{s,t}^{i*}, \mu_{s,t}^*$  are the values of the  $x_{s,t}^i, \mu_{s,t}$  respectively at the optimal solution.

Then for two paths  $i, j$  from  $s$  to  $t$ , we have

$$\frac{x_{s,t}^{i*}}{x_{s,t}^{j*}} = \frac{e^{-\left(\sum_{(u,v)} K_{P_{s,t}^i}^{(u,v)} \lambda_{u,v}\right)}}{e^{-\left(\sum_{(u,v)} K_{P_{s,t}^j}^{(u,v)} \lambda_{u,v}\right)}}. \tag{14}$$

We pause to examine the engineering implications of (14). If we use  $\lambda_{u,v}$  as the weight  $w_{u,v}$  for link  $(u,v)$ , the probability of using path  $P_{s,t}^i$  is inversely proportional to the exponential value of its path length. It is important to observe at this point that, since (14) has no factor of  $\mu_{s,t}^*$ , an intermediate router can ignore the source of the packet when making forwarding decisions. Equally importantly, from (9), in iteration  $q$ , the procedure of link weight updating does not need the values of  $x_{s,t}^i(q)$ . Instead, it just needs  $f_{u,v}(q)$ , the aggregated bandwidth usage. We will show how to calculate it efficiently in Sec. 5.2.

Now, combining the optimality results in Sec. 2.2 and Lemma 1 with the existence of (14), we have

**Theorem 1.** *Optimal traffic engineering (i.e., the optimal multi-commodity flow) for a given traffic matrix can be realized with link weights using exponential flow splitting (14).*

#### 4. A NEW LINK-STATE ROUTING PROTOCOL: PEFT

In this section, we translate the theoretical results in Sec. 3 into a new link-state routing protocol run by routers. Each router makes an *independent* decision on how to forward traffic to a destination (i.e., flow-splitting ratios) among its outgoing links using *only* the link weights. We first present PEFT from (14), and summarize the

<sup>6</sup>KKT is a necessary condition. But NEM must have a global optimal solution, thus we must have one set of  $x_{s,t}^{i*}, \mu_{s,t}^*$  for (13).

notation of traffic-splitting function [Xu et al. 2007] for calculating flow-splitting ratios. Then for a PEFT flow, we show an efficient way to calculate its traffic-splitting function, which can be approximated to further simplify the computation of traffic splitting ratios in practice.

#### 4.1 PEFT

Based on (14), we propose a new link-state routing protocol, called Penalizing Exponential Flow-splitting (PEFT). The fraction of the traffic (from  $u$  to  $t$ ) distributed across the  $i$ -th path (or probability of forwarding a packet),  $x_{u,t}^i$ , is inversely proportional to the exponential value of its path length  $p_{u,t}^i \triangleq \sum_{(u',v) \in \mathbb{E}} K_{P_{u,t}^i}^{(u',v)} w_{u',v}$ , as shown in (15).

$$\text{PEFT: } x_{u,t}^i = \frac{e^{-p_{u,t}^i}}{\sum_j e^{-p_{u,t}^j}}. \quad (15)$$

Theorem 1 in Sec. 3 shows PEFT can achieve optimal TE. A PEFT flow can be realized with hop-by-hop forwarding. For the sample network in Fig. 1, for the two paths from  $s$  to  $t$ ,  $s \rightarrow u \rightarrow a \rightarrow t$  and  $s \rightarrow u \rightarrow b \rightarrow t$ , and two paths from  $u$  to  $t$ , the flows on them for PEFT (15) satisfy (16).

$$f_{s \rightarrow u \rightarrow a \rightarrow t} : f_{s \rightarrow u \rightarrow b \rightarrow t} = f_{u \rightarrow a \rightarrow t} : f_{u \rightarrow b \rightarrow t} \quad (16)$$

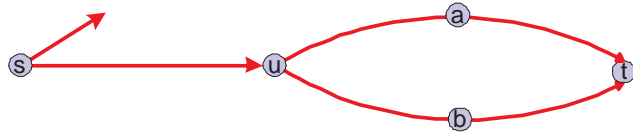


Fig. 1. Realize a PEFT flow using hop-by-hop forwarding

Therefore, router  $u$  can treat the packets from different sources (e.g.  $s$  or  $u$ ) equally by forwarding them among the outgoing links with precalculated splitting ratios. Formally, we have the following

**Proposition 1.** The PEFT flow for a set of link weights can be realized with hop-by-hop forwarding.

**PROOF.** For the traffic from  $s$  to  $t$ , assume  $P_i(s, u, t)$  is the set of all the paths (having flow from  $s$  to  $t$ ) that share  $i$ , a sub-path (segment) from  $s$  to  $u$ , and  $P(u, t)$  is the set of all paths having flow from  $u$  to  $t$ . From PEFT (15), the traffic splitting ratio of the flows on  $P_i(s, u, t)$  is equal to that of  $P(u, t)$ . The equality holds for every set of  $P_i(s, u, t)$  for a PEFT flow. Thus, the flow can be realized with hop-by-hop forwarding.  $\square$

As a link-state routing protocol, we need to define the traffic splitting function for PEFT as follows.

## 4.2 Review: Traffic Splitting Function

The notation of traffic-splitting (allocation) function was introduced in [Xu et al. 2007] to succinctly describe link-state routing protocols. In a directed graph, each unidirectional link  $(u, v)$  has a single, configurable weight  $w_{u,v}$ . Based on a complete view of the topology and link weights, a router can compute the shortest distance  $d_u^t$  from any node  $u$  to node  $t$ ;  $d_v^t + w_{u,v}$  represents the distance from  $u$  to  $t$  when routed through neighboring node  $v$ . *Shortest distance gap*,  $h_{u,v}^t$ , is defined as  $d_v^t + w_{u,v} - d_u^t$ , which is always greater than or equal to 0. Then,  $(u, v)$  lies on a shortest path to  $t$  if and only if  $h_{u,v}^t = 0$ . Traffic-splitting function  $(\Gamma(h_{u,v}^t))$  indicates the relative amount of traffic destined to  $t$  that node  $u$  will forward via outgoing link  $(u, v)$ <sup>7</sup>. Let  $f_u^t$  denote the total incoming flow (destined to  $t$ ) at node  $u$  (including the passing-through flow and self-originated flow). The total outgoing flow of traffic (destined to  $t$ ) traversing link  $(u, v)$ ,  $f_{u,v}^t$ , can be computed as follows:

$$f_{u,v}^t = f_u^t \frac{\Gamma(h_{u,v}^t)}{\sum_{(u,j) \in \mathbb{E}} \Gamma(h_{u,j}^t)}. \quad (17)$$

Consistent with hop-by-hop forwarding,  $u$  splits the traffic over the outgoing links without regard to the source node or the incoming link where the traffic arrived.

## 4.3 Exact Traffic Splitting Function for PEFT

The traffic splitting function for PEFT can be calculated by each node autonomously and in polynomial time. From the definition of PEFT (15), more traffic should be sent along an outgoing link used by more paths and the paths should be treated differently based on their path lengths. To compute the traffic splitting on each outgoing link, we first define a positive real number  $\Upsilon_u^t$ , possibly interpretable as the “equivalent number” of shortest paths from node  $u$  to destination  $t$ , and let  $\Upsilon_t^t \triangleq 1$ .

For a PEFT flow, we have

$$\Upsilon_u^t \triangleq \sum_{i \in P_{u,t}} e^{-(p_{u,t}^i - d_u^t)} \quad (18a)$$

$$\begin{aligned} &= \sum_{(u,v) \in \mathbb{E}} \left( \sum_{j \in P_{v,t}} e^{-(p_{v,t}^j + w_{u,v} - d_u^t - d_v^t + d_v^t)} \right) \\ &= \sum_{(u,v) \in \mathbb{E}} \left( e^{-(d_v^t + w_{u,v} - d_u^t)} \sum_{j \in P_{v,t}} e^{-(p_{v,t}^j - d_v^t)} \right) \\ &= \sum_{(u,v) \in \mathbb{E}} \left( e^{-h_{u,v}^t} \Upsilon_v^t \right) \end{aligned} \quad (18b)$$

<sup>7</sup>For example, the traffic-splitting function for even-splitting across shortest paths (e.g., OSPF) is

$$\Gamma_O(h_{u,v}^t) = \begin{cases} 1 & \text{if } h_{u,v}^t = 0, \\ 0 & \text{if } h_{u,v}^t > 0. \end{cases}$$

The recursive relationship represented in (18b)<sup>8</sup> can be used in the following way:  $e^{-h_{u,v}^t} \Upsilon_v^t$  is an “equivalent number” of shortest paths from  $u$  to  $t$  for those paths passing through link  $(u, v)$  and the router should distribute the traffic from  $u$  on link  $(u, v)$  in proportion to  $e^{-h_{u,v}^t} \Upsilon_v^t$ . Then we have an exact traffic splitting function<sup>9</sup> for PEFT at link  $(u, v)$ :

$$\Gamma_{PX}(h_{u,v}^t) = \Upsilon_v^t e^{-h_{u,v}^t} \quad (19)$$

To enable hop-by-hop forwarding, each router needs to independently calculate  $\Gamma_{PX}(h_{u,v}^t)$  for all node pairs. Then each router first computes the all-pairs shortest paths, using, e.g., the Floyd-Warshall algorithm with time complexity  $O(N^3)$  [Cormen et al. 1990], and calculates the values of  $e^{-h_{u,v}^t}$ . Then for each destination  $t$ , to compute the values of  $\Upsilon_u^t$ , each router needs to solve  $N$  linear equations (18b), which requires  $O(N^3)$  time [Cormen et al. 1990]. Thus the total complexity is  $O(N^4)$ .

#### 4.4 A Detour: Traffic Splitting Function for “Downward PEFT”

To prevent cycles in link-state routing, packets are usually forwarded along a “downward path” where the next hop is closer to destination. This inspires the following *Downward PEFT*, whose traffic splitting function is  $\Gamma_{PD}(h_{u,v}^t)$ <sup>10</sup>:

$$\Gamma_{PD}(h_{u,v}^t) = \begin{cases} \Upsilon_v^t e^{-h_{u,v}^t} & \text{if } d_u^t > d_v^t, \\ 0 & \text{otherwise.} \end{cases} \quad (20)$$

$\Gamma_{PD}(h_{u,v}^t)$  can approximate  $\Gamma_{PX}(h_{u,v}^t)$  and further simplify the computation of  $\Upsilon_u^t$  and traffic splitting as discussed below and utilized in Sec. 5.3.

We consider each destination  $t$  independently. After temporarily removing link  $(u, v)$  where  $d_u^t \leq d_v^t$  since there is no flow on it, we get an acyclic network and do topological sorting on the remaining network. Proceeding through the nodes  $u$  in *increasing* topological order (starting with destination  $t$ ), we compute the value of  $\Upsilon_u^t$  using (18b). For each destination, topology sorting requires  $O(N + E)$  time, and summarizing the  $\Upsilon_u^t$  across the outgoing links requires  $O(N + E)$  time. Thus, the total time complexity to calculate  $\Upsilon_u^t$  is  $O(N^3 + N(N + E)) = O(N^3)$ .

In general, “*Downward PEFT*” does not provably achieve optimal TE, in contrast to PEFT, although it comes extremely close to optimal TE in practice, with the associated link weight computation even faster than that for PEFT. In the case where the lower bound of all link weights,  $w_{min}$ , is large enough, the downward PEFT is same as PEFT<sup>11</sup>.

<sup>8</sup>Allowing for paths with cycles is required for the recursive derivation of (18b) (i.e. from  $\sum_{j \in P_{v,t}} e^{-(p_{v,t}^j - d_v^t)}$  to  $\Upsilon_v^t$ ). Consider a simple example with two unidirectional links between  $u$  and  $v$  (i.e.  $(u, v)$  and  $(v, u)$ ) and  $P_{u,t}^i$  and  $P_{v,t}^i$  are the sets of the paths to  $t$  from  $u$  and  $v$  respectively. Then the concatenation of link  $(u, v)$  and  $P_{v,t}^i$ , which may create paths with cycle, is a subset of  $P_{u,t}^i$ . Similarly, the concatenation of link  $(v, u)$  and  $P_{u,t}^i$  is a subset of  $P_{v,t}^i$ . However, in practice, only cycle-free paths will be used because longer paths are exponentially penalized.

<sup>9</sup>P in the subscript emphasizes that the calculation of traffic splitting considers the paths towards destination, and X means the exactness.

<sup>10</sup>D in the subscript emphasizes “downward”.

<sup>11</sup>For link  $(u, v)$ , if the shortest distance to  $t$  of  $u$ ,  $d_u^t \leq d_v^t$ , then  $h_{u,v}^t = d_v^t + w_{u,v} - d_u^t \geq w_{u,v}$  and

## 5. LINK WEIGHT COMPUTATION FOR PEFT

The last section described traffic splitting by each router under PEFT. A new way to use link weights also means the network operator needs a new way to compute, centrally and off-line, the optimal link weights. It turns out that the NP-hard problem of link weight computation in OSPF can be turned into a convex optimization when link weights are used by PEFT. To do that, we will convert the iterative method of solving the NEM problem in Sec. 3 into a simple and efficient algorithm. We first present an algorithm that iteratively chooses a tentative set of link weights and evaluates the corresponding traffic distribution by simulating the PEFT traffic splitting run by the routers. From Theorem 1, the algorithm is guaranteed to converge to a set of link weights, which realizes optimal TE with PEFT. To further speed up the calculation, the traffic distribution with PEFT for each iteration can be *approximated* with downward PEFT. The simulation in Sec. 6 shows that such an approximation is very close to optimal and provides substantial speedup in practice.

### 5.1 Algorithm Framework for Optimizing Link Weights

The iterative algorithm consists of two main parts:

- (1) Computing the optimal traffic distribution (necessary capacities) for a given traffic matrix by solving the COMMODITY problem (2).
- (2) Computing the link weights that would achieve the optimal traffic distribution.

The second step uses the optimal traffic distribution found in the first step as input, and need not consider the objective function ( $\Phi(\{f_{u,v}, c_{u,v}\})$ ) any further. Starting with an initial setting of link weights, the algorithm (see Algorithm 1) repeatedly updates the link weights until the load on each link is the same as the necessary capacity. Each setting of the link weights corresponds to a particular way of splitting the traffic over a set of paths. The *Traffic\_Distribution* procedure computes the resulting link loads  $f_{u,v}$ , based on the traffic matrix. Then, the *Link\_Weight\_Update* procedure (see Algorithm 2) increases the weight of each link  $(u, v)$  linearly if the traffic exceeds the necessary capacity, or decreases it otherwise. The parameter  $\alpha$  is a positive step size, which can be constant or dynamically adjusted; we find that setting  $\alpha$  to the reciprocal of the maximum necessary link capacity ( $\frac{1}{\max c_{u,v}}$ ) performs well in practice. Algorithm 1 is guaranteed to converge to the global optimal solution as stated in Lemma 1.

In terms of computational complexity, we know that COMMODITY can be solved efficiently. The complexity of Algorithm 2 is  $O(E)$ . The remaining question is how to solve the subproblem *Traffic\_Distribution*( $\mathbf{w}$ ) efficiently.

### 5.2 Compute Traffic Distribution with PEFT

To compute the traffic distribution for PEFT, we should first compute the shortest paths between each pair of nodes and all the values  $\Gamma_{PX}(h_{u,v}^t)$  as in Sec. 4.3. Computing the resulting distribution of traffic is complicated by the fact that  $\Gamma_{PX}(\cdot)$

---

$\Gamma_{PX}(h_{u,v}^t) \leq \Upsilon_v^t e^{-w_{u,v}}$ , and the flow destined to  $t$  on  $(u, v)$  is close to 0 if  $w_{u,v}$  is large enough, e.g.,  $e^{-10} \approx 0.005\%$ . Therefore, most flow in PEFT always makes forward progress towards the destination, i.e., from router  $u$  with larger  $d_u^t$  to router  $v$  with smaller  $d_v^t$ .

```

1: Compute necessary capacities  $\tilde{c}$  by solving (2)
2:  $\mathbf{w} \leftarrow$  Any set of link weights
3:  $\mathbf{f} \leftarrow$  Traffic_Distribution( $\mathbf{w}$ )
4: while  $\mathbf{f} \neq \tilde{c}$  do
5:    $\mathbf{w} \leftarrow$  Link_Weight_Update( $\mathbf{f}$ )
6:    $\mathbf{f} \leftarrow$  Traffic_Distribution( $\mathbf{w}$ )
7: end while
8: Return  $\mathbf{w}$  /*final link weights*/

```

**Algorithm 1:** Optimize Over Link Weights

```

1: for each link  $(u, v)$  do
2:    $w_{u,v} \leftarrow w_{u,v} - \alpha (\tilde{c}_{u,v} - f_{u,v})$ 
3: end for
4: Return new link weights  $\mathbf{w}$ 

```

**Algorithm 2:** Link-Weight\_Update( $\mathbf{f}$ )

may direct traffic “backwards” to a node that is further away from the destination. To capture these effects, recall that  $f_u^t$  is the total incoming flow at node  $u$  (including traffic originating at  $u$  as well as any traffic arriving from other nodes) that is destined to node  $t$ . In particular, the traffic  $D(s, t)$  that enters the network at node  $s$  and leaves at node  $t$  satisfies the following linear equation:

$$f_s^t - \sum_{x:(x,s) \in \mathbb{E}} f_x^t \left( \frac{\Gamma_{PX}(h_{x,s}^t)}{\sum_{(x,j) \in \mathbb{E}} \Gamma_{PX}(h_{x,j}^t)} \right) = D(s, t). \quad (21)$$

That is, the traffic  $D(s, t)$  entering the network at node  $s$  matches the total incoming flow  $f_s^t$  at node  $s$  (destined to node  $t$ ), excluding the traffic entering  $s$  from other nodes. The transit flow is captured as a sum over all incoming links from neighboring nodes  $x$ , which split their incoming traffic  $f_x^t$  over their links based on the traffic-splitting function.

The  $N$  linear equations (21) for each  $t$  typically require  $O(N^3)$  time [Cormen et al. 1990] to solve. Thus the total complexity is  $O(N^4)$ .

### 5.3 Approximate Traffic Distribution with “Downward PEFT”

To further reduce the computational overhead in link weight computation, we realize that the optimal traffic distribution should be cycle free. Thus, in the last iteration in Algorithm 1, the flow cycle should be negligible. In addition, the accurate solution for each intermediate iteration is not necessary in practice, we can approximate PEFT ( $\Gamma_{PX}(\cdot)$ ) with Downward PEFT ( $\Gamma_{PD}(\cdot)$ ) to forward traffic only on “downward” paths, the traffic distribution for each intermediate iteration can be computed using a combinatorial algorithm, which is significantly faster than solving linear equations (21).

As in Sec. 5.2, we first compute the shortest paths between all pairs of nodes, as well as the values of  $\Gamma_{PD}(h_{u,v}^t)$ , as shown in the first step of Algorithm 3. The following procedure is very similar to but subtly different from that for calculating  $\Gamma_{PD}(h_{u,v}^t)$ . We consider each destination  $t$  independently, since the flow to each

```

1: For link weights  $\mathbf{w}$ , construct all-pairs shortest paths and compute  $\Gamma_{PD}(h_{u,v}^t)$ 
2: for each destination  $t$  do
3:   Temporarily remove link  $(u, v)$  where  $d_u^t \leq d_v^t$ 
4:   Do topological sorting on the remaining network
5:   for each source  $s \neq t$  in the decreasing topological order do
6:      $f_s^t \leftarrow D(s, t) + \sum_{x:(x,s) \in \mathbb{E}} f_{x,s}^t$ 
7:      $f_{s,v}^t \leftarrow f_s^t \frac{\Gamma_{PD}(h_{s,v}^t)}{\sum_{(s,j) \in \mathbb{E}} \Gamma_{PD}(h_{s,j}^t)}$ 
8:   end for
9: end for
10:  $f_{u,v} \leftarrow \sum_{t \in \mathbb{V}} f_{u,v}^t$ 
11: Return  $\mathbf{f}$  /*set of  $f_{u,v}^*$ */

```

**Algorithm 3:** Traffic\_Distribution( $\mathbf{w}$ ) with  $\Gamma_{PD}(\cdot)$

destination can be computed without regard to the other destinations. After temporarily removing link  $(u, v)$  where  $d_u^t \leq d_v^t$  since there is no flow on it, we get an acyclic network and do topological sorting on the remaining network. The computation starts at the node without incoming link in the acyclic network, since this node would never carry any traffic to  $t$  that originates at other nodes. Proceeding through the nodes  $s$  in *decreasing* topological order, we compute the total incoming flow at node  $s$  (destined to  $t$ ) as the sum of the flow originating at  $s$  (i.e.,  $D(s, t)$ ) and the flow arriving from neighboring nodes  $x$  ( $f_{x,s}^t$ ). Then, we use the total incoming flow at  $s$  to compute the flow of traffic toward  $t$  on each of its outgoing links  $(s, v)$ , using the traffic-splitting function  $\Gamma_{PD}(\cdot)$ .

In Algorithm 3, computing the all-pairs shortest paths with the Floyd-Warshall algorithm has time complexity  $O(N^3)$  [Cormen et al. 1990]. For each destination, topology sorting requires  $O(N + E)$  time, and summarizing the incoming flow and splitting across the outgoing links requires  $O(N + E)$  time. Thus, the total time complexity to run Algorithm 3 in each iteration of Algorithm 1 is  $O(N^3 + N(N + E)) = O(N^3)$ .

Finally, the total running time for Algorithm 1 depends on the time required to solve (2) and the total number of iterations required for Algorithms 2 and 3. Interesting, although the original NEM problem involves an infinite number of variables, the complexity of Algorithm 1 is still comparable to solving a convex optimization with polynomial number of variables (like the COMMODITY problem (2)) using the gradient descent algorithm, since we do not need to solve NEM directly. However, in the terminology of complexity theory, link weight computation for PEFT is not yet proved to be polynomial-time, although in the special case of single destination, we will present the result of polynomial-solvability of PEFT in Appendix F.

## 6. PERFORMANCE EVALUATION

How well can the new routing protocol PEFT perform and how fast can the new link weight computation be? PEFT has been already proven to achieve optimal TE in Sec. 3, with a complexity of link weight computation similar to that of solving convex optimization (with a polynomial number of variables). In this section, we

numerically demonstrate that its approximate version, Downward PEFT, can make convergence very fast in practice while coming extremely close to TE optimality.

## 6.1 Simulation Environment

We consider two network objective functions ( $\Phi(\{f_{u,v}, c_{u,v}\})$ ): maximum link utilization, and total link cost (1) (as used in operator’s TE formulation). For benchmarking, the optimal values of both objectives are computed by solving linear program (2) with CPLEX 9.1 [ILOG ] via AMPL [Fourer et al. 1993].

To compare with OSPF, we use the state-of-the-art local-search method in [Fortz and Thorup 2004]. We adopt TOTEM 1.1 [TOTEM ], which follows the same approach as [Fortz and Thorup 2004], and has similar quality of the results<sup>12</sup>. We use the same parameter setting for local search as in [Fortz and Thorup 2000; 2004] where the link weights are restricted as integers from 1 to 20 since a larger weight range would slow down the searching [Fortz and Thorup 2000], initial link weights are chosen randomly, and the best result is collected after 5000 iterations.

Note that, here we do not evaluate and compare some previous works using non-even splitting over shortest paths [Wang et al. 2001; Sridharan et al. 2005] since these solutions do not enable routers to *independently* compute the flow-splitting ratios from link weights.

To determine link weights under PEFT, we run Algorithm 1 with up to 5000 iterations of computing the traffic distribution and updating link weights. Abusing terminology a little, in this section we use the term PEFT to denote the traffic engineering with Algorithm 1 (including two sub-Algorithms 2 and 3).

We run the simulation for a real backbone network and several synthetic networks. The properties of the networks used are summarized in Table IV, which will be presented in Subsection 6.5. First is the Abilene network (Fig. 2) [Abi ], which has 11 nodes and 28 directional links with 10Gbps capacity. The traffic demands are extracted from the sampled Netflow data on Nov. 15th, 2005. To simulate networks with different congestion levels, we create different test cases by uniformly decreasing the link capacity until the maximal link utilization reaches 100% with optimal TE.

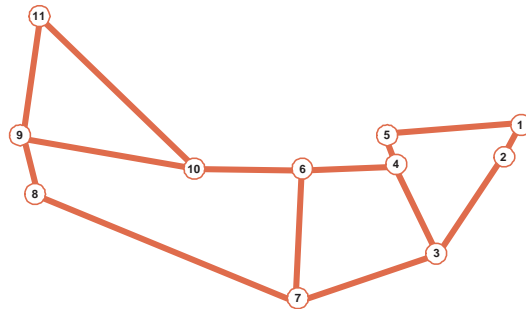


Fig. 2. Abilene Network

<sup>12</sup>Proprietary enhancements can bring in factors of improvement, but as we will see, PEFT’s advantage on computational speed is orders-of-magnitude.

We also test the algorithms on the same topologies and traffic matrices as those in [Fortz and Thorup 2004]. The 2-level hierarchical networks were generated using GT-ITM, which consists of two kinds of links: local access links with 200-unit capacity and long distance link with 1000-unit capacity. In the random topologies, the probability of having a link between two nodes is a constant parameter and all link capacities are 1000 units. In these test cases, for each network, traffic demands are uniformly increased to simulate different congestion levels.

## 6.2 Minimization of Maximum Link Utilization

Since we create different levels of congestion for the same network by uniformly decreasing link capacities or uniformly increasing traffic demands, we just need to compute the Maximum Link Utilization (MLU) for one test case in each network because MLU is proportional to the ratio of total demand over total capacity. In addition to MLU, we are particularly interested in the metric “efficiency of capacity utilization”,  $\eta$ , which is defined as the following ratio: the percentage of the traffic demand satisfied when the MLU reaches 100% under a traffic engineering scheme over that in optimal traffic engineering. The improvement in  $\eta$  is referred to as the “Internet capacity increase” in [Fortz and Thorup 2004].

For any test case of a network, if MLU of optimal TE, OSPF, and PEFT are  $\xi$ ,  $\xi_O$  and  $\xi_D$  respectively, then  $\eta_O = \frac{\xi}{\xi_O}$ , and  $\eta_D = \frac{\xi}{\xi_D}$ . Thus PEFT can increase Internet capacity over OSPF by  $\eta_D - \eta_O$ . Table III shows the maximum link utilizations of optimal traffic engineering, PEFT, and Local Search OSPF for the test case with the lightest loading of each network. Fig. 3 illustrates the efficiency of capacity utilization of the three schemes. They show that PEFT is very close to optimal traffic engineering in minimizing MLU, and increases Internet capacity over OSPF by **15%** for Abilene network and **24%** for hier50b network, respectively.

Table III. Maximum link utilization of optimal traffic engineering, PEFT and Local Search OSPF for light-loading networks

Net. ID	Optimal TE	PEFT	OSPF
abilene	33.9%	33.9%	39.8%
hier50a	56.4%	56.5%	58.6%
hier50b	44.7%	45.0%	59.2%
rand50	60.6%	60.6%	60.6%
rand50a	60.8%	60.8%	64.7%
rand100	55.0%	55.0%	71.5%

## 6.3 Minimization of Total Link Cost

We also employ the cost function (1) as in [Fortz and Thorup 2004]. Comparison is on the optimality gap, in terms of the total link cost, compared against the value achieved by optimal traffic engineering. Typical results for different topologies with various traffic matrices are shown in Fig. 4, where the network loading is the ratio of total demand over total capacity. From the results, we observe that the gap between OSPF and optimal traffic engineering can be very significant (up to 821%) for the most congested case of Abilene network. In contrast, PEFT can achieve almost the same performance as the optimal traffic engineering in terms of total

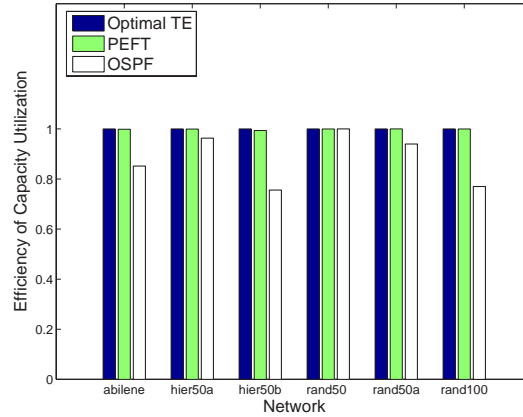


Fig. 3. Efficiency of capacity utilization of optimal traffic engineering, PEFT and local Search OSPF

link cost. Note that, within those figures, the maximum optimality gap of PEFT is only up to 8.8% in Fig. 4(b), which can be further reduced to 1.5% with a larger step-size and more iterations (which is feasible as the algorithm runs very fast to be shown in Sec. 6.5).

#### 6.4 Convergence Behavior

Fig. 5 shows the optimality gap in terms of total cost achieved by PEFT, using different step-sizes, within the first 5000 iterations for Abilene network with the least link capacities. It provides convergence behavior typically observed. The legends show the ratio of the step-size over the default setting. It demonstrates that the algorithms developed in Sec. 5 for the PEFT protocol converges very fast even with default setting, and reduces the gap to 5% after 100 iterations and 1% after 3000 iterations. In addition, increasing step-size a little will speed up the convergency, and as expected, too large a step-size (e.g., 2.5 in the above example) would cause oscillation. Notice that there is a wide range of step-sizes that can make convergence very fast. Further fine-tuning of step-size is a future work.

#### 6.5 Running Time Requirement

Besides the convergence behavior, the actual running time is also an important evaluation criteria. The tests for PEFT and local search OSPF were performed under the time-sharing servers of Redhat Enterprise Linux 4 with Intel Pentium IV processors at 2.8~3.2 Ghz. Note that the running time for local search OSPF is sensitive to the traffic matrix since a near-optimal solution can be reached very fast for light traffic matrices. Therefore, we show the range of their average running times per iteration for qualitative reference.

Fig. 6 shows the optimality gap (on a log scale) achieved by local search OSPF and PEFT, within the first 500 iterations for a typical scenario (Fig. 4(c)). It demonstrates that Algorithm 1 for PEFT converges much faster than local search for OSPF. Table IV shows the average running time per iteration for different networks. We observe that our algorithm is very fast, requiring at most 2 minutes even for the largest network (with 100 nodes) tested, while the OSPF local search

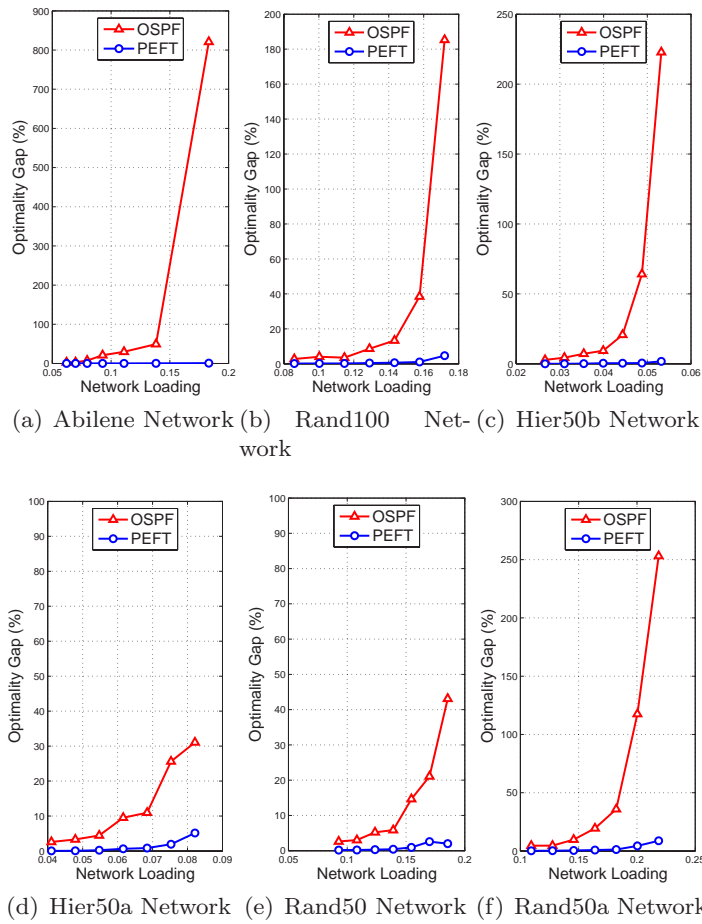


Fig. 4. Comparison of PEFT and Local Search OSPF in terms of optimality gap on minimizing total link cost

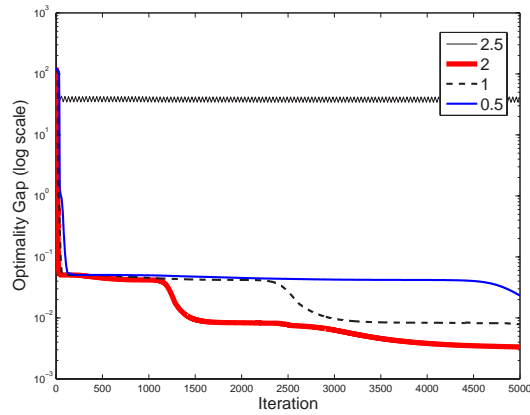


Fig. 5. Evolution of optimality gap of PEFT with different step sizes

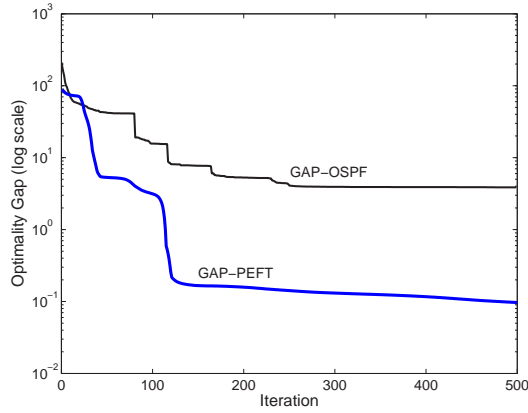


Fig. 6. Comparison of the drop in optimality gap between Local Search OSPF and PEFT in a 2-level topology with 50 nodes and 212 links

needs tens of hours on the same computer. On average, the algorithm developed in this paper to find link weights for PEFT routing is 2000 times faster than local search algorithms for OSPF routing.

Table IV. Average running time per iteration required by PEFT and local search OSPF to attain the performance in Fig. 4

Net. ID	Topology	Node #	Link #	Time per Iteration (second)	
				PEFT	OSPF
abilene	Backbone	11	28	0.002	6.0~13.9
hier50a	2-level	50	148	0.006	6.0~13.9
hier50b	2-level	50	212	0.007	6.4~17.4
rand50	Random	50	228	0.007	3.2~9.0
rand50a	Random	50	245	0.007	6.1~14.1
rand100	Random	100	403	0.042	39.5~105.1

## 7. NEM: A FRAMEWORK FOR LINK-STATE ROUTING

In this section, we highlight the conceptual framework of NEM and the differences between NEM and Network Utility Maximization (NUM).

As explained in Sec. 3, NEM is developed in this paper as a unifying mathematical model that enables the discovery and development of new link-state routing protocol, PEFT. Although NEM is solved by neither routers nor operators, its solution leads to both the development of PEFT traffic splitting and link weight computation algorithms. More discussions on the intuitions behind NEM can be found in Appendix B.

On the other hand, TCP congestion control protocols have been studied extensively since 1998 as solutions to another family of optimization models called NUM. The notion of network utility was first advocated in [Shenker 1995] in 1995 for bandwidth allocation among elastic demands on *source rates*. The NUM problem (22) was first introduced for TCP congestion control (e.g., [Kelly et al. 1998; Yäiche et al. 2000; Low 2003; Srikant 2004]). Consider a communication network with  $L$  logical links, each with a fixed capacity of  $c_l$  bps, and  $S$  sources (i.e., end users), each transmitting at a source rate of  $x_s$  bps. Each source  $s$  emits one flow,

using a fixed set  $L(s)$  of links in its path, and has an increasing (and often concave) function  $U_s(x_s)$  called utility function. Each link  $l$  is shared by a set  $S(l)$  of sources. NUM, in its basic version, is the following problem of maximizing the network utility  $\sum_s U_s(x_s)$ , over the source rates  $\mathbf{x}$ , subject to linear flow constraints  $\sum_{s \in S(l)} x_s \leq c_l$  for all links  $l$  (Note that routing is fixed in NUM formulation):

$$\begin{aligned} & \text{maximize} && \sum_s U_s(x_s) \\ & \text{subject to} && \sum_{s \in S(l)} x_s \leq c_l, \forall l, \\ & \text{variables} && \mathbf{x} \succeq 0. \end{aligned} \tag{22}$$

There is a useful economics interpretation of the dual-based distributed algorithm for NUM, in which the Lagrange dual variables can be interpreted as shadow prices for resource allocation, and end users and the network maximize their net utilities and net revenue, respectively. Many reverse-engineering of existing TCP variants and forward-engineering of new congestion control protocols have been developed with the NUM model as a starting point.

The NEM problem proposed in this paper is *not* a special case of NUM, since entropy is not an increasing function, and the design freedom in NEM is routing rather than rate control. Instead, there is a useful and interesting *parallel* between the framework of NEM proposed this paper, for link-state routing protocols in IP layer, and that of NUM matured over the last decade, for end-to-end congestion control protocols in TCP layer. The comparison between the two frameworks is shown in Table V, where results from this paper are highlighted in italics.

Table V. NUM for TCP and NEM for IP: Main Differences

Property	Congestion Control (TCP)	Traffic Engineering (IP)
Traffic type	Elastic	Inelastic
Flow distribution	Fixed	Variable
Participants	End user and router	Operator and router
Timescale	Seconds	Hours
Optimization Model	Network Utility Maximization	<i>Network Entropy Maximization</i>
Lagrange multipliers	Congestion price	<i>Link weight</i>
Reverse engineering	Tahoe, Reno, Vegas, etc.	<i>Even splitting in OSPF</i>
Forward engineering	FAST TCP, etc.	<i>PEFT</i>

## 8. DIFFERENCES BETWEEN PEFT AND DEFT

Here we explain several points of potential confusion between PEFT in this paper and DEFT in [Xu et al. 2007]. Link-state routing protocols can be categorized as link-based and path-based in terms of flow splitting. Their difference is illustrated in Fig. 7, with a network that only has traffic demand from  $s$  to  $t$ . Assume the weights of the links are shown in Fig. 7(a). Obviously, the shortest distance from  $s$  to  $t$  is 2 units and both nodes  $t$  and  $u$  are on the shortest paths from  $s$  to  $t$ . In a link-based splitting scheme (e.g. OSPF, Fong [Fong et al. 2005] and DEFT [Xu et al. 2007]), node  $s$  evenly splits traffic across its *two* outgoing links ( $s, t$ ) and ( $s, u$ ) as shown in Fig. 7(b). Whereas in a path-based splitting scheme, e.g. PEFT,

there are *three* equal-length paths from  $(s, t)$  and  $s$  evenly splits traffic across them as shown in Fig. 7(c). Note that, the path-based model does not imply explicit routing to set up tunnels for all the possible paths. Instead, each node just needs to compute and stores the aggregated flow-splitting ratio across its outgoing links, like 66% on link  $(s, u)$  for the sample network in Fig 7(c). Therefore, path-based splitting schemes can still be realized with hop-by-hop forwarding.

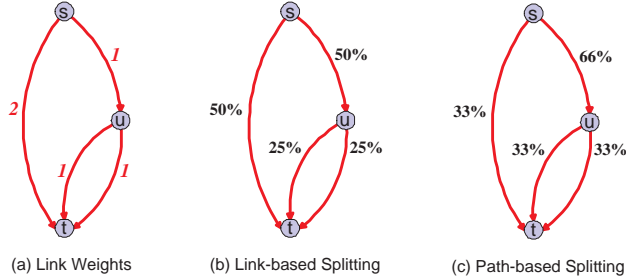


Fig. 7. Difference in traffic splittings for link-based and path-based link-state routing protocol

This paper is substantially different from our previous work on [Xu et al. 2007], with the following key differences:

- (1) DEFT is a link-based flow splitting while PEFT is a path-based flow splitting.
- (2) The core algorithms for setting link weights are completely different. [Xu et al. 2007] introduces a non-convex non-smooth optimization for DEFT and a two-stage iterative solution method, while the theory for PEFT is Network Entropy Maximization. The two-stage method for DEFT is *much slower* than the algorithms developed for PEFT in this paper.
- (3) [Xu et al. 2007] *numerically* shows DEFT can realize *near optimal* TE in terms of a particular objective (total link cost), while this paper *proves* that PEFT can realize *optimal* TE with any convex objective function.

## 9. CONCLUDING REMARKS

Commodity-flow-based routing protocols are optimal for any convex objective in Internet TE but introduce much configuration complexity. Link-state routing is simple but prior work suggests it does not achieve optimal TE. This paper proves that optimal traffic engineering, in fact, *can* be achieved by link-state routing with hop-by-hop forwarding, and the right link weights *can* be computed efficiently, as long as flow splitting on non-shortest paths is allowed but properly penalized. In Appendices, we also show uniqueness of the exponential penalty in achieving optimal TE, and discuss interpretations of NEM from the viewpoints of statistical physics and combinatorics.

Before concluding this paper, we would like to highlight that optimization is used in three different ways in this paper. First and obviously, it is used when developing algorithms to solve the link weight computation problem for PEFT.

In a more interesting way, the level of difficulty of optimizing link weights for OSPF is used as a hint that perhaps we need to revisit the standard assumption on how link weights should be used. In this approach of “Design For Optimizability”,

sometimes a restrictive assumption in the protocol can be perturbed at low “cost” and yet turn a very hard network-management problem into an efficiently solvable one. In this case, better (and indeed the best) TE and faster weight computation are simultaneously achieved.

In yet another way, optimization in the form of NEM is introduced as a conceptual framework to develop routing protocols. The NEM framework for distributed routing also leads to several interesting future directions, including extensions to robust TE and to the interactions between congestion control at sources with link-state routing in the network.

### Acknowledgment

This research is in part supported by DARPA W911NF-07-1-0057, ONR YIP N00014-07-1-0864, AFOSR FA9550-06-1-0297, NSF CNS-0519880 and CNS 0720570. We appreciate the helpful discussions with D. Applegate, B. Fortz, J. He, J. Huang, D. Johnson, H. Karloff, Y. Li, J. Liu, M. Prytz, A. Tang, M. Thorup, J. Yu, and J. Zhang.

### Appendices

In the Appendices, we present more details about NEM and PEFT. Appendix A answers several potential questions from the readers. Appendix B introduces the physical interpretation of entropy for IP routing. Appendix C shows the uniqueness of choosing the entropy function to pick out the right flow distributions realizable with link-state routing. Appendix D proves Lemma 1 on the convergence of solving the NEM problem with the gradient descent algorithm. Appendix E introduces how to realize the multi-commodity-flow solution with up to  $O(N^2E)$  tunnels, which also can be used as an initial feasible solution for the NEM problem (4). Appendix F shows a polynomial-time algorithm of setting optimal link weights for PEFT in a single-destination network.

#### A FAQ: Key Hints for Readers

1. **Q:** Why optimal TE with link-state routing was an open problem given Wang et al.’s work [Wang et al. 2001]?

**A:** Wang et al. [Wang et al. 2001] shows one set of link weights can be found such that optimal flow is forwarded along shortest paths using non-even splitting. However, the network operator needs to specify  $O(NE)$  parameters to control flow splitting in addition to  $E$  link weights.

2. **Q:** The proposed NEM problem has an infinite number of variables. Can it still be solved scalably?

**A:** Yes. We do not need to write down the NEM problem explicitly or obtain the optimal value for each variable. Instead, we just search for  $E$  dual variables (link weights) which can enable optimal solution of NEM problem. Each step in the proposed gradient descent algorithm has polynomial time complexity in terms of the number of nodes and edges.

3. **Q:** What are the impacts on control plane and data plane on a router from implementing PEFT?

**A:** In the control plane, PEFT does not change the routing-protocol messages that

are sent between the routers, (an important consideration for practical use), but does change the computation done locally on each router based on the weights.

In the data plane, routers today implement hash-based splitting over multiple outgoing links, typically with an even (1 out of  $n$ ) splitting ratio. PEFT requires flexible splitting over multiple outgoing links, thus we need to store the splitting percentages in the data plane – whereas for  $1/n$  spitting, the splitting ratio is implicitly even. It requires a little bit extra storage and processing, not enough to become a new bottleneck, when packets arrive to direct packets to the appropriate outgoing links.

4. **Q:** Is this PEFT work significantly different from DEFT [Xu et al. 2007]?

**A:** Yes. Though both exponential splitting, PEFT is a path-based splitting while DEFT is link-based splitting. The theory, optimization techniques and key results for PEFT are all different from that for DEFT as explained in Section 8.

## B Entropy Maximization and Most Likely Flow Configuration

There are several intriguing relationships between the framework of Network Entropy Maximization for link-state routing and statistical physics. We speculate about some of the thought-provoking connections in this appendix.

In classical statistical mechanics, many microscopic behaviors aggregate into macroscopic states, and an isolated thermodynamic system will eventually reach an equilibrium macroscopic state that is the most likely one. Interestingly, entropy maximization for traffic engineering can be motivated by an argument of most likely flow configuration, as shown below.

Consider a network with only one source-destination pair  $(s, t)$  and  $P$  un-capacitated paths between them. If there are  $T$  packets to be transmitted from  $s$  to  $t$ , let  $T_i \geq 0$  be the number of packets on path  $i$ , with  $\sum_i T_i = T$ . Each set of such  $\{T_i\}$ , which can be represented as a vector, is referred to as a *macroscopic state*. In contrast, each collection of routing decisions for individual packets represents a *microscopic state*. There are a total of  $P^T$  possible microscopic states. The number of microscopic states consistent with a given macroscopic state can be viewed as a measure of likelihood of that macroscopic state.

The number of microscopic states corresponding to the macroscopic state  $\{T_i\}$  is  $K = \frac{T!}{\prod_i T_i!}$ . We want to search for the macroscopic state with the largest number of  $K$ , i.e.,  $\max K$ , or, equivalently,  $\max \log K = \max \log \frac{T!}{\prod_i T_i!}$ . For large system asymptote,  $T$  and  $T_i$  are large numbers, hence using Stirling's approximation:  $n! \approx n^n e^{-n}$ , we have  $\log K \approx \log(e^{-T} T^T) - \sum_i \log(e^{-T_i} T_i^{T_i}) = -T \sum_i \frac{T_i}{T} \log \frac{T_i}{T}$ .

This shows that the system equilibrium is the flow configuration that maximizes the entropy,  $-\sum_i T x_i \log x_i$ , where  $x_i = \frac{T_i}{T}$  is the fraction of flow on path  $i$ .

The optimality result of PEFT through NEM suggests an intriguing connection between the *principle of entropy maximization* and that of *shortest description length*, since maximizing entropy picks out those traffic distribution that can be realized by the simplest set of routing configuration parameters: one weight per link to be used independently by each router.

### C Uniqueness of Exponential Penalty

Can optimal traffic engineering be achieved by other penalty function on longer paths? In this subsection, we demonstrate that exponential penalty is the only way of realizing optimal traffic distribution with path-based link-state routing.

As in (12), we use  $\lambda_{u,v}$  as weight for link  $(u,v)$ , denote  $p \triangleq K_{P_{s,t}^i}^{(u,v)} \lambda_{u,v}$  as the length of the  $i$ -th path, define  $\frac{\mu_{s,t}^*}{D(s,t)}$  as  $q$ , and simplify  $x_{s,t}^{i*}$  as  $x$ , then we have

$$z'(x) - p - q = 0, \quad (23)$$

then

$$z(x) = (p + q)x + C_1, \quad (24)$$

where  $C_1$  is a constant, and

$$p + q = \frac{z(x) - C_1}{x} \triangleq \psi(x). \quad (25)$$

Assume  $\psi(x)$  is reversible, then we have

$$x = \psi^{-1}(p + q) \quad (26)$$

We also denote  $x = \varphi(p, q)$ . Note that, for path-based link-state routing, for two paths of the same demand  $D(s, t)$ , the ratio of the traffic over them should depend only on their path lengths. For a path of length  $p$  and a shortest path of length  $p_0$ , we have

$$\begin{aligned} & \frac{\varphi(p, q)}{\varphi(p_0, q)} = f_1(p, p_0) \\ \Rightarrow & \log \varphi(p, q) - \log \varphi(p_0, q) = \log f_1(p, p_0) \\ \Rightarrow & \frac{d \log \varphi(p, q)}{dq} - \frac{d \log \varphi(p_0, q)}{dq} = 0 \\ \Rightarrow & \int_{q_0}^q \frac{d \log \varphi(p, q)}{dq} dq = \int_{q_0}^q \frac{d \log \varphi(p_0, q)}{dq} dq \\ \Rightarrow & \log \varphi(p, q) \Big|_{q_0}^q = \log \varphi(p_0, q) \Big|_{q_0}^q \\ \Rightarrow & \varphi(p, q) = \frac{\varphi(p, q_0) \varphi(p_0, q)}{\varphi(p_0, q_0)} \end{aligned} \quad (27)$$

where  $p_0, q_0$  are constants.

Therefore, we can define two functions  $f(p) \geq 0$  and  $g(q) \geq 0$ , such that

$$x = f(p)g(q), \quad (28)$$

where

$$\begin{aligned} \frac{dx}{dp} &= f'(p)g(q) \\ \frac{dx}{dq} &= f(p)g'(q) \end{aligned} \quad (29)$$

From (26),  $\frac{dx}{dp} = \frac{dx}{dq}$ , thus

$$\begin{aligned} f'(p)g(q) &= f(p)g'(q) \\ \Rightarrow \frac{f'(p)}{f(p)} &= \frac{g'(q)}{g(q)}. \end{aligned} \quad (30)$$

Since  $\frac{f'(p)}{f(p)}$  is a function of  $p$  and  $\frac{g'(q)}{g(q)}$  is a function of  $q$ , thus

$$\frac{f'(p)}{f(p)} = \frac{g'(q)}{g(q)} = C. \quad (31)$$

where  $C < 0$  since  $f'(p) \leq 0$  assuming we send more traffic on a shorter path.

Therefore,  $f(p) = Ae^{Cp}$  and  $g(q) = Be^{Cq}$ ,  $x = AB e^{C(p+q)}$ . Then  $z(x) = \frac{x \log \frac{x}{AB}}{C} + C_1 = \frac{x \log x}{C} - \frac{\log(AB)}{C}x + C_1$ . Consider the objective function (4a) and constraint (4c) of NEM problem, and ignore the exact values of the constant parameters  $A, B, C$  and  $C_1$ . It is now clear that we can choose  $z(x) = -x \log x$  as the objective function, and there is no other format of  $z(x)$  resulting in a flow which can be realized by link-state routing.

#### D Proof of Lemma 1

PROOF. Since strong duality holds for problem (4) and its Lagrange dual problem (7), we solve the dual problem through gradient method and recover the primal optimizers from the dual optimizers. By Danskin's Theorem [Bertsekas 1999],

$$\frac{\partial Q(\boldsymbol{\lambda}(q))}{\partial \lambda_{u,v}(q)} = \tilde{c}_{u,v} - \sum_{s,t,i} D(s,t) K_{P_{s,t}^i}^{(u,v)} x_{s,t}^i(q), \forall (u,v) \in \mathbb{E}.$$

Hence, the algorithm in (9) is a gradient descent algorithm for dual problem (7). Since the dual objective function  $Q(\boldsymbol{\lambda})$  is a convex function, there exists a step size  $\alpha(q)$  that guarantees  $\boldsymbol{\lambda}(q)$  to converge to the optimal dual solutions  $\boldsymbol{\lambda}^*$  [Bertsekas 1999]. Also, if  $\nabla Q(\boldsymbol{\lambda})$  satisfies a Lipschitz continuity condition, i.e., there exists a constant  $H > 0$  such that

$$\|\nabla Q(\boldsymbol{\lambda}_1) - \nabla Q(\boldsymbol{\lambda}_2)\| \leq H \|\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2\|, \forall \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2 \succeq \mathbf{0},$$

then  $\boldsymbol{\lambda}(q)$  converges to the optimal dual solution  $\boldsymbol{\lambda}^*$  with a sufficiently small constant step size  $\alpha(q) = \alpha, 0 < \alpha < 2/H$  [Bertsekas 1999]. The Lipschitz continuity condition is satisfied if the curvatures of the entropy functions are bounded away from zero, see [Low and Lapsley 1999] for further details. Furthermore, since problem (4) is a strictly convex optimization problem and TRAFFIC-DISTRIBUTION problems (8) have unique solutions,  $\boldsymbol{x}^*$  are the globally optimal primal solutions of (4) [Minoux 1986].  $\square$

#### E Tunnel-based Routing to Realize Optimal TE

A tunnel-based routing can be derived from the optimal solution of the COMMODITY problem (2) based on dual-composition. The approach follows the same way as the flow decomposition technique in [Mittra and Ramakrishnan 1999]. We rephrase the approach and illustrate its complexity. The flow destined to the same destination is treated as a commodity. In the optimal solution of (2), there are up to  $N$  acyclic commodity flows where  $N$  is the node number. The paths with flow can be determined for each commodity independently. For commodity  $t$ , starting with any source  $s$ , temporarily remove all the links without flow to  $t$  (i.e.,  $f_{u,v}^t = 0$ ). In the remaining network, choose any path from  $s$  to  $t$  and let  $(u', v')$  be the link with the least  $f_{u',v'}^t$  along the path, then deduct  $f_{u',v'}^t$  from demand  $D(s, t)$  and flow  $f_{u',v'}^t$  for all the links along the path. Remove link  $(u', v')$  from further consideration.

Repeat the above procedure until the paths for  $D(s, t)$  have been determined. For each demand  $D(s, t)$ , there are at most  $E$  paths with flow since at least one link is removed during each step. Therefore, the total number of paths for  $N$  commodities (and  $O(N^2)$  source/destination pair) is  $O(N^2E)$ . Hence, the above procedure finishes within polynomial time.

#### F Polynomial-time Algorithm of Link Weight Setting for Single-destination Network

For a single-destination (sink) network, the link weights to realize optimal TE with PEFT can be found in polynomial-time. The method is much faster than solving the NEM problem with the gradient descent algorithm. We have the following lemma first.

**Lemma 2.** “Downward PEFT” can realize any acyclic flow for single destination in polynomial time.

**PROOF.** The links without flow can be assigned infinitely large weights and excluded from further processing. Denote  $f_u^t = \sum_{(u,v) \in \mathbb{E}} f_{u,v}^t$ , where  $f_{u,v}^t$  is the amount of flow on link  $(u, v)$ . The nodes are processed in their reverse topological order in the acyclic flow, where the first node is the destination  $t$ , with  $\Upsilon_t^t = 1$  (Sec. 4.3). When node  $u$  is processed, from (17), (18b) and (19), we have

$$f_{u,v}^t = f_u^t \frac{e^{-h_{u,v}^t} \Upsilon_v^t}{\Upsilon_u^t}, \quad (32)$$

and

$$h_{u,v}^t = -\log \frac{f_{u,v}^t \Upsilon_u^t}{f_u^t \Upsilon_v^t} \geq 0, \quad (33)$$

then

$$\Upsilon_u^t \leq \frac{f_u^t \Upsilon_v^t}{f_{u,v}^t}. \quad (34)$$

We can set  $\Upsilon_u^t = \min_{(u,v) \in \mathbb{E}} \frac{f_u^t \Upsilon_v^t}{f_{u,v}^t}$  since at least one link  $(u, v_0)$  is on the shortest path from  $u$  to  $t$ , i.e.  $h_{u,v_0}^t = 0$ . Then we set the weight for link  $(u, v_0)$  as  $w_{min}$ , and the shortest distance from node  $u$  to  $t$ ,  $d_u^t = w_{min} + d_{v_0}^t$ . Then the weight of link  $(u, v)$  is  $-\log \frac{f_{u,v}^t \Upsilon_u^t}{f_u^t \Upsilon_v^t} + d_u^t - d_v^t$  from (33). It is easy to verify that the above link weighting satisfies the definition of downward PEFT (20)<sup>13</sup> and the time complexity is  $O(N + E)$ .  $\square$

**Proposition 2.** Downward PEFT can achieve optimal traffic engineering with single destination in polynomial time.

**PROOF.** An obvious conclusion from Lemma 2 since optimal TE is cycle free.  $\square$

#### REFERENCES

Abilene Backbone Network, <http://abilene.internet2.edu/>.

<sup>13</sup>All  $d_v^t$  have been determined since the nodes are processed in the reverse topological order and  $d_t^t \equiv 0$

- AGRAWAL, A. K., MOHAN, D., AND SINGH, R. S. 2005. Traffic planning in a constrained network using entropy maximisation approach. *Journal of the Institution of Engineers, India. Civil Engineering Division* 85, 236–240.
- AWDUCHE, D. 1999. MPLS and traffic engineering in IP networks. *IEEE Communication Magazine* 37, 12 (Dec.), 42–47.
- BERTSEKAS, D. P. 1999. *Nonlinear Programming*, Second ed. Athena Scientific.
- BLUNDEN, W. R. 1967. *Introduction to traffic science*. Printerhall, London.
- BOYD, S. AND VANDENBERGHE, L. 2004. *Convex Optimization*. Cambridge University Press.
- CORMEN, T., LEISERSON, C., AND RIVEST, R. 1990. *Introduction to Algorithms*. The MIT Press, Cambridge.
- FONG, J. H., GILBERT, A. C., KANNAN, S., AND STRAUSS, M. J. 2005. Better alternatives to OSPF routing. *Algorithmica* 43, 1-2, 113–131.
- FORTZ, B. AND THORUP, M. 2000. Internet traffic engineering by optimizing OSPF weights. In *INFOCOM'00, Tel Aviv, Israel*. 519–528.
- FORTZ, B. AND THORUP, M. 2004. Increasing Internet capacity using local search. *Computational Optimization and Applications* 29, 1, 13–48.
- FOURER, R., GAY, D. M., AND KERNIGHAN, B. W. 1993. *AMPL: A Modeling Language for Mathematical Programming*. Boyd & Fraser Publishing Co., Danvers, MA, USA.
- ILOG. CPLEX, <http://www.ilog.com/products/cplex/>.
- KELLY, F., MAULLOO, A., AND TAN, D. Mar. 1998. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society* 49, 3, 237–252.
- LOW, S. H. 2003. A duality model of TCP and queue management algorithms. *IEEE/ACM Transactions on Networking* 11, 4, 525–536.
- LOW, S. H. AND LAPSLEY, D. E. 1999. Optimization flow control - I: basic algorithm and convergence. *IEEE/ACM Transactions on Networking* 7, 6, 861–874.
- MINOUX, M. 1986. *Mathematical programming: theory and algorithms*. Wiley.
- MITRA, D. AND RAMAKRISHNAN, K. G. Dec. 1999. A case study of multiservice multipriority traffic engineering design for data networks. In *GLOBECOM'99, Rio de Janeiro, Brazil*. 1077–1083.
- SHENKER, S. Sep. 1995. Fundamental design issues for the future Internet. *IEEE Journal on Selected Areas in Communications (JSAC)* 13, 7 (September), 1176–1188.
- SLEATOR, D. D. AND TARJAN, R. E. 1983. A data structure for dynamic trees. *Journal of Computer and System Sciences* 26, 3, 362–391.
- SRIDHARAN, A., GUÉRIN, R., AND DIOT, C. 2005. Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks. *IEEE/ACM Transactions on Networking* 13, 2, 234–247.
- SRIKANT, R. 2004. *The Mathematics of Internet Congestion Control (Systems and Control: Foundations and Applications)*. Springer Verlag.
- SRIVASTAVA, S., AGRAWAL, G., PIORO, M., AND MEDHI, D. 2005. Determining link weight system under various objectives for OSPF networks using a Lagrangian relaxation-based approach. *IEEE e-Trans on Network & Service Management* 2, 1, 9–18.
- TOMLIN, J. A. 2003. A new paradigm for ranking pages on the world wide web. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*. ACM Press, New York, NY, USA, 350–355.
- TOMLIN, J. A. AND TOMLIN, S. G. 1968. Traffic distribution and entropy. *Nature* 220, 974–976.
- TOTEM. <http://totem.info.ucl.ac.be>.
- WANG, Z., WANG, Y., AND ZHANG, L. 2001. Internet traffic engineering without full mesh overlaying. In *INFOCOM'01, Anchorage, AK*.
- XU, D., CHIANG, M., AND REXFORD, J. May 2007. DEFT: Distributed exponentially-weighted flow splitting. In *INFOCOM'07, Anchorage, AK*.
- YÄICHE, H., MAZUMDAR, R. R., AND ROSENBERG, C. 2000. A game theoretic framework for bandwidth allocation and pricing in broadband networks. *IEEE/ACM Transactions on Networking* 8, 5, 667–678.