# Optimal Link-state Hop-by-hop Routing

Nithin Michael*, Ao Tang* and Dahai Xu†
*ECE Dept., Cornell University, Ithaca, NY 14853, USA. Email: {nm373@,atang@ece.}cornell.edu
†AT&T Labs Research, Florham Park, NJ 07932, USA Email: dahaixu@research.att.com

***Abstract*—Current intra-domain routing protocols like OSPF and IS-IS use link-state routing algorithms with hop-by-hop forwarding that sacrifice traffic engineering performance for ease of implementation and management. Though optimal traffic engineering algorithms exist, they tend to be either not link-state algorithms or to require source routing – characteristics that make them difficult to implement. As the focus of this paper, we introduce HALO, the first optimal link-state routing algorithm with hop-by-hop forwarding, where link weights can be calculated locally. Furthermore, our solution can adapt to changing traffic patterns automatically. The optimality of the algorithm is proved theoretically and also verified numerically.**

## I. Introduction

Finding optimal routes [1] in packet-switched networks has been of fundamental research and practical interest since the early 1970s with the advent of ARPANET [2], the predecessor of the Internet. But today, sub-optimal distributed link-state routing protocols with hop-by-hop forwarding like OSPF and IS-IS are the dominant intra-domain routing solutions on the Internet. These algorithms have become ubiquitous despite their potentially very large performance loss because of the explosive growth of the Internet. As the network scaled, it was clear that the simplicity of these schemes (the main idea is to centrally assign weights to links and locally calculate shortest paths) made them easier to implement and manage compared to the optimal solutions that had been proposed. Some of the lost performance was recouped through extensive capital expenditure. For instance, due to the poor resource utilization resulting from these protocols, many of the "backbone" links of the internet are so over-provisioned to support peak traffic that they run at very low utilizations on average. Unsurprisingly, the search for an optimal routing algorithm that has the same ease of management and implementation as OSPF has continued unabated. In this paper we present just such an algorithm, HALO (Hop-by-hop Adaptive Link-state Optimal). To the best of our knowledge, this is the first optimal link-state hop-by-hop routing algorithm.

Before we proceed, we define a few basic terms followed by some more background to motivate our study.

- Link-state: Routers make routing decisions based on knowledge of the network topology and the weights associated with the links.
- Hop-by-hop: Each router, based on the destination address, controls only the next hop that a packet takes.
- Optimal: The routing algorithm minimizes some cost function (e.g. minimize total delay) determined by the network operator. The problem of guiding network traffic through routing to minimize a given global cost function is called traffic engineering (TE).

Additionally, our algorithm has the important benefit of adaptivity by which we mean that given the link flow rates, the algorithm does not require the traffic demand matrix as an input in order to compute link weights or optimal routes. Specifically, the algorithm seamlessly recognizes and adapts to changes in the network, both topology changes and traffic variations, as inferred from the link flow rates – a useful property for networks with ever changing traffic demand, such as the Internet.

As noted earlier, our work was motivated by the performance loss of OSPF and IS-IS and the resulting inefficiencies. In fact, given the offered traffic, finding the optimal link weights, if they even exist, for OSPF/IS-IS is a well-known NP-hard problem [3]. Furthermore, it is possible for even the best weight setting to lead to traffic that deviates significantly from the optimal traffic distribution [3]. But as can be expected, designing an optimal protocol while keeping the simplicity of link-state hop-by-hop protocols comes with a few challenges.

Firstly, relying only on link-state information means that no router is aware of the individual communicating pairs in the network or their requirements and yet have to act independently such that the TE objective is optimized. This is a very real restriction as in any large dynamic network like the Internet, it is not possible to obtain information about individual communicating pairs. If the link-state requirement is set aside, optimal distance-vector protocols that rely on locally transmitted node-states exist [4]. However, the main reason that a distance-vector protocol is not preferred for intra-domain routing is because it suffers from scalability issues as well as decreased robustness like vulnerability to a single rogue router taking down the network as in the "Internet Routing Black Hole" incident of 1997 [5].

Secondly, the hop-by-hop forwarding requirement prevents routers from controlling anything except the next hop that a packet can take. As a result, a router cannot determine the entire path that traffic originating at it takes to its destination. If this requirement is set aside, a projected gradient approach [6] can be used to yield optimal link-state algorithms which can be implemented with source routing, where the path a packet takes through the network is encoded in its entirety at the source. Unfortunately, source routing is not feasible for even moderate size networks. Even though such schemes can be implemented with MPLS [7], optimality comes at the cost of establishing multiple end-to-end virtual circuits. Moreover, as the traffic changes, the end-to-end virtual circuits that were

978-1-4799-1270-4/13/$31.00 

established for a particular traffic pattern become less useful and performance degrades.

Lastly, our solution has the advantage of being adpative and if this requirement is set aside, recently significant progress was made in this direction with PEFT, a link-state protocol with hop-by-hop forwarding based on centralized weight calculations [8]. Since the link weights are not updated locally, the routers cannot automatically react to changes in the network state by modifying the link weights, making PEFT not adaptive. Also, PEFT does not guarantee optimality as claimed in the paper[1].

In this paper we introduce new ideas and reach the first adaptive link-state optimal routing algorithm with hop-by-hop forwarding. The rest of the paper is organized as follows. In Section II we review the different algorithms that have been proposed for the traffic engineering problem. Next we present the problem formulation in Section III before presenting and analyzing our solution in Sections IV and V. Numerical evaluations are used to verify the performance of the protocol in Section VI before we conclude the paper with a summary and future research directions in Section VII.

## II. Related Work

Over the years, due to its importance, the TE problem has attracted a lot of research attention from different research communities. Below we provide a brief overview of major related existing results from different communities such as control, optimization and networking.

Broadly, the existing work can be divided into studies of heuristic improvements to OSPF [3,9,10] and studies of optimal routing algorithms. The former work was motivated by the need to improve the efficiency of OSPF after it had become the dominant routing protocol on the internet. While these techniques have been shown to improve the performance of OSPF by finding better weight settings for the algorithm, the results are far from optimal. Since we are interested in an optimal TE algorithm that uses the same information as OSPF, instead of further exploring the sub-optimal algorithms, we will focus on reviewing optimal solution techniques.

The optimal traffic distribution is obtained by the solution of a multicommodity flow problem. Since OSPF and IS-IS owe their scalability to traffic forwarding done via local calculations, we will adopt a decentralized approach in our search for an optimal protocol. The decentralized solution techniques for this problem that have been proposed so far can be broadly classified into protocols that are distributed per node and protocols that are distributed per commodity or source-destination pair.

The class of decentralized algorithms that are node-based are distance-vector protocols and include the ones proposed by Gallager in his classic paper [4], Stern [11] and Agnew [12]. The idea behind these algorithms was that as long as a node was aware of the "price" ("average distance") to each destination at each of its neighbors, it had enough information to make optimal forwarding decisions. From an optimization standpoint, this is a natural and mathematically elegant approach since the main ideas follow directly from the decomposition of the dual of the TE optimization problem. Decompositions like this, which have been very successful for problems of this type [13], can be used to yield updating rules for primal and dual variables (split ratios and node prices in [4]) that can be shown to converge to optimal solutions. Similar ideas have also been applied to cross-layer optimization of networks [14,15].

[1]When the optimal routing solution does not use all available paths to the destination, as is the case in many networks, for example, any network with a loop, a set of finite optimal weights for PEFT [8] does not exist.

TABLE I: Comparison of Routing Algorithms

| Algorithm | Link-state | Hop-by-hop | Optimal | Adaptive |
|---|---|---|---|---|
| OSPF | ✓ | ✓ | × | × |
| Gallager's [4] | × | ✓ | ✓ | ✓ |
| Projected Gradient [6] | ✓ | × | ✓ | ✓ |
| PEFT [8] | ✓ | ✓ | × | × |
| HALO | ✓ | ✓ | ✓ | ✓ |

On the other hand, we have optimal link-state routing algorithms that are decentralized by source-destination pairs. Examples of this variety include the flow deviation technique advocated by Fratta et al [2], projection methods proposed by Bertsekas and Gafni [6] as well as proximal decomposition methods [16]. These solutions are based on iteratively calculating a shortest path at the source for each commodity and transferring varying amounts of flow from the non-shortest paths to the shortest path to obtain the optimal traffic distribution. An important limitation of these algorithms is that, as noted earlier, they require source routing, i.e., the route a packet will take through the network has to be completely encoded at the source.

The preceding overview of how the literature has developed in this area, clearly reveals the missing link in the search for an optimal link-state hop-by-hop routing algorithm. In this paper, we provide this missing link by introducing HALO. A comparison of several existing solutions and ours can be seen in Table I.

## III. Problem Formulation

The optimization problem that is used for traffic engineering is the Multi-Commodity Flow problem (MCF). For a given directed graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ with node/router set $\mathbb{V}$ and edge/link set $\mathbb{E}$ with link capacities $c_{u,v},\ \forall (u,v) \in \mathbb{E}$, and demands $D(s,t)$ defined as the rate required for communication from $s$ to $t$, the MCF problem can been summarized below.

$$\min_{f^t_{u,v}} \ \Phi(f)$$

$$s.t. \sum_{v:(s,v)\in\mathbb{E}} f^t_{s,v} - \sum_{u:(u,s)\in\mathbb{E}} f^t_{u,s} = D(s,t),\ \forall s \neq t$$

$$f_{u,v} = \sum_{t \in \mathbb{V}} f^t_{u,v} \leq c_{u,v}, \ \forall (u,v)$$
$$f^t_{u,v} \geq 0$$

Here commodities are defined in terms of their final destination $t$. $f^t_{u,v}$ is the flow on link $(u,v)$ corresponding to commodity $t$ and $f_{u,v}$ is the total flow on link $(u,v)$. The cost function, $\Phi$, is typically selected to be a convex function of the link rate vector $f = \{f_{u,v}\}, \ \forall (u,v) \in \mathbb{E}$. For example, if we use the M/M/1 delay formula for the cost function, then $\Phi(f) = \sum_{u,v} \Phi_{u,v}(f_{u,v}) = \sum_{u,v} f_{u,v}/(c_{u,v} - f_{u,v})$ [1]. Throughout the paper, for numerical examples, we will use this cost function unless specified otherwise. It is also assumed that $\Phi'_{u,v}(f_{u,v}) \to \infty$ when $f_{u,v} \to c_{u,v}$. This captures the common practice of not allowing links to operate too close to capacity. In this paper, given a function $\gamma(x(\tau))$, we will use $\gamma'$ to represent the derivative of $\gamma$ with respect to $x$ and $\dot{\gamma}$ to represent the time ($\tau$) derivative of $\gamma$.

We also define the price of a link $(u,v)$ as $w_{u,v} = \Phi'_{u,v}(f_{u,v})$, the price of a path $p$ as $\sum_{(u,v)\in p} w_{u,v}$ and the price at a node $u$ to a destination $t$ as,

$$q^t_u = \sum_{v:(u,v)\in\mathbb{E}} \alpha^t_{u,v}[w_{u,v} + q^t_v] \tag{1}$$

where $q^t_t = 0$. The price at a node can be interpreted as the average price to the destination from that node where the average is taken over all outgoing edges to the destination weighted by the split ratios along those edges. If instead the average is done over all possible paths, Equation (1) can be stated without recursion as,

$$q^t_u = \sum_{p \in P_{u,t}} d_p \prod_{(i,j)\in p} \alpha^t_{i,j} \tag{2}$$

where $P_{u,t}$ is the set of all paths from $u$ to $t$ and $d_p = \sum_{(u,v)\in p} w_{u,v}$.

A fact about MCF is that its optimal solution generally results in multi-path routing instead of single-path routing [17]. However, finding the right split ratios for each router for each commodity is a difficult task. Our starting point is to merge the link-state feature of protocols that were decentralized by source-destination pairs with the hop-by-hop forwarding feature of the node-based decentralization schemes. More concretely we,

- adjust each router's split ratios and move traffic from one outgoing link to another. This only controls the next hop on a packet's path leading to hop-by-hop routing. If instead we controlled path rates we would get source routing.
- increase the split ratio to the link which is part of the shortest path even though the average price via the next hop node may not be the lowest. If instead we forwarded traffic via the next hop node with the lowest average price we get Gallager's approach, which is a distance vector solution.
- adapt split ratios dynamically and incrementally by decreasing along links that belong to non-shortest paths while increasing along the link that is part of the shortest path at every router. If instead split ratios are set to be positive instantaneously only to the links leading to shortest paths, then we get OSPF with weights, $w_{u,v}$.

## IV. Special Cases

In order to develop an intuitive understanding of why our solution takes its current form, it is helpful to consider a few concrete special cases first. These four cases, each of which clearly highlights the reason for including a particular factor in our solution, progressively lead us to the final algorithm, which we will prove to always work for any general case in Section V. The calculations in this section also give hints for the main idea of the proof.

### A. Finding the Right Split Dynamically

First let us consider a very simple example illustrated in Figure 1a. It is worth noting that the KKT optimality conditions [18] of the MCF problem require that at the optimal solution the traffic rate is positive only along paths with the lowest price. In this example, assuming initially $w_l > w_s$, a simple strategy to reach optimality might be to dynamically shift traffic from the more expensive link to the cheaper link at some rate $\delta > 0$ till the prices of the two links become the same. In terms of the change in split ratios at node $A$ this would be equivalent to decreasing $\alpha_l$ and increasing $\alpha_s$ at rate $\delta/r$.

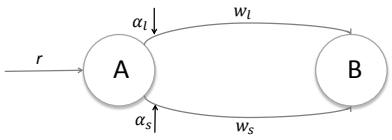
There are two ways to interpret and generalize the intuition gained from this scenario. Both give the same solution for this very simple example but in general will lead to different dynamics (see Figure 2) and possibly even different split ratios. One interpretation, which forms the basis of the technique proposed by [4], is that the router shifts traffic headed to neighbor nodes with higher average price to the neighbor node with the lowest average price. A different interpretation, which is the basis of our protocol, is that the router shifts traffic from links along more expensive paths to the link along the path with the lowest price. Mathematically, we reach the following update rule for the split ratios,

$$\dot{\alpha}^t_{u,v} = -\frac{\delta}{r^t_u} \tag{3}$$

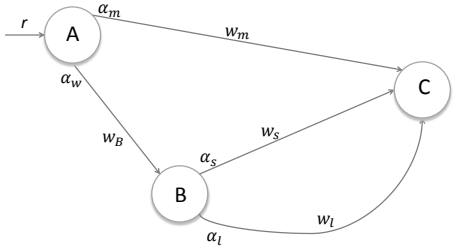
where $(u,v) \in \mathbb{E}$ but is not on the shortest path from $u$ to destination $t$ and $r^t_u$ is the incoming rate at node $u$ to destination $t$.

### B. A First Test

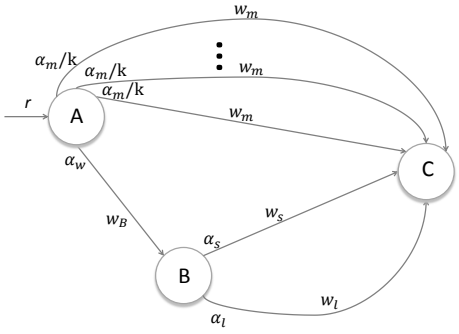
However, as a potential counter-example to this interpretation, it is possible to suggest some version of the scenario described in Figure 1b. Here there is traffic demand of rate $r$ from router $A$ to router $C$. The initial splits at router $A$ are $\alpha_m$ along an intermediate price link with price $w_m$ and $\alpha_w$ along the more expensive route with price $w_B + w_l$, assuming $\alpha_l = 1$ initially. The relationship between the initial link prices are assumed to be $w_l > w_m > w_s + w_B$, i.e., link $(A,B)$ is along the shortest path from $A$ to $C$, but $B$ also has the most expensive way to reach $C$. The concern is that router



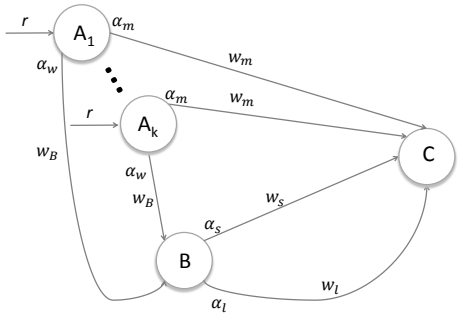

(a) **Finding the right split dynamically.** Suppose there is a single demand of rate $r$ trying to get to destination $B$. Initially, the split ratios at $A$ are $\alpha_l$ along the more expensive ("longer") link with price $w_l = \Phi_l'(\alpha_l r)$ and $\alpha_s$ along the cheaper ("shorter") link with price $w_s = \Phi_s'(\alpha_s r)$.




(b) **A first test.** Suppose the link weights are as shown and $w_l > w_m > w_s + w_B$. There is a single demand $D(A, C) = r$.




(c) **Multiple outgoing paths.** Suppose the link weights are as shown and $w_l > w_m > w_s + w_B$. There is a single demand $D(A, C) = r$.




(d) **Multiple inputs.** Suppose the link weights are as shown and $w_l > w_m > w_s + w_B$. There are $k$ demands $D(A_i, C) = r,\ i = 1, \ldots, k$.

Fig. 1: Four illustrative examples

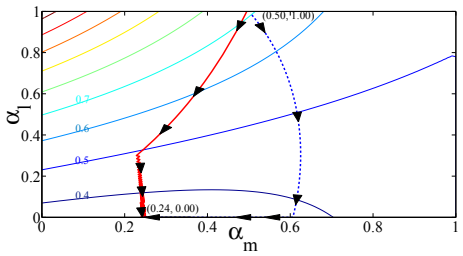



Fig. 2: Trajectories taken by Gallager's algorithm (dashed line) and HALO (solid line) to converge to the optimal solution. Cost values are shown for some contour lines.

$A$ shifting traffic from the intermediate price link to the link with price $w_B$ might result in the cost increasing as router $B$ currently routes traffic only through the most expensive link ($\alpha_l = 1$). But because router $B$ decreases $\alpha_l$ and increases $\alpha_s$ (in conjunction with the changes at router $A$), the total cost does in fact decrease. More precisely, the cost derative can be calculated as follows,

$$\begin{aligned}\dot{\Phi} =& -r \times \frac{\delta}{r} \times w_m + r \times \frac{\delta}{r} \times (w_B + w_l) \\ & - r_B \times \frac{\delta}{r_B} \times w_l + r_B \times \frac{\delta}{r_B} \times w_s \\ =& -\delta(w_m - w_B - w_s) \leq 0\end{aligned}$$

where $r_B$ is the incoming rate to $C$ at $B$ (superscript dropped for convenience since $C$ is the only destination) and the inequality follows from the relationship between the prices.

This particular example can also be used to illustrate the difference between our approach and Gallager's technique which arises from the fact that the link leading to the neighbor with the lowest average price (path A-C with price $w_m$) may not lead to the cheapest path (path A-B-C with price $w_B + w_s$). Figure 2 shows the trajectories taken by the two different algorithms to converge to the optimal solution for this topology. In order to simulate the long link between node $B$ and node $C$, an intermediate dummy node $D$ is introduced that splits the bottom link between $B$ and $C$ into two equal capacity links. The capacities used were $(A, B) = 5, (B, C) = 10, (A, C) = (B, D) = (D, C) = 3$. The rate $r = 1$ and initially $\alpha_w = \alpha_m = 0.5$ and $\alpha_l = 1$. Here we take only one split ratio at each node because the value of that split ratio automatically defines the value of the other at each node. Using Gallager's algorithm, initially, as can be seen, following the lowest average price path to the destination $(A, C)$, there is an increase in the value of $\alpha_m$. Also, as expected from theory, the trajectory of the algorithm (gradient descent) is perpendicular to the objective function contour curves. On the other hand, using HALO, both split ratios are decreased initially. HALO's trajectory is usually not perpendicular to the counter curves, however, it still goes along a descent direction and drives the total cost down.

### C. Multiple Outgoing Paths

The above case study might lead us to ask whether the simple rule developed thus far (Equation (3)) is sufficient to

guarantee decreasing network cost along any trajectory. In order to see why it is not, consider the situation presented in Figure 1c. Now there are $k$ intermediate price links from router $A$ to router $C$ each of which gets $\alpha_m/k$ fraction of the demand. The relationship between the link prices is the same as in the previous example. Now the concern is that shifting traffic in an unrestricted fashion from the intermediate price links to router $B$ with $\alpha_l = 1$, might result in an increase in the cost and it is a valid concern as illustrated by the following calculation.

$$\begin{aligned}\dot{\Phi} =& -k \times r \times \frac{\delta}{r} \times w_m + k \times r \times \frac{\delta}{r} \times (w_B + w_l) \\ & - r_B \times \frac{\delta}{r_B} \times w_l + r_B \times \frac{\delta}{r_B} \times w_s \\ =& -k\delta w_m + \delta(kw_B + w_s) + (k-1)\delta w_l\end{aligned}$$

which may be positive for $k > 1$. But this problem can be surmounted by modifying the update rule followed by the split ratios by adding a weighting factor of the split ratio itself. Mathematically, we have

$$\dot{\alpha}^t_{u,v} = -\frac{\alpha^t_{u,v}\delta}{r^t_u} \tag{4}$$

where $(u, v) \in \mathbb{E}$ but is not on the shortest path from $u$ to destination $t$.

With this new rule, the cost derivative can be evaluated as,

$$\begin{aligned}\dot{\Phi} =& -k \times r \times \frac{\delta\alpha_m}{rk} \times w_m + kr \times \frac{\delta\alpha_m}{rk} \times (w_B + w_l) \\ & - r_B \times \frac{\delta}{r_B} \times w_l + r_B \times \frac{\delta}{r_B} \times w_s \\ =& -\delta[\alpha_m w_m + (1-\alpha_m)(w_B + w_l)] + \delta(w_B + w_l) \\ & - \delta w_l + \delta w_s \\ =& -\delta[\alpha_m w_m + (1-\alpha_m)(w_B + w_l)] + \delta(w_B + w_s) \\ \leq& 0\end{aligned}$$

where the last inequality follows from the fact that the average price from router $A$ to $C$, which is $\alpha_m w_m + (1-\alpha_m)(w_B + w_l)$ has to be at least as large as the price of the shortest path from $A$ to $C$, which is $w_B + w_s$.

### D. Multiple Inputs

Does Equation (4) ensure that total cost decreases along its trajectory? Another case worth considering is illustrated in Figure 1d. This time there are $k$ sources $A_1, \ldots, A_k$ that have data to send to router $C$. Now the concern is that shifting traffic in an unrestricted manner from all the sources to router $B$ with $\alpha_l = 1$, could cause the total cost to increase as shown by the calculations below,

$$\begin{aligned}\dot{\Phi} =& -k \times r \times \frac{\delta\alpha_m}{r} \times w_m + k \times r \times \frac{\delta\alpha_m}{r} \times (w_B + w_l) \\ & - r_B \times \frac{\delta}{r_B} \times w_l + r_B \times \frac{\delta}{r_B} \times w_s \\ =& -k\delta[\alpha_m w_m + (1-\alpha_m)(w_B + w_l)] + (k-1)\delta w_l \\ & + \delta(kw_B + w_s)\end{aligned}$$

which may be positive for $k > 1$. Once again it is possible to modify the update rule for the split ratios from $\delta\alpha^t_{u,v}/r^t_u$ to $\delta\alpha^t_{u,v}/\eta^t_u r^t_u$. In this case, $\eta^t_u = k$ while for a general network we will specify how to calculate $\eta^t_u$ in Section V, Algorithm 1. With the new rule, the cost derivative will be the same as in Section IV-C, and always no more than zero.

Formally, the above discussion leads us to further modify the update rule in Equation (4) to

$$\dot{\alpha}^t_{u,v} = -\frac{\alpha^t_{u,v}\delta}{\eta^t_u r^t_u} \tag{5}$$

where $(u, v) \in \mathbb{E}$ but is not on the shortest path from $u$ to destination $t$. In the following section we will show that for any network, this update rule for the split ratios (5) makes the total cost of the network always decrease, resulting in the split ratios converging to a set where every element of the set achieves the global optimum to the MCF problem, and therefore achieves optimal TE.

## V. General Solution

We first introduce some additional necessary notation. For a particular destination $t$ at node $s$ we define,

$$r^t_s = \sum_{u:(u,s)\in\mathbb{E}} f^t_{u,s} + D(s,t)$$

the inflow rate to a node $s$ destined to $t$ which because of node flow balance requirements is also the outflow at $s$ to $t$. We will also use $\alpha$ without indexing to represent the set of all the split ratios from all the routers in the network. We have already noted that at a router $u$, $\alpha^t_{u,v}$ controls the fraction of traffic to destination $t$ that uses outgoing link $(u, v)$ while satisfying $\alpha^t_{u,v} \geq 0$ and $\sum_{v:(u,v)\in\mathbb{E}} \alpha^t_{u,v} = 1$.

Next, we define $\eta^t_u$, the *branch cardinality*, as the product of the number of branches encountered in traversing the shortest path tree rooted at $t$ from $t$ to $u$. Being a link-state routing algorithm, each node $u$ has the link-state information to run Dijkstra's algorithm to compute the shortest path tree to destination $t$. Here additional care is required because every node has to independently arrive at the same shortest path tree to ensure that the algorithm proceeds as expected. So at any stage of Dijkstra's algorithm, if there is ambiguity as to which node should be added next, tie-breaking based on node index is used. The calculation of $\eta^t_u$ proceeds as shown in Algorithm 1. For an illustration of how $\eta^t_u$ is calculated, please refer to the example following the proof of Theorem 1.

**Algorithm 1** Algorithm to calculate $\eta^t_u$ $\{w_e, \forall e \in \mathbb{E}\}$

1: Compute shortest path tree for destination $t$ using Dijkstra's algorithm with tie-breaking based on node index
2: Traverse the tree from $t$ to $u$
3: Initialize $\eta^t_u \leftarrow 1$
4: At every junction do $\eta^t_u \leftarrow \eta^t_u b$ where $b$ is the number of branches from that junction

We are now in a position to describe the adaptive link-state routing algorithm. For any node $u$, it controls the evolution of

the destination specific split ratio $\alpha^t_{u,v}$. Suppose that $(u,\bar{v}) \in \mathbb{E}$ and $(u,\bar{v})$ is part of the shortest path to $t$ from $u$. Then HALO calculates the split ratios as follows.

$$\text{if } r^t_u > 0, \qquad \dot{\alpha}^t_{u,v} = -\frac{\alpha^t_{u,v}\delta}{\eta^t_u r^t_u},\ v \neq \bar{v} \quad (6)$$

$$\dot{\alpha}^t_{u,\bar{v}} = -\sum_{v:(u,v)\in\mathbb{E},\ v\neq\bar{v}} \dot{\alpha}^t_{u,v} \quad (7)$$

$$\text{else if } r^t_u = 0, \qquad \alpha^t_{u,v} = 0,\ v \neq \bar{v} \quad (8)$$

$$\alpha^t_{u,\bar{v}} = 1 \quad (9)$$

We present the formal description of HALO in Algorithm 2.

**Algorithm 2** Forwarding Algorithm at Router $u$ $\{w_e, \forall e \in \mathbb{E}\}$

**for all** $t$ **do**
  Calculate $\eta^t_u$
  **if** $r^t_u == 0$ **then**
    **for all** $v \neq \bar{v}$ **do**
      $\alpha^t_{u,v} = 0$
    **end for**
    $\alpha^t_{u,\bar{v}} = 1$
  **else**
    **for all** $v \neq \bar{v}$ **do**
      $\dot{\alpha}^t_{u,v} = -\frac{\alpha^t_{u,v}\delta}{\eta^t_u r^t_u}$
    **end for**
    $\dot{\alpha}^t_{u,\bar{v}} = -\sum_{v:(u,v)\in\mathbb{E},v\neq\bar{v}} \dot{\alpha}^t_{u,v}$
  **end if**
**end for**

To prove the optimality of the above link-state hop-by-hop algorithm, we will need the following two lemmas. The first one was derived by Gallager [4], which relates the node prices to the link weights for each destination $t$.

**Lemma 1.** $\sum_{u\in\mathbb{V}} D(u,t) q^t_u = \sum_{(u,v)\in\mathbb{E}} f^t_{u,v} w_{u,v}$

It analytically states the intuitive idea that the total price of sending traffic to meet the demand in the network, as defined by the sum of the products of the traffic demand rate and the node price for each demand node, is equal to the sum over all links of the price of sending traffic through each link. The next lemma describes how to calculate the rate of change of network cost [14].

**Lemma 2.**

$$\sum_{(u,v)\in\mathbb{E}} \dot{f}^t_{u,v} w_{u,v} = \sum_{u\in\mathbb{V}} \sum_{(u,v)\in\mathbb{E}} r^t_u \dot{\alpha}^t_{u,v}[w_{u,v} + q^t_v]$$

The above expression captures the fact that the change in network cost can either be expressed in terms of the change in the link flow rates, i.e., how each link affects the network cost or in terms of the change in the split ratios at each node, i.e., how each node affects the network cost. Now we are finally in a position to prove the main result of the paper which is summarized in the following theorem.

**Theorem 1.** *In a network, at every node u, for every destination t, let the evolution of the split ratios be defined by equations* $(6)-(9)$. *Then starting from any initial conditions we have,*

*Convergence:* $\alpha$ *converges to the largest invariant set in* $\{\alpha|\dot{\Phi}(f) = 0\}$
*Optimality: any element of this set yields an optimal solution to the MCF problem.*

*Proof:* We will prove the result in three steps. First, we will show that $\dot{\Phi}(f) \leq 0$, which is the key step of the whole proof. Then, we will use this result to invoke LaSalle's Invariance Principle for hybrid systems [19] to argue that $\alpha$ converges to the largest invariant set in $\{\alpha|\dot{\Phi}(f) = 0\}$. Lastly, we will establish that any element of this set is an optimal solution to the MCF problem.

Step 1 (Monotonicity): Note that,

$$\dot{\Phi}(f) = \sum_{t\in\mathbb{V}} \sum_{(u,v)\in\mathbb{E}} \dot{f}^t_{u,v} w_{u,v} = \sum_{t\in\mathbb{V}} \dot{\Phi}^t(f)$$

where $\dot{\Phi}^t(f) = \sum_{(u,v)\in\mathbb{E}} \dot{f}^t_{u,v} w_{u,v}$ is the rate of change of the network cost as the flows to destination $t$ change. Consequently, if we show that $\dot{\Phi}^t(f) \leq 0$ for each destination $t$, then we have that $\dot{\Phi}(f) \leq 0$. From Lemma 2,

$$\dot{\Phi}^t(f) = \sum_{(u,v)\in\mathbb{E}} \dot{f}^t_{u,v} w_{u,v} = \sum_{u\in\mathbb{V}} \sum_{(u,v)\in\mathbb{E}} r^t_u \dot{\alpha}^t_{u,v}[w_{u,v} + q^t_v]$$

The key idea behind step 1 is to decompose the change in cost to a particular destination $t$, by grouping the terms from the summation derived in Lemma 2, using the branches of the shortest path tree rooted at that destination. More precisely, we define a *branch* ($\mathcal{B}$) as the set of nodes on the path from a leaf node on the shortest path tree to the destination node $t$. Given the definition, it is easy to see that some intermediate nodes will be shared among multiple branches. The change in cost contributed by these nodes is properly divided among the different branches that pass through these routers in the following way. Each node $u$ has a corresponding $\eta^t_u$ value which appears in the denominator of the expression for the change in cost. The idea is, when grouping terms, for a particular branch passing through an intermediate node, to only take a fraction, $1/\pi^{\mathcal{B}}_u$, of the change in cost contributed by the intermediate node, to be summed with that branch so that $\pi^{\mathcal{B}}_u \eta^t_u$ for that node $u$ is the same as the branch cardinality of the leaf router which defines the branch. Consequently, $\pi^{\mathcal{B}}_u \eta^t_u$ will be the same for all routers $u$ encountered in a traversal from the leaf router of the branch to the destination. Given the definition of $\eta^t_u$ and $\pi^{\mathcal{B}}_u$, one can check $\sum_{\mathcal{B}} \frac{1}{\pi^{\mathcal{B}}_u} = 1$, so the total contribution from node u is distributed over different branches. Formally, we have,

$$\sum_{u\in\mathbb{V}} \sum_{(u,v)\in\mathbb{E}} r^t_u \dot{\alpha}^t_{u,v}[w_{u,v} + q^t_v] =$$
$$\sum_{\forall\mathcal{B}} \sum_{u\in\mathcal{B}} \frac{1}{\pi^{\mathcal{B}}_u} \sum_{(u,v)\in\mathbb{E}} r^t_u \dot{\alpha}^t_{u,v}[w_{u,v} + q^t_v]$$

For a given branch $\mathcal{B}$, with $n$ nodes numbered $1,\ldots,n$ from the leaf node to the destination, as noted above, $1/\pi^{\mathcal{B}}_u$ is the

fraction of the change in cost due to node $u$ that it contributes to the branch summation. For ease of notation, in what follows, we will use $\eta$ to represent $\pi_u^{\mathcal{B}}\eta_u^t$ for every router $u$ that belongs to the branch $\mathcal{B}$. For any $u \in \{1, 2, \ldots, n-1\}$, we have the following important equation.

$$\frac{1}{\pi_u^{\mathcal{B}}} \sum_{(u,v)\in\mathbb{E}} r_u^t \dot{\alpha}_{u,v}^t [w_{u,v} + q_v^t] = -\frac{\delta}{\eta}(q_u^t - w_{u,u+1} - q_{u+1}^t) \tag{10}$$

Note if $r_u^t = 0$, following equations (8) and (9), the left hand side of (10) is zero because $\dot{\alpha}_{u,v}^t = 0$, the right hand side of (10) is also zero because $\alpha_{u,u+1}^t = 1$. If $r_u^t > 0$, (10) is still valid because

$$\begin{aligned}
&\frac{1}{\pi_u^{\mathcal{B}}} \sum_{(u,v)\in\mathbb{E}} r_u^t \dot{\alpha}_{u,v}^t [w_{u,v} + q_v^t] \\
&= -\frac{\delta}{\eta}\Big(\sum_{(u,v)\in\mathbb{E}} \alpha_{u,v}^t [w_{u,v} + q_v^t] - \sum_{(u,v)\in\mathbb{E}} \alpha_{u,v}^t [w_{u,u+1} + q_{u+1}^t]\Big) \\
&= -\frac{\delta}{\eta}(q_u^t - w_{u,u+1} - q_{u+1}^t)
\end{aligned}$$

Therefore

$$\begin{aligned}
&\sum_{u\in\mathcal{B}} \frac{1}{\pi_u^{\mathcal{B}}} \sum_{(u,v)\in\mathbb{E}} r_u^t \dot{\alpha}_{u,v}^t [w_{u,v} + q_v^t] \\
&= \sum_{u=1}^{n-1} -\frac{\delta}{\eta}(q_u^t - w_{u,u+1} - q_{u+1}^t) \\
&= -\frac{\delta}{\eta}[q_1^t - w_{1,2} - \cdots - w_{n-1,n}] \\
&\leq 0
\end{aligned}$$

The last inequality follows from the fact that the average price from the leaf router (node 1) to the destination (node $n$) which can be thought of as an average over paths from Equation (2) has to be no less than the price of the shortest path. Note that this relationship holds with equality only when the node price of the leaf node is the same as the price of the shortest path which means that all the traffic from every node in the branch to the destination is along shortest paths to the destination.

We then have

$$\dot{\Phi} = \sum_t \dot{\Phi}^t(f) = \sum_{(u,v)\in\mathbb{E}} \dot{f}_{u,v}^t \Phi'(f_{u,v}) \leq 0 \tag{11}$$

Step 2 (Convergence): Given the control laws we have established that $\dot{\Phi}(f) \leq 0$. In order to show convergence, we will rely on the language of hybrid automata [19] to model the dynamics of our system. Specifically, our system is an example of a non-blocking, deterministic and continuous hybrid automaton. Consequently, a generalization of LaSalle's Invariance Principle to hybrid automata [19] ensures that the set of split ratios converges to the largest invariant set within $\{\alpha | \dot{\Phi}(f) = 0\}$.

Step 3 (Optimality): The only way to have $\dot{\Phi}(f) = 0$ is if each $\dot{\Phi}^t(f) = 0$ which implies that the change in cost along each branch,

$$\sum_{u\in\mathcal{B}} \frac{1}{\pi_u^{\mathcal{B}}} \sum_{\substack{(u,v)\in\mathbb{E} \\ \text{such that } u\in\mathcal{B}}} r_u^t \dot{\alpha}_{u,v}^t [w_{u,v} + q_v^t] = 0$$

for every $t$. From the preceding analysis, the change in cost along a branch $\mathcal{B}$ is zero only when all the traffic from the nodes that belong to the branch is being routed to the destination through shortest paths with respect to the link prices. Since this is a necessary and sufficient condition for optimality in MCF [1], the proof is complete. ■

Next, as an illustrative example to help understand the first step of the above proof, we consider a sample shortest path tree and perform the corresponding cost change calculations explicitly. Consider the shortest path tree of Figure 3. The number of branches that we divide the tree into is determined by the number of leaf nodes. In this example, the shortest path tree rooted at $t$ has 12 leaf routers and consequently we will divide the summation into 12 branches. Following the algorithm for the calculation of $\eta$, we find, $\eta_i^t = 1$, $\eta_h^t = 3$, $\eta_g^t = 9$ and $\eta_s^t = 18$. As noted in the proof, the change in the cost function due to the routers increasing traffic along the links in the shortest path tree can be calculated using Lemma 2. In order to evaluate it, we further divide the terms in the summation and group them per branch. Recall from the proof that for the routers that are downstream to a leaf router in a branch, only a fraction of the change in the cost contributed by the downstream router is selected where the fraction is determined by the need to have the same $\eta$ for all routers in the summation for a branch. The contribution to the change in the cost by the routers for the highlighted branch can be calculated as follows,

$$\begin{aligned}
&\sum_{u\in\mathcal{B}} \frac{1}{\pi_u^{\mathcal{B}}} \sum_{(u,v)\in\mathbb{E}} r_u^t \dot{\alpha}_{u,v}^t [w_{u,v} + q_v^t] \\
&= -r_s^t \sum_{(s,v)\in\mathbb{E}} \frac{\alpha_{s,v}^t \delta}{\eta_s^t r_s^t} [w_{s,v} + q_v^t] + r_s^t \sum_{(s,v)\in\mathbb{E}} \frac{\alpha_{s,v}^t \delta}{\eta_s^t r_s^t} [w_{s,g} + q_g^t] \\
&\quad - r_g^t \sum_{(g,v)\in\mathbb{E}} \frac{\alpha_{g,v}^t \delta}{2\eta_g^t r_g^t} [w_{g,v} + q_v^t] + r_g^t \sum_{(g,v)\in\mathbb{E}} \frac{\alpha_{g,v}^t \delta}{2\eta_g^t r_g^t} [w_{g,h} + q_h^t] \\
&\quad - r_h^t \sum_{(h,v)\in\mathbb{E}} \frac{\alpha_{h,v}^t \delta}{6\eta_h^t r_h^t} [w_{h,v} + q_v^t] + r_h^t \sum_{(h,v)\in\mathbb{E}} \frac{\alpha_{h,v}^t \delta}{6\eta_h^t r_h^t} [w_{h,i} + q_i^t] \\
&\quad - r_i^t \sum_{(i,v)\in\mathbb{E}} \frac{\alpha_{i,v}^t \delta}{18\eta_i^t r_i^t} [w_{i,v} + q_v^t] + r_i^t \sum_{(i,v)\in\mathbb{E}} \frac{\alpha_{i,v}^t \delta}{18\eta_h^t r_i^t} [w_{i,t}] \\
&= -\frac{\delta}{\eta_s^t}[q_s^t - w_{s,g} - w_{g,h} - w_{h,i} - w_{i,t}] \leq 0
\end{aligned}$$

## VI. Numerical Evaluation

In this section, we consider numerical evaluations of the performance of HALO from the point of view of optimality and rate of convergence to the optimal solution. We also present evidence of the adaptivity of the algorithm as the traffic changes. The evaluations are primarily performed on three networks – the benchmark Abilene network (Figure 4),

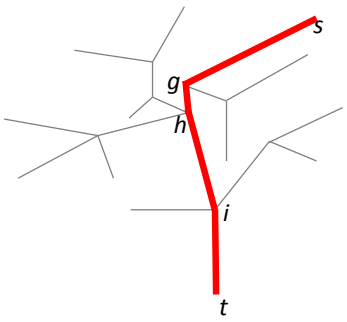



Fig. 3: **Shortest Path Tree.** Only the links in the shortest path tree for terminal $t$ is shown with the other links in the network not shown for ease of exposition.


Fig. 4: Abilene Network

a $4 \times 4$ Mesh network and a two-level hierarchical $50$ node network [3]. The $4 \times 4$ Mesh network is selected to study the affects of intermediate routing loops on the optimality of the algorithm as this topology is particularly prone to such loops while the hierarchical network is selected to mimic larger networks with high capacity backbone links and lower capacity local links. An additional test is performed on an even larger randomly generated $100$ node network in order to confirm that the algorithm converges quickly for large networks (Figure 11). Randomly generated traffic demands are used for the mesh network and the hierarchical network while for the Abilene network uniform traffic demand is used. In order to study the algorithms' performance, in all three cases, the demand is scaled up till at least one link in the network is close to saturation at the optimal solution.

### A. Convergence

As expected, the speed of convergence depends on the step-size. Here, the metric, network load, is defined as the ratio of the total traffic on the network to its total capacity. In general, smaller step-sizes guarantee convergence of the algorithm to the optimal solution at the expense of speed of convergence. This is demonstrated to be the case in Figure 6. But, as can be seen in Figure 6a and Figure 6c, larger step-sizes quickly approach the optimal solution though they can be prone to oscillations which prevent convergence to optimality. Often, it is sufficient to come to some neighborhood of the optimal solution and in such cases, exact convergence ceases to be an issue and small oscillations around the optimal solution are acceptable. In such situations, a larger step-size may be used. It is encouraging to note that in all our test cases, including for the larger $100$ node network (Figure 11) the algorithm was fairly quick, converging to a small neighborhood of the optimal solution within a few hundred iterations.

Another factor that affects the rate of convergence of the algorithm is the load on the network. The maximum network load for the Abilene network is $24.6\%$, mesh network is $26.1\%$ and the hierarchical network is $5.3\%$. These values indicate the point at which further scaling up the demand for the given traffic pattern would exceed the capacity of at least one link in the network, even with optimal routing. From Figure 5, we can see that the algorithm takes more iterations to converge to the optimal solution for more heavily loaded networks. Promisingly, even in such limiting cases, HALO converges to the optimal solution on the order of a thousand iterations. Given that today, link-state advertisements can be broadcast on the order of milliseconds [20], our evaluations indicate the possibility of convergence times of less than a second to a few seconds for the protocol when transmission delay is not a limiting factor.

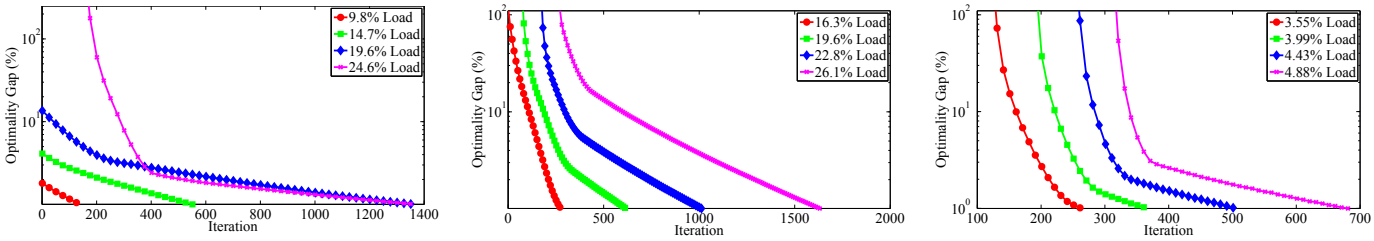
### B. Performance

In order to verify that the algorithm does in fact achieve the optimal solution, the optimal solution was calculated for the test networks by solving the corresponding MCF problem using cvx [21] under different network load conditions. The objective value obtained by using HALO matched the optimal solution for each test case as can be seen from Figures 7a, 7b and 7c. Also, as expected from theory the intermediate routing loops produced while determining the optimal solution for the mesh network did not affect the optimality of the algorithm.

The major advantage that HALO offers is the significant performance improvement that an optimal solution offers over sub-optimal techniques like OSPF even when it is aided by locally optimal weight settings. In Figure 8, we compare the performance of HALO with OSPF boosted by better weight settings obtained from the algorithms of the TOTEM toolbox [22] for demand matrices that placed increasing loads on the test networks. The local search algorithm used by TOTEM minimizes a piecewise-linear approximation of our convex cost function. The power of optimality is demonstrated by the performance improvements on the order of $1000\%$.

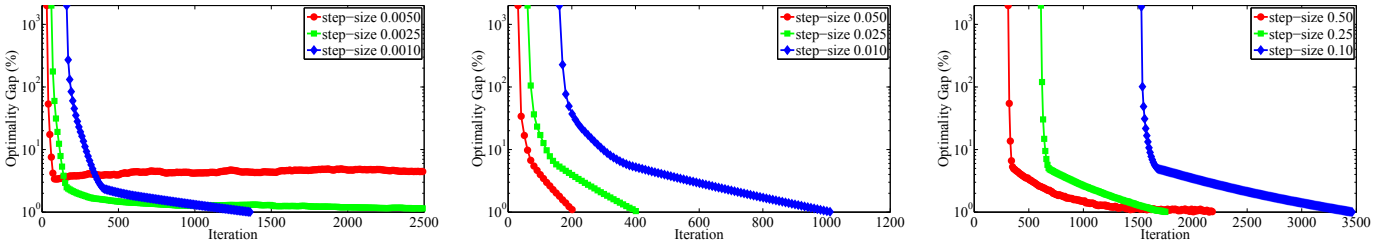
### C. Adaptivity

Another attraction of HALO is that it dynamically adapts to changes in the traffic on the network. In Figure 9, we plot the evolution of the optimality gap as the traffic matrix undergoes changes for the Abilene network under different network load conditions. In this example after around $300$ iterations the network load is changed by changing $20\%$ of the flows in the network. As can be seen, the algorithm quickly adapts and the optimality gap increases very little before beginning to converge to the new optimal solution. The traffic pattern is



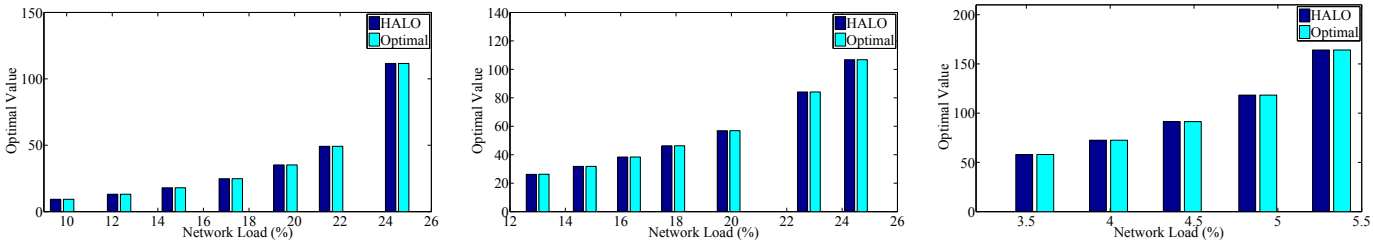

(a) Abilene (b) $4 \times 4$ Mesh network (c) Hierarchical 50 node network

Fig. 5: (a)Evolution of the optimality gap for the Abilene network as the number of iterations increases with different network loads (step-size = 0.001) (b) Evolution of the optimality gap for the $4 \times 4$ Mesh network as the number of iterations increases with different network loads (step-size = 0.01) (c)Evolution of the optimality gap for the Hierarchical 50 node network as the number of iterations increases with different network loads (step-size = 0.4)



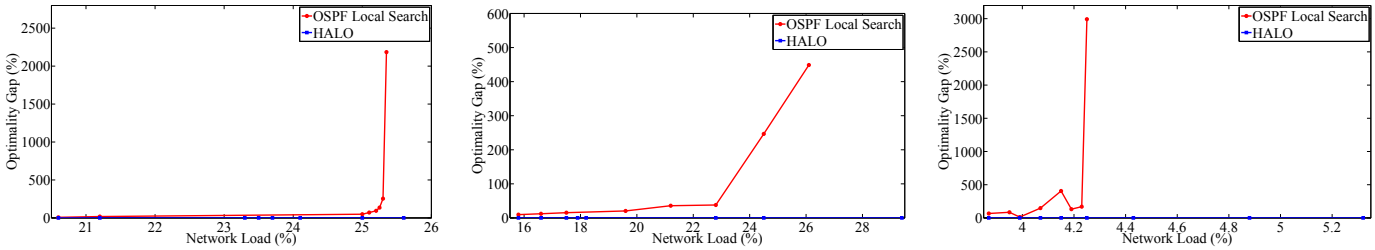

(a) Abilene (b) $4 \times 4$ Mesh network (c) Hierarchical 50 node network

Fig. 6: (a) Evolution of the optimality gap for the Abilene network as the number of iterations increases with varying step-sizes (network load = 24.6%) (b) Evolution of the optimality gap for the $4 \times 4$ Mesh network as the number of iterations increases with varying step-sizes (network load = 22.8%) (c) Evolution of the optimality gap for the Hierarchical 50 node network as the number of iterations increases with varying step-sizes (network load = 5.3%).




(a) Abilene (b) $4 \times 4$ Mesh Network (c) Hierarchical 50 Node Network

Fig. 7: (a) Mesh Network Optimal Performance (b)Abilene Network Optimal Performance (c)Hierarchical 50 Node Network Optimal Performance – Centralized optimal value matched by HALO




(a) Abilene (b) $4 \times 4$ Mesh Network (c) Hierarchical 50 Node Network

Fig. 8: (a) Abilene Network Algorithm Comparison (b) Mesh Network Algorithm Comparison (c)Hierarchical 50 Node Network Algorithm Comparison – Relative performance of different algorithms for different network loads

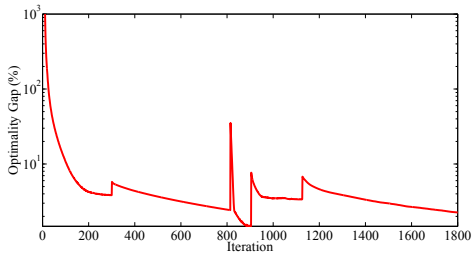



Fig. 9: Evolution of the optimality gap for the Abilene Network as the number of iterations increases with varying demand matrices

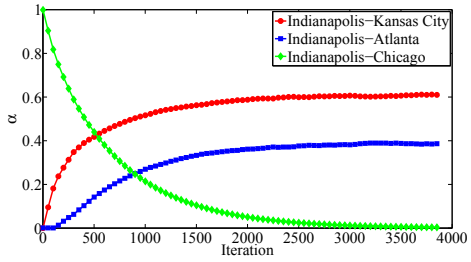



Fig. 10: Evolution of the split ratios to Chicago, Kansas City and Atlanta for traffic to LA at Indianapolis on the Abilene Network

again changed by varying $50\%$ of the flows in the network after $800$ iterations. This time the change in the optimality gap is greater but the convergence to the new optimal value is seen to be quicker. The traffic pattern in the network is changed two more times and as can be observed from the figure in both cases the algorithm quickly converges to the new optimal solution.

A closely related concept to the adaptivity of the algorithm is the evolution of the split ratios at individual routers. Additionally, it serves as a visualization of HALO in action. We pick the Indianapolis node for the Abilene network and plot the evolution of the split ratios to Los Angeles in Figure 10. For our test traffic, the initial sub-optimal allocation of split ratios is quickly corrected as HALO reduces traffic sent to Chicago and increases traffic sent to Kansas City and Atlanta.

## VII. Summary

In this paper we developed HALO, the first link-state, hop-by-hop routing algorithm that optimally solves the traffic engineering problem for intra-domain routing on the internet. The algorithm uses exactly the same information as OSPF. Furthermore, the link weights can be computed locally by routers and the algorithm automatically reacts to traffic demand changes by adjusting router split ratios. In terms of future directions, there are still important areas to be explored. For instance, the convergence rate of the algorithm needs to be analyzed. Another interesting direction involves using time averages for the link-states in order to test how well the algorithm performs without synchronous updates.

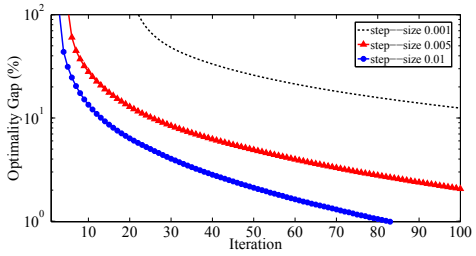



Fig. 11: Evolution of the optimality gap for a randomly generated $100$ node network with varying step-sizes

## References

[1] D. Bertsekas and R. Gallager, *Data Networks*. Prentice Hall, 1992.

[2] L. Fratta, M. Gerla, and L. Kleinrock, "The flow deviation method: An approach to store-and-forward communication network design," *Networks*, vol. 3, no. 2, pp. 97–133, 1973.

[3] B. Fortz and M. Thorup, "Increasing internet capacity using local search," *Comput. Optim. Appl.*, vol. 29, no. 1, pp. 13–48, Oct 2004.

[4] R. Gallager, "A minimum delay routing algorithm using distributed computation," *Communications, IEEE Transactions on*, vol. 25, no. 1, pp. 73 – 85, Jan 1977.

[5] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach, 5/E*. New York, NY, USA: Addison-Wesley, 2010.

[6] D. Bertsekas and E. Gafni, "Projected Newton methods and optimization of multicommodity flows," *Automatic Control, IEEE Transactions on*, vol. 28, no. 12, pp. 1090 – 1096, Dec 1983.

[7] D. Awduche, "MPLS and traffic engineering in IP networks," *Communications Magazine, IEEE*, vol. 37, no. 12, pp. 42 –47, Dec 1999.

[8] D. Xu, M. Chiang, and J. Rexford, "Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering," *Networking, IEEE/ACM Transactions on*, vol. 19, no. 6, pp. 1717 –1730, Dec 2011.

[9] A. Sridharan, R. Guerin, and C. Diot, "Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks," *Networking, IEEE/ACM Transactions on*, vol. 13, no. 2, pp. 234 – 247, Apr 2005.

[10] S. Srivastava, G. Agrawal, M. Pioro, and D. Medhi, "Determining link weight system under various objectives for OSPF networks using a lagrangian relaxation-based approach," *IEEE Trans. on Netw. and Serv. Manag.*, vol. 2, no. 1, pp. 9–18, Nov 2005.

[11] T. Stern, "A class of decentralized routing algorithms using relaxation," *Communications, IEEE Transactions on*, vol. 25, no. 10, pp. 1092 – 1102, Oct 1977.

[12] C. E. Agnew, "On quadratic adaptive routing algorithms," *Commun. ACM*, vol. 19, no. 1, pp. 18–22, Jan 1976.

[13] D. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 8, pp. 1439 –1451, Aug 2006.

[14] F. Paganini and E. Mallada, "A unified approach to congestion control and node-based multipath routing," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, pp. 1413–1426, Oct 2009.

[15] Y. Xi and E. Yeh, "Node-based optimal power control, routing, and congestion control in wireless networks," *Information Theory, IEEE Transactions on*, vol. 54, no. 9, pp. 4081–4106, Sep 2008.

[16] P. Mahey, A. Ouorou, L. J. LeBlanc, and J. Chifflet, "A new proximal decomposition algorithm for routing in telecommunication networks," *Networks*, vol. 31, no. 4, pp. 227–238, 1998.

[17] M. Wang, C. W. Tan, W. Xu, and A. Tang, "Cost of not splitting in routing: characterization and estimation," *IEEE/ACM Trans. Netw.*, vol. 19, no. 6, pp. 1849–1859, Dec 2011.

[18] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[19] J. Lygeros, K. Johansson, S. Simic, J. Zhang, and S. Sastry, "Dynamical properties of hybrid automata," *Automatic Control, IEEE Transactions on*, vol. 48, no. 1, pp. 2–17, Jan 2003.

[20] Cisco Systems Inc., "OSPF Link-State Advertisement (LSA) Throttling," http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/fsolsath.html, Nov 2012.

[21] CVX Research, Inc., "CVX: Matlab software for disciplined convex programming, version 2.0 beta," http://cvxr.com/cvx, Sep 2012.

[22] J. Lepropre, S. Balon, and G. Leduc, "Totem: A toolbox for traffic engineering methods," Poster and Demo Session of INFOCOM'06, Apr 2006.