End-to-End Restorable Oblivious Routing of Hose Model Traffic

M. Kodialam, *Associate Member, IEEE*, T. V. Lakshman, *Fellow, IEEE, ACM*, James B. Orlin, and Sudipta Sengupta, *Senior Member, IEEE*

***Abstract*—Two-phase routing, where traffic is first distributed to intermediate nodes before being routed to the final destination, has been recently proposed for handling widely fluctuating traffic without the need to adapt network routing to changing traffic. Preconfiguring the network in a traffic-independent manner using two-phase routing simplifies network operation considerably. In this paper, we extend this routing scheme by providing resiliency against link failures through end-to-end shared backup path restoration. We view this as important progress toward adding carrier-class reliability to the robustness of the scheme so as to facilitate its future deployment in Internet service provider (ISP) networks. In shared backup path restoration, each connection consists of a link-disjoint primary and backup path pair; two backup paths can share bandwidth on their common links if their primary paths are link-disjoint. We show that the optimization problem for maximum throughput two-phase routing with shared backup path restoration is NP-hard. Assuming an approximation oracle for a certain disjoint paths problem (called SBPR-DISJOINT-PATHS, which is also NP-hard) involving the dual variables of a path indexed linear programming formulation for the problem, we design a combinatorial algorithm with provable guarantees. We also provide heuristics for finding approximating solutions to the SBPR-DISJOINT-PATHS problem. We evaluate the throughput performance and number of intermediate nodes in two-phase routing for the above and other restoration mechanisms for two-phase routing on actual ISP topologies collected for the Rocketfuel project and three research network topologies.**

***Index Terms*—End-to-end restoration, hose traffic model, oblivious routing, shared backup path restoration, two-phase routing, variable traffic.**

I. Introduction

With the rapid rise in new Internet-based applications, such as peer-to-peer and voice-over-IP, it has become increasingly important to accommodate widely varying traffic patterns in networks. This requires Internet service providers (ISPs) to accurately monitor traffic and to deploy mechanisms for adapting network routing to changing traffic patterns. This dynamic adaptation increases network operations complexity. To avoid this complexity, service providers would like to provision their networks such that the provisioning is robust to large changes in the traffic pattern. This has led to interest in the *hose traffic model* [5], where the assumption is that we have knowledge regarding the maximum traffic entering and leaving the network at each node, but we do not have knowledge of the actual traffic matrix itself. Several algorithms for routing traffic in the hose model have recently been proposed. These schemes [5], [6], [10] route traffic *directly from the source to the destination along fixed paths*.

A recently proposed approach is *two-phase routing* [11], [21]. Here, in the first phase, incoming traffic is sent from the source to a set of intermediate nodes in predetermined proportions and then, in the second phase, from the intermediate nodes to the final destination. The proportion of traffic that is distributed to each intermediate node in the first phase can depend on the intermediate nodes [11]. This routing scheme can be used in a wide variety of networking scenarios as discussed in [12]. Some examples are: 1) provisioning service overlays such as the Internet Indirection Infrastructure (i3) [20]; 2) provisioning virtual private networks (VPNs); 3) adding QoS guarantees to services that require routing through a network-based middlebox; and 4) reducing IP-layer transit traffic and handling extreme traffic variability in IP-over-Optical networks without dynamic reconfiguration of the optical layer.

In this paper, we extend this routing scheme by providing resiliency against link failures through shared backup path restoration. In this restoration scheme, each connection consists of a link-disjoint primary and backup path pair. Two backup paths can share bandwidth on their common links if their primary paths are link-disjoint. We view this as important progress toward adding carrier-class reliability to the robustness of the scheme so as to facilitate its future deployment in ISP networks. (Backup bandwidth can also be allocated in a dedicated manner. The focus on shared allocation in this paper is because of its reduced cost, the rarity of concurrent multiple-link failures in networks, and the increased complexity of the optimization problems that arises from sharing backup bandwidth.)

We show that the optimization problem for maximum throughput two-phase routing with shared backup path restoration is NP-hard. Assuming an approximation oracle for a certain disjoint paths problem (called SBPR-DISJOINT-PATHS, which is also NP-hard) involving the dual variables of a path indexed linear programming formulation for the problem, we design a combinatorial algorithm with provable guarantees. We also provide heuristics for finding approximating solutions to the SBPR-DISJOINT-PATHS problem.

Maximum link utilization is an important, but not the only, optimization metric for network routing. We focus on this metric in this paper because it is one of the most common metrics used in the literature, is used in capacity-planning decisions by

Manuscript received March 21, 2010; revised October 19, 2010; accepted December 20, 2010; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor P. Van Mieghem. Date of publication April 15, 2011; date of current version August 17, 2011.

M. Kodialam and T. V. Lakshman are with Bell Laboratories, Alcatel-Lucent, Murray Hill, NJ 07733 USA.

J. B. Orlin is with the Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

S. Sengupta is with Microsoft Research, Redmond, WA 98052 USA (e-mail: sudipta@lcs.mit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2011.2121918

ISPs, and is directly related to other metrics like link congestion. We define *throughput* as the reciprocal of maximum network utilization. This can be interpreted as the maximum factor by which traffic can be scaled and still remain feasible for the current routing under given link capacities.

This paper is structured as follows. In the remainder of this section, we briefly describe some notation, introduce the traffic variation model, and review related work. In Section II, we briefly discuss the two-phase routing scheme so as to provide context for this paper. In Section III, we propose the addition of shared backup restoration to the two-phase routing scheme, establish the hardness of the associated optimization problem, and develop combinatorial algorithms for the problem with provable guarantees (assuming an approximation oracle). In Section IV, we evaluate the throughput performance and number of intermediate nodes in two-phase routing for the above and other restoration mechanisms for two-phase routing on actual ISP topologies collected for the Rocketfuel project [19] and three research network topologies. We conclude in Section V.

A. Notation

We will work with a directed graph $G = (N, E)$ with node set N and (directed) link (edge) set E that models the network topology. In order to protect against link failures, we require that the network be biconnected, i.e., the removal of any single link (and its reverse) does not disconnect the graph. Let $|N| = n$ and $|E| = m$. The nodes in N are labeled $\{1, 2, \ldots, n\}$. The sets of incoming and outgoing links at node i are denoted by $E^-(i)$ and $E^+(i)$, respectively. We let (i, j) represent a directed link in the network from node i to node j. To simplify the notation, we will also refer to a link by e instead of (i, j). The capacity of link (i, j) will be denoted by u_{ij}, and its cost by c_{ij}.

We assume that a single-link failure brings down the link in both directions. This is motivated by the fact that ports (line cards) of network routers/switches are bidirectional, hence a port failure can affect traffic in both directions on the link. Thus, we will refer to link failures in the undirected sense. To simplify the notation, we use $\tilde{f}$ to denote the undirected link corresponding to link f. Let $\tilde{E}$ denote the set of undirected links $\tilde{e}$ for all $e \in E$, i.e., $\tilde{E} = \{\tilde{e} \,|e \in E\}$. Let $\hat{e}$ denote the reverse of link e (if it exists). Thus, the failure of links f and $\hat{f}$ will be referred to by the failure of the undirected link $\tilde{f}$. A working path for connection P, denoted by $W(P)$, is affected by the failure of link $\tilde{f}$ if $f \in W(P)$ or $\hat{f} \in W(P)$. We will denote the latter condition by $\tilde{f} \in W(P)$ and consider the (non)containment of undirected links in directed paths in the undirected sense. Also, we will use $B(P)$ to denote the backup path for connection P.

Because we are considering link failures in both directions, we need to define link diversity of two paths to reflect this. Throughout this paper, two paths are said to be link-disjoint if they do not have any common link *in the undirected sense*, i.e., for any link e on one path, the other path cannot contain link e or its reverse.

B. Traffic Variation Model

We consider a traffic variation model where the total amount of traffic that enters (leaves) an ingress (egress) node in the network is bounded by the total capacity of all external ingress links at that node (see Fig. 1). This is known as the *hose model* and

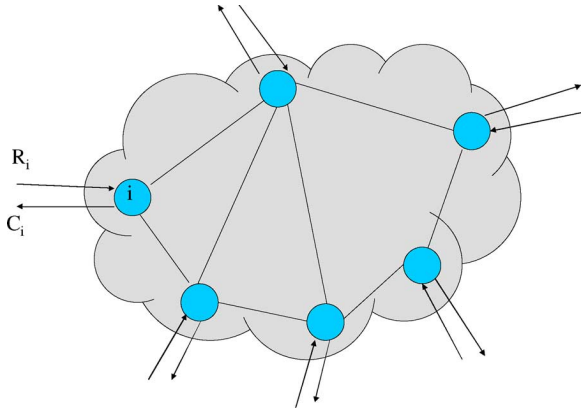

Fig. 1. Traffic model.

was proposed by Fingerhut *et al.* [7] and subsequently used by Duffield *et al.* [5] as a method for specifying the bandwidth requirements of a VPN. Note that the hose model naturally accommodates the network's ingress–egress capacity constraints.

We denote the upper bounds on the total amount of traffic entering and leaving at node i by R_i and C_i, respectively. The point-to-point matrix for the traffic in the network is thus constrained by these ingress–egress link capacity bounds. These constraints are the only known aspects of the traffic to be carried by the network, and knowing these is equivalent to knowing the row and column sum bounds on the traffic matrix. That is, any allowable traffic matrix $T = [t_{ij}]$ for the network must obey

$$\sum_{j:j\neq i}^{n} t_{ij} \leq R_i \qquad \sum_{j:j\neq i}^{n} t_{ji} \leq C_i \qquad \forall\, i \in N.$$

For given R_i and C_i values, denote the set of all such matrices that are partially specified by their row and column sums by $\mathcal{T}(\vec{R}, \vec{C})$, that is

$$\mathcal{T}(\vec{R}, \vec{C}) = \left\{ [t_{ij}] \Big| \sum_{j\neq i} t_{ij} \leq R_i \text{ and } \sum_{j\neq i} t_{ji} \leq C_i \;\; \forall\, i \right\}.$$

We will use $\lambda \cdot \mathcal{T}(\vec{R}, \vec{C})$ to denote the set of all traffic matrices in $\mathcal{T}(\vec{R}, \vec{C})$ with their entries multiplied by λ.

Note that the traffic distribution T could be any matrix in $\mathcal{T}(\vec{R}, \vec{C})$ and could change over time. Two-phase routing provides a routing architecture that does not make any assumptions about T apart from the fact that it is partially specified by row and column sum bounds and can provide QoS guarantees for routing all matrices in $\mathcal{T}(\vec{R}, \vec{C})$ without requiring any detection of changes in traffic patterns or dynamic network reconfiguration in response to it. For the Rocketfuel topologies, the throughput of two-phase routing is within 6% of the optimal scheme among the class of all schemes that are allowed to reconfigure the network in response to traffic changes [12].

C. Related Work

Direct routing from source to destination (instead of in two phases) along *fixed* paths for the hose traffic model has been considered by Duffield *et al.* [5] and Kumar *et al.* [10]. In related work, Applegate *et al.* [2] consider fixed path routing and

provide *relative guarantees* for routing an arbitrary traffic matrix with respect to the best routing for that matrix. However, they do not provide *absolute bandwidth guarantees* for routing variable traffic under the hose model.

Two aspects of direct source–destination path routing—namely: 1) the source needs to *know the destination of a packet* for routing it; and 2) the bandwidth requirements of the (fixed) paths change with traffic variations—render them unsuitable for some network architectures and applications. Because of 1), these methods cannot be used to provide *indirection in service overlay models* like i3 where the *final destination of a packet is not known at the source*. Because of 2), the adaptation of these methods for IP-over-Optical networks necessitates detection of changes in traffic patterns and dynamic reconfiguration of the provisioned optical-layer circuits in response to it, a functionality that is not present in current IP-over-Optical network deployments.

Applegate *et al.* [1] extend their work to cope with network failures. However, they do not consider any restoration mechanisms with preprovisioned backup paths. Instead, they provide linear programming formulations for rerouting affected traffic after failure. In addition to the above points in favor of two-phase routing, providing restoration through preprovisioned shared backup paths is another differentiating aspect of our current work.

II. Overview of Two-Phase Routing Scheme

In this section, we give an overview of the two-phase routing scheme from [11]. As mentioned earlier, the scheme does not require the network to detect changes in the traffic distribution or reconfigure the network in response to it. The only assumption about the traffic is the limits imposed by the ingress–egress constraints at each node, as outlined in Section I-B.

As is indicative from the name, the routing scheme operates in two phases.

- **Phase 1:** A predetermined fraction α_j of the traffic entering the network at any node is distributed to every node j *independent of the final destination of the traffic*.
- **Phase 2:** As a result of the routing in Phase 1, each node receives traffic destined for different destinations that it routes to their respective destinations in this phase.

This is illustrated in Fig. 2. Note that the traffic split ratios $\alpha_1, \alpha_2, \ldots, \alpha_n$ in Phase 1 of the scheme are such that $\sum_{i=1}^{n} \alpha_i = 1$. A simple method of implementing this routing scheme in the network is to form *fixed bandwidth paths between the nodes*. In order to differentiate between the paths carrying Phase 1 and Phase 2 traffic, we will refer to them as Phase 1 and Phase 2 paths, respectively. The critical reason the two-phase routing strategy works is that the *bandwidth required for these tunnels depends on the ingress–egress capacities R_i, C_i and the traffic split ratios α_j, but not on the (unknown) individual entries in the traffic matrix.*

We now derive the bandwidth requirement for the Phase 1 and Phase 2 paths. Consider a node i with maximum incoming traffic R_i. Node i sends $\alpha_j R_i$ amount of this traffic to node j during the first phase for each $j \in N$. Thus, the traffic demand from node i to node j as a result of Phase 1 routing is $\alpha_j R_i$.

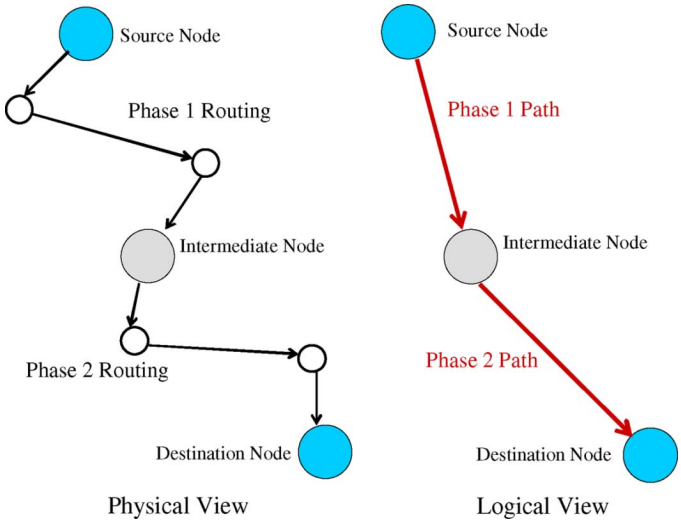

Fig. 2. Two-phase routing.

At the end of Phase 1, node i has received $\alpha_i R_k$ traffic from any other node k. Out of this, the traffic destined for node j is $\alpha_i t_{kj}$ since all traffic is initially split without regard to the final destination. The traffic that needs to be routed from node i to node j during Phase 2 is $\sum_{k \in N} \alpha_i t_{kj} \leq \alpha_i C_j$. Thus, the traffic demand from node i to node j as a result of Phase 2 routing is $\alpha_i C_j$.

Hence, the maximum demand from node i to node j as a result of routing in Phases 1 and 2 is $\alpha_j R_i + \alpha_i C_j$. Note that this does not depend on the matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$. The scheme handles variability in traffic matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$ by effectively routing the fixed matrix $D = [d_{ij}] = [\alpha_j R_i + \alpha_i C_j]$ that depends only on aggregate ingress–egress capacities and the traffic split ratios $\alpha_1, \alpha_2, \ldots, \alpha_n$, and not on the specific matrix $T \in \mathcal{T}(\vec{R}, \vec{C})$. This is what makes the routing scheme oblivious to changes in the traffic distribution.

An instance of the scheme requires specification of the traffic distribution ratios $\alpha_1, \alpha_2, \ldots, \alpha_n$ and routing of the Phase 1 and Phase 2 paths. In [12], linear programming formulations and fast combinatorial algorithms are developed for computing the above so as to maximize network throughput.

III. Adding Shared Backup Path Restoration to Two-Phase Routing

In order to make two-phase routing resilient to link failures, the Phase 1 and Phase 2 paths are protected using shared backup path restoration. We first describe this restoration scheme and then move on to the optimization problem of maximizing throughput.

A. Shared Backup Path Restoration

Under shared backup path restoration [4], a connection consists of a primary path and a link-disjoint backup path. Traffic is sent on the primary path during normal (no-failure) conditions and switched to the backup path after the primary is affected by a link failure. The sharing of backup bandwidth across different connections can occur in two possible ways as we will explain.

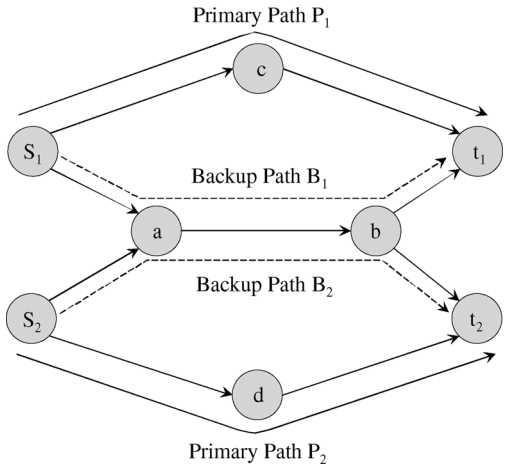

Fig. 3. Bandwidth sharing between backup paths in shared backup path restoration.

Both ways of sharing backup bandwidth guarantee that all connections affected by the failure of a single link have sufficient bandwidth on their backup paths to be completely restored.

First, two backup paths can share bandwidth on their common links provided their corresponding primary paths are link-disjoint (and, hence, cannot be affected simultaneously by a single-link failure). This sharing of bandwidth occurs across two backup paths for all possible single-link failure scenarios. This method of sharing is illustrated in Fig. 3, where two connections, one from s_1 to t_1 and the other from s_2 to t_2, have primary paths P_1, P_2 and backup paths B_1, B_2, respectively. Since primary paths P_1 and P_2 are link-disjoint, hence their backup paths B_1 and B_2 can share bandwidth on link a–b.

Second, if the primary paths of two connections have a link f in common, then the primary path of one connection and the backup path of another (and vice versa) can share bandwidth (on common links) under failure of link f. This is because after failure of link f, there will be no traffic on the primary path of either of the connections. This sharing of bandwidth occurs across a primary path and a backup path (of two different connections) for specific single-link failure scenarios (i.e., those links common to both primary paths).

B. Maximizing Throughput

Given a network with link capacities u_e and constraints R_i, C_j on the ingress–egress traffic, we consider the problem of two-phase routing with shared backup path restoration so as to maximize throughput. The throughput is the maximum multiplier λ such that all matrices in $\lambda \cdot \mathcal{T}(\vec{R}, \vec{C})$ can be feasibly routed with shared backup path restoration under given link capacities.

We will show that this problem is NP-hard. Hence, there is no polynomial-size linear programming formulation for the problem. We develop a primal-dual scheme by considering a path-flow-based linear programming formulation and adapting the technique in [8] for solving the maximum multicommodity flow problem, where flows are augmented in the primal solution and dual variables are updated in an iterative manner. Checking feasibility for the dual linear program reduces to finding a link-disjoint path pair between every pair of nodes that minimizes a certain cost metric involving the dual program variables. We show that this problem is strongly NP-hard and not approximable within any constant factor even in pseudopolynomial time unless $\mathrm{P} = \mathrm{NP}$. The approximation guarantee of the overall primal-dual method for our problem depends on how closely this disjoint path pair problem can be solved to optimality.

C. Path-Flow-Based LP Formulation

We begin with the path flow-based linear programming formulation for this problem. Let $\mathcal{PB}_{ij}$ denote the set of all link-disjoint primary–backup path pairs from node i to node j. Let $x(P)$ denote the traffic associated with path pair P. The primary (working) path carrying traffic under normal (no-failure) conditions in P will be denoted by $W(P)$, and the backup path to which traffic is rerouted after failure by $B(P)$. The problem of two-phase routing with shared backup path restoration so as to maximize the network throughput can be formulated as the following path-flow-based linear program:

$$\text{maximize} \quad \sum_{i \in N} \alpha_i$$

subject to

$$\sum_{P \in \mathcal{PB}_{ij}} x(P) = \alpha_j R_i + \alpha_i C_j \qquad \forall\, i, j \in N \tag{1}$$

$$\sum_{i,j \in N} \sum_{P \in \mathcal{PB}_{ij}, W(P) \ni e} x(P) \le u_e \qquad \forall\, e \in E \tag{2}$$

$$\sum_{i,j \in N} \sum_{P \in \mathcal{PB}_{ij}, W(P) \ni e, W(P) \not\ni \tilde{f}} x(P) + \sum_{i,j \in N} \sum_{P \in \mathcal{PB}_{ij}, B(P) \ni e, W(P) \ni \tilde{f}} x(P) \le u_e \qquad \forall\, e, f \in E,\ \tilde{e} \ne \tilde{f} \tag{3}$$

$$\alpha_i \ge 0 \qquad \forall\, i \in N \tag{4}$$

$$x(P) \ge 0 \qquad \forall\, P \in \mathcal{PB}_{ij},\ \forall\, i, j \in N. \tag{5}$$

Constraints (1) correspond to the routing of $\alpha_j R_i + \alpha_i C_j$ amount of demand from node i to node j along primary paths that are protected by link-disjoint backup paths. Constraints (2) are the link capacity constraints under normal (no-failure) conditions. Constraints (3) state that after failure of any link $\tilde{f}$ (i.e., link f and its reverse $\hat{f}$), the sum of working traffic (not affected by the failure of link $\tilde{f}$) on a link and the restoration traffic that appears on the link after failure of link $\tilde{f}$ is at most the capacity of the link. These constraints model the two ways of sharing backup capacity described in Section III-A. Note that the (non)containment $\tilde{f} \in W(P)$ of failure links in primary paths for the summation in these constraints are in the *undirected sense*.

The objective function $\sum_{i \in N} \alpha_i$ can be interpreted as the throughput as follows. Suppose we relax the requirement that the traffic split ratios α_i sum to 1 in a feasible solution of the

problem. Consider the sum $\lambda = \sum_{i \in N} \alpha_i$. The traffic split ratios can be divided by λ (normalized) so that they sum to 1, in which case all matrices in $\lambda \cdot \mathcal{T}(\vec{R}, \vec{C})$ can be feasibly routed. Thus, the appropriate measure of throughput is the quantity $\sum_{i \in N} \alpha_i$ *when the traffic split ratios are not constrained to sum to* 1.

D. Dual of Path-Flow-Based LP Formulation

The dual formulation of the above linear program associates a variable π_{ij} with each demand constraint in (1), a nonnegative variable $z(e)$ with each link capacity constraint in (2), and a nonnegative variable $w(e, \tilde{f})$ with each link capacity constraint in (3). The dual program can be written as

$$\text{minimize} \quad \sum_{e \in E} u_e z(e) + \sum_{e \in E} \sum_{\tilde{f} \in \tilde{E}, \tilde{e} \neq \tilde{f}} u_e w(e, \tilde{f})$$

subject to

$$\sum_{e \in W(P)} z(e) + \sum_{e \in W(P), \tilde{f} \notin W(P)} w(e, \tilde{f}) + \sum_{e \in B(P), \tilde{f} \in W(P)} w(e, \tilde{f}) \geq \pi_{ij} \qquad \forall P \in \mathcal{PB}_{ij}, \forall i, j \in N \tag{6}$$

$$\sum_{i \in N, i \neq k} R_i \pi_{ik} + \sum_{j \in N, \neq k} C_j \pi_{kj} \geq 1 \qquad \forall k \in N \tag{7}$$

$$z(e), w(e, \tilde{f}) \geq 0 \qquad \forall e, f \in E, \tilde{e} \neq \tilde{f}. \tag{8}$$

Because of the nature of constraints (7), we can assume that the variables π_{ij} attain the maximum possible value given by constraints (6) in any optimal solution. Then, we have

$$\pi_{ij} = \min_{P \in \mathcal{P}_{ij}} \left[\sum_{e \in W(P)} z(e) + \sum_{e \in W(P), \tilde{f} \notin W(P)} w(e, \tilde{f}) + \sum_{e \in B(P), \tilde{f} \in W(P)} w(e, \tilde{f}) \right] \qquad \forall i, j \in N.$$

This allows us to eliminate the dual variables π_{ij}. Let $\Psi(z, w, P)$ denote the quantity on the left-hand side (LHS) of (6) [or, inside the min on the right-hand side (RHS) of the above equation] for a primary–backup path pair P and given weights $z(e), w(e, \tilde{f})$. That is

$$\Psi(z, w, P) = \sum_{e \in W(P)} z(e) + \sum_{e \in W(P), \tilde{f} \notin W(P)} w(e, \tilde{f}) + \sum_{e \in B(P), \tilde{f} \in W(P)} w(e, \tilde{f}). \tag{9}$$

After removal of constraints (6), the dual problem can be written as

$$\text{minimize} \quad \sum_{e \in E} u_e z(e) + \sum_{e \in E} \sum_{\tilde{f} \in \tilde{E}, \tilde{e} \neq \tilde{f}} u_e w(e, \tilde{f})$$

subject to

$$\sum_{i \in N, i \neq k} R_i \min_{P \in \mathcal{PB}_{ik}} \Psi(z, w, P) + \sum_{j \in N, j \neq k} C_j \min_{P \in \mathcal{PB}_{kj}} \Psi(z, w, P) \geq 1 \qquad \forall k \in N \tag{10}$$

$$z(e), w(e, \tilde{f}) \geq 0 \qquad \forall e, f \in E, \tilde{e} \neq \tilde{f}. \tag{11}$$

In order to check feasibility for the dual problem for given set of weights $z(e), w(e, \tilde{f})$, we need to compute $\min_{P \in \mathcal{PB}_{ij}} \Psi(z, w, P)$ for all node pairs $i, j \in N$ and verify inequality (10) for all $k \in N$. For given $i, j \in N$ and weights $z(e), w(e, \tilde{f})$, let SBPR-DISJOINT-PATHS denote the problem of computing a link-disjoint primary backup path pair from node i to node j that minimizes $\Psi(z, w, P)$. We show next that this problem is strongly NP-hard and not approximable within any constant factor even in pseudopolynomial time unless P = NP.

E. Hardness of SBPR-DISJOINT-PATHS Problem for Checking Dual Feasibility

To show that the problem SBPR-DISJOINT-PATHS is NP-hard, we exhibit a reduction from another disjoint paths problem that we call SIMPLE-DISJOINT-PATHS.

> Problem SIMPLE-DISJOINT-PATHS: *Given a graph $G = (V, E)$ with two sets of nonnegative link costs a_e and b_e for all $e \in E$ and source, destination nodes $i, j \in V$, find a minimum cost link-disjoint path pair from node i to node j where the link costs on one of the paths is a_e and on the other path is b_e.*

This problem is known to be strongly NP-hard and not approximable within any constant factor even in pseudopolynomial time unless P = NP [17]. Without any loss of generality, we can assume that one of the path pairs in this problem is designated as primary and has link costs a_e, while the other path is designated as backup and has link costs b_e.

Theorem 3.1: The problem SBPR-DISJOINT-PATHS is strongly NP-hard and not approximable within any constant factor even in pseudopolynomial time unless P = NP.

Proof: We construct an approximation preserving reduction from the NP-hard problem SIMPLE-DISJOINT-PATHS. Consider an instance of this problem on a graph $G = (V, E)$ with link costs a_e and b_e and source, destination nodes $s, t \in V$. We construct a graph $G' = (V', E')$ by adding a few nodes and links to G as follows: $V' = V \cup \{v_1, v_2, v_3, v_4\}$ and $E' = E \cup \{e_1 = (v_1, s), e_2 = (v_1, v_2), e_3 = (v_2, s), e_4 = (v_3, v_4)\}$. This is illustrated in Fig. 4.

Now consider an instance of the SBPR-DISJOINT-PATHS problem on graph G' with source, destination nodes v_1, t and weights $z(e)$ and $w(e, f)$ as follows.

- $z(e) = 0$ for all $e \in E'$.
- $w(e_2, \tilde{e_4})$ is sufficiently large.
- $w(e, \tilde{e_4}) = a_e$ for all $e \in E$.
- $w(e, \tilde{e_1}) = b_e$ for all $e \in E$.
- All other $w(e, \tilde{f})$ values for $e, f \in E', \tilde{e} \neq \tilde{f}$ are zero.

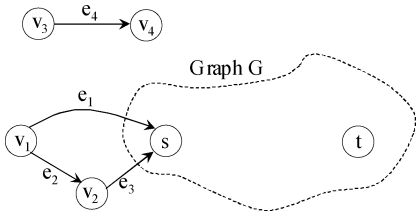

Fig. 4. Reduction to show NP-hardness of SBPR-DISJOINT-PATHS.

The problem SBPR-DISJOINT-PATHS consists of computing a primary–backup path pair P from node v_1 to node t so as to minimize $\Psi(z, w, P)$ as given by (9). Since e_4 is an isolated link, it does not appear on any of the paths from v_1 to t. Since there are exactly two links emanating out of node v_1 in G', one of these must be on the primary path and the other on the backup path. If link e_2 is on the primary path in an optimal primary–backup path pair P for SBPR-DISJOINT-PATHS, then the weight $w(e_2, \tilde{e_4})$ will appear in $\Psi(w, z, P)$ and lead to a large value. Thus, we can assume that link e_1 appears on the primary path.

Given a link-disjoint primary–backup path pair (P, B) from s to t in G, these naturally map to link-disjoint primary–backup path pairs (P', B') from v_1 to t in G' as follows: $P' = \{e_1\} \cup P$ and $B' = \{e_2, e_3\} \cup B$. The inverse mapping is also obvious.

Because of the choice of the weights $z(e)$ and $w(e, f)$ in G', we can simplify the expression for $\Psi(z, w, (P', B'))$ on the RHS of (9) as follows:

$$\sum_{e \in P'} z(e) = 0$$
$$\sum_{e \in P', \tilde{f} \notin P'} w(e, \tilde{f}) = \sum_{e \in P'} w(e, \tilde{e_4}) = \sum_{e \in P} a_e$$
$$\sum_{e \in B', \tilde{f} \in P'} w(e, \tilde{f}) = \sum_{e \in B'} w(e, \tilde{e_1}) = \sum_{e \in B} b_e.$$

Adding the above three equations, we have

$$\sum_{e \in P'} z(e) + \sum_{e \in P', \tilde{f} \notin P'} w(e, \tilde{f}) + \sum_{e \in B', \tilde{f} \in P'} w(e, \tilde{f}) = \sum_{e \in P} a_e + \sum_{e \in B} b_e. \quad (12)$$

Thus, minimizing the RHS of (12) in an instance of SIMPLE-DISJOINT-PATHS is equivalent to minimizing the LHS of (12) in the above defined instance of SBPR-DISJOINT-PATHS. This completes the approximation preserving reduction and proves the theorem. ■

F. Hardness of Two-Phase Routing With Shared Backup Path Restoration

In this section, we use the NP-hardness of the SBPR-DISJOINT-PATHS problem to show that the problem of maximum throughput two-phase routing with shared backup path restoration is NP-hard. We use the equivalence of polynomial-time optimization and polynomial-time separation for linear programming established by Grötschel *et al.* [9]. The problem of separation consists of determining whether a given set of values for the variables of the linear program is feasible and, if not, identifying at least one constraint that is violated. Such a procedure is also called a *separation oracle* for the linear program.

Theorem 3.2: The problem of computing the maximum throughput for two-phase routing with shared backup path restoration under given link capacities and ingress–egress traffic capacities is NP-hard.

Proof: The problem SBPR-DISJOINT-PATHS consists of computing a link-disjoint primary–backup path pair P from node i to node j so as to minimize $\Psi(z, w, P)$ as given by (9). Since $\Psi(z, w, P)$ is linear in the variables $z(e), w(e, \tilde{f})$, this problem can be expressed as the following linear program:

$$\text{maximize} \quad x$$

subject to

$$\Psi(z, w, P) \geq x \qquad \forall\, P \in \mathcal{PB}_{ij} \quad (13)$$
$$z(e), w(e, \tilde{f}) \geq 0 \qquad \forall\, e, f \in E, \tilde{e} \neq \tilde{f}. \quad (14)$$

Because this problem was shown to be NP-hard (Theorem 3.1) and because of the equivalence of polynomial-time optimization and polynomial-time separation for linear programming, it follows that the problem of separation for the constraints (13) is NP-hard.

Now consider the first dual linear program in Section III-D for the problem of maximum throughput two-phase routing with shared backup path restoration. In this linear program, the constraints (6) for each $i, j \in N$ are identical to constraints (13) in the linear program for SBPR-DISJOINT-PATHS [since the LHS of constraints (6) is simply $\Psi(z, w, P)$]. Hence, the separation problem for constraints (6) is NP-hard. [The constraints (7) can easily be verified in polynomial time.] Again using the equivalence of polynomial-time optimization and polynomial-time separation for linear programming, it follows that the dual and hence the primal optimization problem is NP-hard. This completes the proof. ■

We will show that having access to an approximation oracle for the SBPR-DISJOINT-PATHS problem that outputs a solution within $(1 + \xi)$-factor of the optimum value allows us to compute a solution for two-phase routing with shared backup path restoration whose throughput is guaranteed to be arbitrarily close to $(1 + \xi)$-factor of the optimum. The combinatorial algorithm we develop makes a polynomial number of calls to the oracle for SBPR-DISJOINT-PATHS (the rest of the computation is also polynomial-time).

Before we proceed with the development of this algorithm, we give a heuristic for the SBPR-DISJOINT-PATHS problem. By using the combinatorial algorithm together with this heuristic as the oracle for SBPR-DISJOINT-PATHS, we obtained solutions within 5% of optimality for maximum throughput two-phase routing with shared backup path restoration, as reported in Section IV.

G. Heuristics for SBPR-DISJOINT-PATHS Problem

Since the problem SBPR-DISJOINT-PATHS is strongly NP-hard and not approximable within any constant factor even

in pseudopolynomial time, we must depend on heuristics to obtain near-optimal solutions. The approach we consider is to generate candidate primary paths in a greedy manner, then compute the best backup path for each such primary path, and choose the least cost solution. By generating more candidate primary paths, the obtained solution can be made closer to the optimal one.

In an optimal solution, it is reasonable to expect that the primary path will have a small number of hops. Thus, we can approximate the sum $\sum_{e\in W(P),\tilde{f}\notin W(P)} w(e,\tilde{f})$ by adding terms for links $\tilde{f}$ that are in $W(P)$. This sum then becomes $\sum_{e\in W(P)}\sum_{\tilde{f}\neq\tilde{e}} w(e,\tilde{f})$. Now define link costs

$$c(e) = z(e) + \sum_{\tilde{f}\neq\tilde{e}} w(e,\tilde{f}) \qquad \forall\, e \in E.$$

Then, we have

$$\sum_{e\in W(P)} z(e) + \sum_{e\in W(P)}\sum_{\tilde{f}\neq\tilde{e}} w(e,\tilde{f}) = \sum_{e\in W(P)} c(e).$$

Using the link costs $c(e)$, we generate a suitably large number of candidate primary paths in increasing order of cost, using a K-shortest paths algorithm [16].

The SBPR-DISJOINT-PATHS problem is easy if the primary path is known. Given a candidate primary path $P_1 = W(P)$, we need to generate a link-disjoint backup path $B_1 = B(P)$ that minimizes $\sum_{e\in B(P),\tilde{f}\in W(P)} w(e,\tilde{f})$. This also involves a shortest path computation as follows. We set the costs of each link (and its reverse) in P_1 to infinity to model the link diversity requirement of the backup path. The costs of the other links are $d(e) = \sum_{\tilde{f}\in W(P)} w(e,\tilde{f})$. This gives

$$\sum_{e\in B(P),\tilde{f}\in W(P)} w(e,\tilde{f}) = \sum_{e\in B(P)} d(e).$$

Thus, a shortest path computation with link costs $d(e)$ will give the best link-disjoint backup path B_1 for the given primary path P_1.

We choose the primary–backup path pair that gives the lowest cost solution to SBPR-DISJOINT-PATHS. By generating more candidate primary paths, we can attempt to improve the cost of the solution. By choosing a K-shortest paths algorithm that generates the paths in increasing order of cost through *incremental* shortest path computations [16], more candidate primary paths can be generated in an efficient manner without starting the procedure from the beginning.

H. Combinatorial Algorithm Using Oracle for SBPR-DISJOINT-PATHS

In this section, we use the primal and dual formulations of the path-flow-based linear program to develop a combinatorial algorithm for the problem. We assume that there is an oracle that outputs a solution to the SBPR-DISJOINT-PATHS problem that is guaranteed to be within $(1+\xi)$-factor of the optimum value and runs in time $T_{\mathrm{DP}}(n,m)$ on a graph with n nodes and m links.

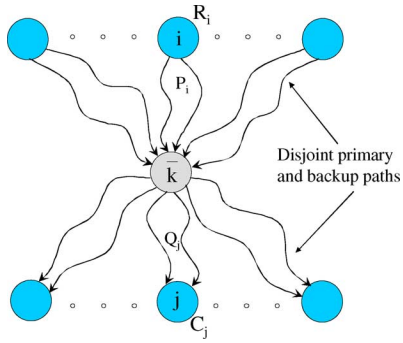

Fig. 5. One step in the primal-dual computation for shared backup path restoration in two-phase routing.

For a given node k and weights $z(e), w(e,\tilde{f})$, let $V(k)$ denote the LHS of constraint (10). The combinatorial algorithm we develop is iterative and needs to compute $\min_{k\in N} V(k)$ *approximately* in every iteration. This can be done as follows. Given the weights $z(e)$ and $w(e,f)$, we compute $\min_{P\in\mathcal{PB}_{ij}} \Psi(z,w,P)$ for each $i,j \in N$ within $(1+\xi)$-factor of its actual value using the approximation oracle for SBPR-DISJOINT-PATHS. We then use these values to compute $V(k)$ and then take the minimum over all $k \in N$. This gives us $\min_{k\in N} V(k)$ within $(1+\xi)$-factor of its actual value.

The overall algorithm, called Algorithm SBPR, works as follows. Start with initial weights $z(e) = \frac{\delta}{u_e}\ \forall\, e \in E$ and $w(e,\tilde{f}) = \frac{\delta}{u_e}\ \forall\, e,f \in E, \tilde{e} \neq \tilde{f}$ (the quantity δ depends on ϵ and is derived later). Repeat the following until the dual objective function value is greater than or equal to 1.

1) Compute node $k = \bar{k}$ for which $V(k)$ is minimum as described above. This identifies a node $\bar{k}$ as well as primary–backup path pairs P_i from node i to node $\bar{k}$ for all $i \neq \bar{k}$ and primary–backup path pairs Q_j from node $\bar{k}$ to node j for all $j \neq \bar{k}$. (These are the link-disjoint primary–backup path pairs between respective nodes obtained from the oracle for SBPR-DISJOINT-PATHS.) This is illustrated in Fig. 5.
2) For a traffic split ratio of 1 for intermediate node $\bar{k}$, the traffic on path P_i is R_i for all $i \neq \bar{k}$ and the traffic on path Q_j is C_j for all $j \neq \bar{k}$. Using this, compute the working traffic $s(e)$ under normal (no-failure) conditions on link e per unit split ratio $\alpha_{\bar{k}}$ for intermediate node $\bar{k}$ as

$$s(e) = \sum_{i\neq\bar{k},W(P_i)\ni e} R_i + \sum_{j\neq\bar{k},W(Q_j)\ni e} C_j \qquad \forall\, e \in E. \quad (15)$$

3) For the above working traffic and primary–backup path pairs P_i, Q_j, compute the traffic $s'(e,\tilde{f})$ on link e after failure of link $\tilde{f}$. The quantity $s'(e,\tilde{f})$ is the sum of working traffic on link e that is not affected by the failure of link $\tilde{f}$ and the restoration traffic that appears on the link after failure of link $\tilde{f}$ and is computed as

$$s'(e,\tilde{f}) = \sum_{i\neq\bar{k},W(P_i)\ni e,W(P_i)\not\ni\tilde{f}} R_i + \sum_{j\neq\bar{k},W(Q_j)\ni e,W(Q_j)\not\ni\tilde{f}} C_j + \sum_{i\neq\bar{k},B(P_i)\ni e,W(P_i)\ni\tilde{f}} R_i + \sum_{j\neq\bar{k},B(Q_j)\ni e,W(Q_j)\ni\tilde{f}} C_j \quad \forall\, e,f \in E, \tilde{e} \neq \tilde{f}. \quad (16)$$

The maximum possible traffic on link e is thus

$$\max\left(s(e), \max_{\tilde{f}\neq\tilde{e}} s'(e,\tilde{f})\right) \qquad \forall\, e \in E.$$

4) Compute the maximum value α for the traffic split ratio for intermediate $\bar{k}$ that does not lead to violation of (original) link capacity constraints for the above traffic as

$$\alpha = \min_{e\in E} \frac{u_e}{\max\left(s(e), \max_{\tilde{f}\neq\tilde{e}} s'(e,\tilde{f})\right)}. \tag{17}$$

5) For this value α of the traffic split ratio for intermediate node $\bar{k}$, send αR_i amount of traffic from node i to node $\bar{k}$ along primary–backup path pair P_i for all $i \neq \bar{k}$ and αC_j amount of traffic from node $\bar{k}$ to node j along primary–backup path pair Q_j for all $j \neq \bar{k}$. Compute the working traffic $\Delta(e)$ on link e under normal (no-failure) conditions as

$$\Delta(e) = \alpha s(e) \qquad \forall\, e \in E$$

and the traffic $\Delta'(e,\tilde{f})$ on link e after failure of any other link f as

$$\Delta'(e,\tilde{f}) = \alpha s'(e,\tilde{f}) \qquad \forall\, e,f \in E, \tilde{e}\neq\tilde{f}.$$

6) Update the weights $z(e)$ and $w(e,\tilde{f})$ as follows:

$$z(e) \leftarrow z(e)\left(1+\frac{\epsilon\Delta(e)}{u_e}\right) \qquad \forall\, e \in E$$

$$w(e,\tilde{f}) \leftarrow w(e,\tilde{f})\left(1+\frac{\epsilon\Delta'(e,\tilde{f})}{u_e}\right) \qquad \forall\, e,f \in E, \tilde{e}\neq\tilde{f}.$$

7) Increment the split ratio $\alpha_{\bar{k}}$ associated with node $\bar{k}$ by α.

When the above procedure terminates, primal capacity constraints will be violated since we were working with the original (and not residual) link capacities at each stage. To remedy this, we scale down the traffic and split ratios α_i uniformly so that capacity constraints are obeyed.

Note that since the algorithm maintains primal and dual solutions at each step, the optimality gap can be estimated by computing the ratio of the primal and dual objective function values. The computation can be terminated immediately after the desired closeness to optimality is achieved.

The pseudocode for the above procedure, called Algorithm SBPR, is provided. Arrays $\mathrm{work}(e)$ and $\mathrm{fail}(e,\tilde{f})$ keep track, respectively, of the working traffic on link e under normal (no-failure) conditions and the traffic on link e after failure of any other link $\tilde{f}$. The variable D is initialized to 0 and remains less than 1 as long as the dual objective function value is less than 1. After the **while** loop terminates, the factor by which the capacity constraint on each link e gets violated is computed into array $\mathrm{scale}(e)$. Finally, the α_i values are divided by the maximum capacity violation factor and the resulting values are output.

We next show that this combinatorial algorithm provides an approximation guarantee within $(1+\xi+\epsilon)$-factor of the optimum for any given $\epsilon > 0$. (Recall that $(1+\xi)$ is the approximation factor guaranteed by the oracle for the SBPR-DISJOINT-PATHS problem.)

Theorem 3.3: For any given $0 < \epsilon' \leq 0.5(1+\xi)$, Algorithm SBPR computes a solution with objective function value within $(1+\xi+\epsilon')$-factor of the optimum for

$$\delta = \frac{1+\epsilon}{[(1+\epsilon)\frac{m^2}{2}]^{1/\epsilon}} \quad \epsilon = \frac{\epsilon'}{2(1+\xi)}.$$

We show that the running time of Algorithm SBPR is a polynomial (in the network size and $\frac{1}{\epsilon}$) times the time $T_{\mathrm{DP}}(n,m)$ taken per call for the oracle for SBPR-DISJOINT-PATHS.

Algorithm SBPR

$\alpha_k \leftarrow 0 \qquad \forall\, k \in N;$
$z(e) \leftarrow \frac{\delta}{u_e} \qquad \forall\, e \in E;$
$w(e,\tilde{f}) \leftarrow \frac{\delta}{u_e} \qquad \forall\, e,f \in E, \tilde{e}\neq\tilde{f};$
$\mathrm{work}(e) \leftarrow 0 \qquad \forall\, e \in E;$
$\mathrm{fail}(e,\tilde{f}) \leftarrow 0 \qquad \forall\, e,f \in E, \tilde{e}\neq\tilde{f};$
$D \leftarrow 0;$

while $D < 1$ **do**
  For each $i,j \in N$, compute primary–backup path pair P from i to j that minimizes $\Psi(z,w,P)$ using oracle for SBPR-DISJOINT-PATHS;
  (Denote value of $\Psi(z,w,P)$ for computed path pair P from i to j by $DP(i,j)$ for all $i,j \in N$.)
  $V(k) \leftarrow \sum_{i\in N, i\neq k} R_i DP(i,k) + \sum_{j\in N, j\neq k} C_j DP(k,j) \quad \forall\, k \in N;$
  $\bar{k} \leftarrow \arg\min_{k\in N} V(k);$
  (Denote primary–backup path pair from i to $\bar{k}$ by P_i for all $i \neq \bar{k}$ and primary–backup path pair from $\bar{k}$ to j by Q_j for all $j \neq \bar{k}$.)
  $s(e) \leftarrow \sum_{i\neq\bar{k}, W(P_i)\ni e} R_i + \sum_{j\neq\bar{k}, W(Q_j)\ni e} C_j \quad \forall\, e \in E;$
  $s'(e,\tilde{f}) \leftarrow \sum_{i\neq\bar{k}, W(P_i)\ni e, W(P_i)\not\ni\tilde{f}} R_i + \sum_{j\neq\bar{k}, W(Q_j)\ni e, W(Q_j)\not\ni\tilde{f}} C_j + \sum_{i\neq\bar{k}, B(P_i)\ni e, W(P_i)\ni\tilde{f}} R_i + \sum_{j\neq\bar{k}, B(Q_j)\ni e, W(Q_j)\ni\tilde{f}} C_j \quad \forall\, e,f \in E, \tilde{e}\neq\tilde{f};$
  $\alpha \leftarrow \min_{e\in E} \frac{u_e}{\max(s(e), \max_{\tilde{f}\neq\tilde{e}} s'(e,\tilde{f}))};$
  $\Delta(e) \leftarrow \alpha s(e) \quad \forall\, e \in E;$
  $\Delta'(e,\tilde{f}) \leftarrow \alpha s'(e,\tilde{f}) \quad \forall\, e,f \in E, \tilde{e}\neq\tilde{f};$
  $\mathrm{work}(e) \leftarrow \mathrm{work}(e) + \Delta(e) \quad \forall\, e \in E;$
  $\mathrm{fail}(e,\tilde{f}) \leftarrow \mathrm{fail}(e,\tilde{f}) + \Delta'(e,\tilde{f}) \quad \forall\, e,f \in E, \tilde{e}\neq\tilde{f};$
  $z(e) \leftarrow z(e)\left(1+\frac{\epsilon\Delta(e)}{u_e}\right) \quad \forall\, e \in E;$
  $w(e,\tilde{f}) \leftarrow w(e,\tilde{f})\left(1+\frac{\epsilon\Delta'(e,\tilde{f})}{u_e}\right) \quad \forall\, e,f \in E, \tilde{e}\neq\tilde{f};$
  $\alpha_{\bar{k}} \leftarrow \alpha_{\bar{k}} + \alpha;$
  $D \leftarrow \sum_{e\in E} u_e z(e) + \sum_{e\in E}\sum_{\tilde{f}\in\tilde{E}, \tilde{f}\neq\tilde{e}} u_e w(e,\tilde{f});$
end while

$\mathrm{fail\_max}(e) \leftarrow \max_{\tilde{f}\neq\tilde{e}} \mathrm{fail}(e,\tilde{f}) \quad \forall\, e \in E;$
$\mathrm{scale}(e) \leftarrow \frac{\max(\mathrm{work}(e), \mathrm{fail\_max}(e))}{u_e} \quad \forall\, e \in E;$
$\mathrm{scale\_max} \leftarrow \max_{e\in E} \mathrm{scale}(e);$
$\alpha_k \leftarrow \frac{\alpha_k}{\mathrm{scale\_max}}$ for all $k \in N;$
Output traffic split ratios $\alpha_k;$

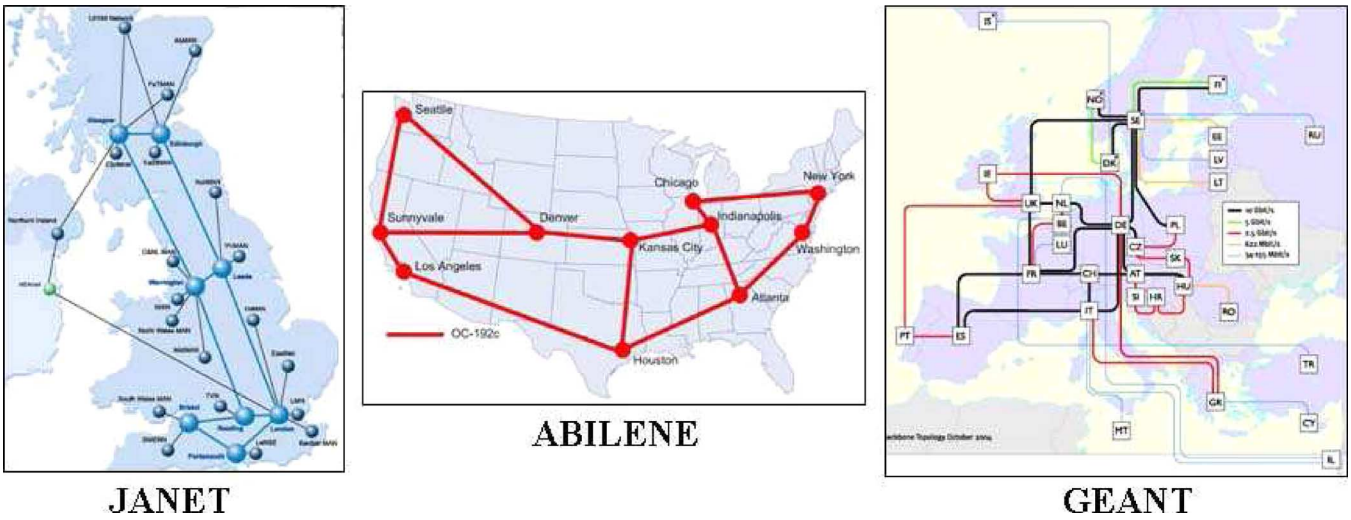

Fig. 6. Three research network topologies.

TABLE I
ROCKETFUEL TOPOLOGIES. THE ORIGINAL NUMBER OF ROUTERS AND INTERROUTER LINKS, AND THE NUMBER OF COALESCED POPS AND INTER-POP LINKS FOR EACH TOPOLOGY IS SHOWN

Topology	Routers (original)	Links (inter-router)	PoPs (coalesced)	Links (inter-PoP)
Telstra (Australia) 1221	108	306	57	59
Sprintlink (US) 1239	315	1944	44	83
Ebone (Europe) 1755	87	322	23	38
Tiscali (Europe) 3257	161	656	50	88
Exodus (Europe) 3967	79	294	22	37
Abovenet (US) 6461	141	748	22	42

Theorem 3.4: For any given $\epsilon > 0$ chosen to provide the desired approximation factor guarantee in accordance with Theorem 3.3, Algorithm SBPR runs in time

$$O\left(\frac{1}{\epsilon^2} n^2 m^2 T_{\mathrm{DP}}(n,m) \log m\right).$$

Proofs of the above theorems are provided in the Appendix.

IV. EVALUATION ON ISP TOPOLOGIES

In this section, we compare the throughput performance of three mechanisms for protecting against link failures—local restoration [15], K-route path restoration [15], and shared backup path restoration—to that of the unprotected case for two-phase routing. To compute the throughput for two-phase routing with link restoration and K-route path restoration, we use the fast combinatorial algorithms from [15]. For the throughput of the unprotected scheme, we use the fast combinatorial algorithm from [12].

Because two of the three restoration mechanisms do not have polynomial-size linear programming formulations for maximizing throughput for two-phase routing, we use the combinatorial algorithms in all cases in order to make fair comparisons. We run the algorithms so as to provide solutions up to 5% of optimality. For maximum throughput two-phase routing with shared backup path restoration, we obtained solutions within 5% of optimality by using the combinatorial algorithm together with the heuristic in Section III-G as the oracle for SBPR-DISJOINT-PATHS. The running times range from tens of seconds to a few minutes on a Pentium III 1-GHz 256-MB machine.

A. Topologies and Link/Ingress–Egress Capacities

For our experiments, we use six ISP topologies collected by Rocketfuel, an ISP topology mapping engine [19]. These topologies list multiple intra-Point of Presence (PoP) routers and/or multiple intracity PoPs as individual nodes. We coalesced PoPs into nodes corresponding to cities so that the topologies represent geographical PoP-to-PoP ISP topologies. Some data about the original Rocketfuel topologies and their coalesced versions are provided in Table I.

We also use three research network topologies, namely the U.K. research network JANET [22], the U.S. research backbone ABILENE [23], and the European research network GEANT [24]. These are shown in Fig. 6.

Link capacities, which are required to compute the maximum throughput, are not available for the Rocketfuel topologies. Rocketfuel computed OSPF/IS-IS link weights for the topologies so that shortest cost paths match observed routes. In order to deduce the link capacities from the weights, we assumed that the given link weights are the default setting for OSPF weights in Cisco routers, i.e., inversely proportional to the link capacities [3]. The link capacities obtained in this manner turned out to be symmetric, i.e., $u_{ij} = u_{ji}$ for all $(i,j) \in E$.

There is also no available information on the ingress–egress traffic capacities at each node for the Rocketfuel topologies. Because ISPs commonly engineer their PoPs to keep the ratio of add/drop and transit traffic approximately fixed, we assumed that the ingress–egress capacity at a node is proportional to the total capacity of network links incident at that node. We also assume that $R_i = C_i$ for all nodes i since network routers and switches have bidirectional ports (line cards), hence the ingress and egress capacities are equal. Thus, we have $R_i (= C_i) \propto \sum_{e \in E^+(i)} u_e$.

In contrast, link capacities are available for all three research network topologies. Ingress–egress capacities are available for the JANET topology only.

TABLE II
THROUGHPUT OF TWO-PHASE ROUTING WITH LOCAL RESTORATION (λ_{LR}), K-ROUTE PATH RESTORATION (λ_{KPR}), AND SHARED BACKUP PATH RESTORATION (λ_{SBPR}) COMPARED TO UNPROTECTED CASE (λ_{UNP})

Topology	$\frac{\lambda_{LR}}{\lambda_{UNP}}$	$\frac{\lambda_{KPR}}{\lambda_{UNP}}$	$\frac{\lambda_{SBPR}}{\lambda_{UNP}}$
Telstra (Australia) 1221	0.4422	0.4415	0.4554
Sprintlink (US) 1239	0.6214	0.5600	0.6182
Ebone (Europe) 1755	0.3855	0.3763	0.3894
Tiscali (Europe) 3257	0.6294	0.5328	0.6459
Exodus (Europe) 3967	0.5461	0.5390	0.5592
Abovenet (US) 6461	0.4319	0.4184	0.4492
JANET (8-node)	0.5029	0.4951	0.5048
ABILENE (11-node)	0.5017	0.4301	0.5003
GEANT (33-node)	0.5356	0.5236	0.5570

TABLE III
OVERHEAD OF LOCAL RESTORATION (O_{LR}), K-ROUTE PATH RESTORATION (O_{KPR}), AND SHARED BACKUP PATH RESTORATION (O_{SBPR}) COMPARED TO UNPROTECTED CASE FOR TWO-PHASE ROUTING

Topology	O_{LR}	O_{KPR}	O_{SBPR}
Telstra (Australia) 1221	55.78%	55.85%	54.46%
Sprintlink (US) 1239	37.86%	44.00%	38.18%
Ebone (Europe) 1755	61.45%	62.37%	61.06%
Tiscali (Europe) 3257	37.06%	46.72%	35.41%
Exodus (Europe) 3967	45.39%	46.10%	44.08%
Abovenet (US) 6461	56.81%	58.16%	55.08%
JANET (8-node)	49.71%	50.49%	49.52%
ABILENE (11-node)	49.83%	57.00%	49.97%
GEANT (33-node)	46.44%	47.64%	44.30%

TABLE IV
NUMBER OF INTERMEDIATE NODES IN TWO-PHASE ROUTING FOR UNPROTECTED CASE (N_{UNP}), AND WITH LOCAL RESTORATION (N_{LR}), K-ROUTE PATH RESTORATION (N_{KPR}), AND SHARED BACKUP PATH RESTORATION (N_{SBPR}).

Topology	N_{UNP}	N_{LR}	N_{KPR}	N_{SBPR}
Telstra (Australia) 1221	1	1	1	1
Sprintlink (US) 1239	5	6	4	7
Ebone (Europe) 1755	4	5	3	5
Tiscali (Europe) 3257	7	6	2	7
Exodus (Europe) 3967	3	7	3	9
Abovenet (US) 6461	7	6	1	6
JANET (8-node)	3	3	2	4
ABILENE (11-node)	2	2	3	2
GEANT (33-node)	3	4	3	4

The coalesced Rocketfuel topologies are not biconnected and hence do not allow diverse link detours for some links and link-disjoint paths between some source–destination pairs. Any graph that is not biconnected has one or more *bridge* links whose removal disconnects the graph into two connected components. We overcome this limited connectivity of the topologies by splitting bridge links into two diverse links, each of half the capacity as the original link. Because the original Rocketfuel topologies contained many parallel links that were coalesced, this bridge-splitting transformation preserves the essential ISP-like topological properties of the networks. Also, the throughput of unprotected routing remains unchanged as a result of this transformation.

B. Experiments and Results

We denote the throughput values for the unprotected and link failure protected versions of two-phase routing as follows: 1) λ_{UNP} for unprotected; 2) λ_{LR} for local restoration; 3) λ_{KPR} for K-route path restoration; and 4) λ_{SBPR} for shared backup path restoration. We are also interested in the number of intermediate nodes i with $\alpha_i > 0$, which we denote for the four cases by N_{UNP}, N_{LR}, N_{KPR}, and N_{SBPR}, respectively.

1) Throughput: In Table II, we list the throughput values for the three protection schemes with respect to that for unprotected two-phase routing for the six Rocketfuel topologies and three research network topologies.

The overhead of protecting against link failures can be measured by the percentage decrease in network throughput over that for the unprotected case. For local restoration, this is $O_{\mathrm{LR}} = \frac{\lambda_{\mathrm{UNP}} - \lambda_{\mathrm{LR}}}{\lambda_{\mathrm{UNP}}}$. For K-route path restoration, this is $O_{\mathrm{KPR}} = \frac{\lambda_{\mathrm{UNP}} - \lambda_{\mathrm{KPR}}}{\lambda_{\mathrm{UNP}}}$. For shared backup path restoration, this is $O_{\mathrm{SBPR}} = \frac{\lambda_{\mathrm{UNP}} - \lambda_{\mathrm{SBPR}}}{\lambda_{\mathrm{UNP}}}$. These values are listed in Table III.

For local restoration and shared backup path restoration, the overhead ranges from 35%–60% for all topologies. For K-route path restoration, the overhead ranges from 45%–60% for the topologies. All three overheads are relatively high because of the limited diversity available in these six topologies. The general trend for these topologies is that O_{LR} is comparable to O_{SBPR}. For some of the topologies, both of these are appreciably lower than O_{KPR} (while comparable in other cases). K-route path restoration is more constrained by the physical diversity of the network than the other two restoration mechanisms because of the following two reasons: 1) sharing of backup capacity increases with more link-disjoint paths between a given pair of nodes; and 2) backup capacity is shared only among paths within the same connection and not with paths belonging to other connections. Hence, we have the increased overhead of K-route path restoration.

2) Number of Intermediate Nodes: In Table IV, we list the number of intermediate nodes with nonzero traffic split ratios for the four cases for the six Rocketfuel topologies and three research network topologies. In our experiments for the three restoration schemes, we observed that some intermediate nodes have quite small α_i values: They carry less than a percentage point of total network traffic. In practice, the traffic split ratios associated with these intermediate nodes can be redistributed to other nodes without any significant decrease in throughput. Hence, in Table IV, for the N_{LR}, N_{KPR}, and N_{SBPR} values, we list the number of intermediate nodes with the largest α_i values (normalized) that sum to at least 0.95, i.e., the nodes with largest traffic split ratios that together carry at least 95% of total network traffic. Similar to that for the unprotected case, the number of intermediate nodes with each restoration mechanism is a small fraction of the total number of nodes.

The number of intermediate nodes for local restoration and shared backup path restoration is about the same as that for the unprotected case for all the topologies. However, we observe a marked decrease in the number of intermediate nodes for K-route path restoration. This is again because of the limited diversity available in these topologies. For K-route path restoration, the algorithm intelligently selects only those intermediate nodes that have sufficient path diversity to many other nodes in the network since sharing of backup bandwidth is only across paths belonging to the same connection and can be increased only by having more mutually diverse paths between a given pair of nodes. In contrast, local restoration involves the restoration of working traffic on a link in a *local manner*: It is

much less constrained by global diversity in the selection of intermediate nodes. Similarly, for shared backup path restoration, higher levels of diversity are not required because every connection consists of exactly two mutually diverse paths. For both local restoration and shared backup path restoration, sharing of backup bandwidth occurs across different connections, hence more opportunities exist for such sharing.

V. Conclusion

The two-phase routing scheme was recently proposed for routing highly dynamic and changing traffic patterns with bandwidth guarantees and in a traffic-oblivious manner. It allows preconfiguration of the network such that all traffic patterns, permissible within the network's natural ingress–egress capacity constraints, can be handled in a capacity-efficient manner *without the necessity to detect any traffic changes in real time*. For the Rocketfuel topologies, the throughput of two-phase routing is within 6% of the optimal scheme among the class of all schemes that are allowed to reconfigure the network in response to traffic changes [12].

In this paper, we extended the two-phase routing scheme by providing resiliency against link failures through shared backup path restoration. We developed combinatorial algorithms for determining optimal traffic split ratios and routing along primary and backup paths for two-phase routing under this restoration mechanism. We view this as important progress for two-phase routing toward achieving carrier-class reliability so as to facilitate its future deployment in ISP networks.

We evaluated the throughput performance and number of intermediate nodes in two-phase routing with shared backup path restoration and compared it to that of two other restoration mechanisms and the unprotected case. For this, we used actual ISP topologies collected for the Rocketfuel project [19] and three research network topologies.

The handling of node failures in two-phase routing poses additional challenges. Failure of nonintermediate nodes lying on Phase 1 or Phase 2 paths can be restored by using node-disjoint primary and backup paths in path restoration. The failure of intermediate nodes is naturally accommodated in two-phase routing by redistributing traffic split ratios to other intermediate nodes, as proposed in [13]. Because a single-node failure can lead to both of the above scenarios, the corresponding mechanisms can be integrated.

Appendix
Analysis of Approximation Guarantee

Given a set of dual weights $z(e)$ and $w(e,\tilde{f})$, let $D(z,w)$ denote the dual objective function value, and let $\Gamma(z,w)$ denote the minimum value of the LHS of dual program constraint (10) over all nodes $k \in N$. Then, solving the dual program is equivalent to finding a set of weights $z(e)$ and $w(e,\tilde{f})$ such that $\frac{D(z,w)}{\Gamma(z,w)}$ is minimized. Denote the optimal objective function value of the latter by θ, i.e., $\theta = \min_{z,w} \frac{D(z,w)}{\Gamma(z,w)}$. Let z_{t-1} and w_{t-1} denote the respective weight functions at the beginning of iteration t of the **while** loop, and let A_{t-1} be the value of $\sum_{j\in N} \alpha_j$ (primal objective function) up to the end of iteration $t-1$. Suppose the algorithm terminates after iteration L. The following lemma upper bounds the value of $D(z,w)$ at the end of every iteration.

Lemma 1.1: At the end of every iteration t, $1 \le t \le L$, of Algorithm SBPR, the following holds:

$$D(z_t, w_t) \le \frac{m^2\delta}{2} \prod_{j=1}^{t} \left[1 + \frac{\epsilon(1+\xi)}{\theta}(A_j - A_{j-1})\right].$$

Proof: Let $k = \bar{k} \in N$ be the node for which $V(k)$ is minimum, and let P_i, Q_j be the corresponding primary–backup path pairs (as defined earlier) along which traffic is sent during iteration t. [Note that $V(k)$ for all $k \in N$ is not computed exactly, but within $(1+\xi)$-factor of its actual value using the approximation oracle for SBPR-DISJOINT-PATHS.] Recall that the weights $z(e), w(e,\tilde{f})$ are updated as

$$z_t(e) \leftarrow z_{t-1}(e)\left(1 + \frac{\epsilon\Delta(e)}{u_e}\right) \qquad \forall\, e \in E$$

$$w_t(e,\tilde{f}) \leftarrow w_{t-1}(e,\tilde{f})\left(1 + \frac{\epsilon[\Delta(e) + \Delta'(e,\tilde{f})]}{u_e}\right) \qquad \forall\, e,f \in E, \tilde{e} \ne \tilde{f}$$

where $\Delta(e)$ is the total working traffic on link e under normal (no-failure) conditions, and $\Delta'(e,\tilde{f})$ is the total traffic on link e after failure of some other link f (both sent during iteration t). Using this, we have

$$\begin{aligned}
&D(z_t, w_t)\\
&= \sum_{e\in E} u_e z_t(e) + \sum_{e\in E}\sum_{\tilde{f}\in\tilde{E},\tilde{e}\ne\tilde{f}} u_e w_t(e,\tilde{f})\\
&= \sum_{e\in E} u_e z_{t-1}(e) + \epsilon\sum_{e\in E} z_{t-1}(e)\Delta(e)\\
&\quad + \sum_{e\in E}\sum_{\tilde{f}\in\tilde{E},\tilde{e}\ne\tilde{f}} u_e w_{t-1}(e,\tilde{f})\\
&\quad + \epsilon\sum_{e\in E}\sum_{\tilde{f}\in\tilde{E},\tilde{e}\ne\tilde{f}} w_{t-1}(e,\tilde{f})\Delta'(e,\tilde{f})\\
&= D(z_{t-1}, w_{t-1})\\
&\quad + \epsilon\sum_{e\in E} z_{t-1}(e)\left(\sum_{i\ne\bar{k}, W(P_i)\ni e} \alpha R_i + \sum_{j\ne\bar{k}, W(Q_j)\ni e} \alpha C_j\right)\\
&\quad + \epsilon\sum_{e\in E}\sum_{\tilde{f}\in\tilde{E},\tilde{e}\ne\tilde{f}} w_{t-1}(e,\tilde{f})\\
&\quad \times \left(\sum_{i\ne\bar{k}, W(P_i)\ni e, W(P_i)\not\ni\tilde{f}} \alpha R_i + \sum_{j\ne\bar{k}, W(Q_j)\ni e, W(Q_j)\not\ni\tilde{f}} \alpha C_j\right.\\
&\quad \left. + \sum_{i\ne\bar{k}, B(P_i)\ni e, W(P_i)\ni\tilde{f}} \alpha R_i + \sum_{j\ne\bar{k}, B(Q_j)\ni e, W(Q_j)\ni\tilde{f}} \alpha C_j\right).
\end{aligned}$$

Interchanging the summations on the RHS of the previous equation and first summing along paths P_i, Q_j, and then over

i, j, respectively, we can rewrite the RHS of the previous equation to obtain

$$\begin{aligned}D(z_t,w_t) &= D(z_{t-1},w_{t-1}) + \epsilon\alpha\left[\sum_{i\neq\bar{k}} R_i \sum_{e\in W(P_i)} z_{t-1}(e) + \sum_{j\neq\bar{k}} C_j \sum_{e\in W(Q_j)} z_{t-1}(e)\right] \\ &\quad + \epsilon\alpha\left[\sum_{i\neq\bar{k}} R_i \sum_{e\in W(P_i),\tilde{f}\notin W(P_i)} w_{t-1}(e,\tilde{f}) + \sum_{j\neq\bar{k}} C_j \sum_{e\in W(Q_j),\tilde{f}\notin W(Q_j)} w_{t-1}(e,\tilde{f}) \right. \\ &\quad \left. + \sum_{i\neq\bar{k}} R_i \sum_{e\in B(P_i),\tilde{f}\in W(P_i)} w_{t-1}(e,\tilde{f}) + \sum_{j\neq\bar{k}} C_j \sum_{e\in B(Q_j),\tilde{f}\in W(Q_j)} w_{t-1}(e,\tilde{f})\right] \\ &= D(z_{t-1},w_{t-1}) \\ &\quad + \epsilon\alpha\left[\sum_{i\neq\bar{k}} R_i\Psi(z_{t-1},w_{t-1},P_i) + \sum_{j\neq\bar{k}} C_j\Psi(z_{t-1},w_{t-1},Q_j)\right]. \end{aligned} \tag{18}$$

Recall that we obtained the primary–backup path pairs P_i, Q_j by using the approximation oracle for SBPR-DISJOINT-PATHS. This oracle computes $\min_P \Psi(z,w,P)$ over all primary–backup path pairs P between a given pair of nodes within $(1+\xi)$-factor of its actual value. We then used these values to compute $V(k)$ and then took the minimum over all $k \in N$ to identify node $\bar{k}$. The value of $V(\bar{k})$ thus obtained must be within $(1+\xi)$-factor of $\Gamma(z_{t-1}, w_{t-1})$. Hence, we have

$$\sum_{i\neq\bar{k}} R_i\Psi(z_{t-1},w_{t-1},P_i) + \sum_{j\neq\bar{k}} C_j\Psi(z_{t-1},w_{t-1},Q_j) \leq (1+\xi)\Gamma(z_{t-1},w_{t-1}).$$

Using this in (18), we have

$$\begin{aligned}D(z_t,w_t) &\leq D(z_{t-1},w_{t-1}) + \epsilon\alpha(1+\xi)\Gamma(z_{t-1},w_{t-1}) \\ &= D(z_{t-1},w_{t-1}) + \epsilon(1+\xi)(A_t - A_{t-1})\Gamma(z_{t-1},w_{t-1}).\end{aligned}$$

The explanation for the step leading to (18) is as follows. Since the oracle for SBPR-DISJOINT-PATHS computes a solution within $(1+\xi)$-factor of the optimal cost, the values $\Psi(z_{t-1},w_{t-1},P_i)$ and $\Psi(z_{t-1},w_{t-1},Q_j)$ are within $(1+\xi)$-factor of $\min_{P\in\mathcal{PB}_{i\bar{k}}} \Psi(z_{t-1},w_{t-1},P)$ and $\min_{P\in\mathcal{PB}_{\bar{k}j}} \Psi(z_{t-1},w_{t-1},P)$, respectively. Using this for each iteration down to the first one, we have

$$D(z_t,w_t) \leq D(w_0) + \epsilon(1+\xi)\sum_{j=1}^{t}(A_j - A_{j-1})\Gamma(z_{j-1},w_{j-1}). \tag{19}$$

From the definition of θ, we have $\theta \leq \frac{D(z_{j-1},w_{j-1})}{\Gamma(z_{j-1},w_{j-1})}$, whence $\Gamma(z_{j-1},w_{j-1}) \leq \frac{1}{\theta}D(z_{j-1},w_{j-1})$. The number of weights $z(e)$ is m. Since a failure brings down both a link and its reverse, the number of possible failures is $\frac{m}{2}$. The number of weights $w(e,\tilde{f})$ is thus $m\left(\frac{m}{2}-1\right)$. Hence, $D(w_0) = \left(m + m\left(\frac{m}{2}-1\right)\right)\delta = \frac{m^2\delta}{2}$. Using these in (19), we have

$$D(z_t,w_t) \leq \frac{m^2\delta}{2} + \frac{\epsilon(1+\xi)}{\theta}\sum_{j=1}^{t}(A_j - A_{j-1})D(z_{j-1},w_{j-1}). \tag{20}$$

The property claimed in the lemma can now be proved using inequality (20) and mathematical induction on the iteration number t. ■

We now estimate the factor by which the objective function value value A_L in the primal solution needs to be scaled when the algorithm terminates so as to ensure that link capacity constraints are not violated.

Lemma 1.2: When Algorithm SBPR terminates, the primal solution needs to be scaled by a factor of at most $\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$ to ensure primal feasibility.

Proof: We need to show two things here after the primal solution is scaled by the above factor. First, the working traffic on every link e under normal (no-failure) conditions is at most the capacity of the link. Second, after failure of any link $\tilde{f}$, the sum of working traffic (not affected by the failure of link $\tilde{f}$) on each link e and the restoration traffic that appears on the link after failure of link $\tilde{f}$ is at most the capacity of the link.

First, consider any link e and associated weight $z(e)$. The value of $z(e)$ is updated by multiplying with the quantity $\left(1+\frac{\epsilon\Delta(e)}{u_e}\right)$ where $\Delta(e)$ is working traffic on this link under normal (no-failure) conditions corresponding to an iteration. Let the sequence of such traffic values $\Delta(e)$ associated with link e over all iterations be $\Delta_1, \Delta_2, \ldots, \Delta_L$. Let $\sum_{t=1}^{L}\Delta_t = \kappa u_e$, i.e., the total working traffic routed on link e exceeds its capacity by a factor of κ.

Because of the way in which α is chosen in accordance with (15)–(17), it follows that all dual weights are updated by a factor of at most $1+\epsilon$ after each iteration. Since the algorithm terminates when $D(z,w) \geq 1$, and since dual weights are updated by a factor of at most $1+\epsilon$ after each iteration, we have $D(z_L,w_L) < 1+\epsilon$. Since the weight $z(e)$, with coefficient u_e, is one of the summing components of $D(z,w)$, we have $u_e z_L(e) < 1+\epsilon$. Also, the value of $z_L(e)$ is given by

$$z_L(e,f) = \frac{\delta}{u_e}\prod_{t=1}^{L}\left(1+\frac{\Delta_t}{u_e}\epsilon\right).$$

Using the inequality $(1+cx) \geq (1+x)^c \ \ \forall\, x \geq 0$ and any $0 \leq c \leq 1$ and setting $x = \epsilon$ and $c = \frac{\Delta_t}{u_e} \leq 1$, we have

$$\begin{aligned}\frac{1+\epsilon}{u_e} > z_L(e) &\geq \frac{\delta}{u_e}\prod_{t=1}^{L}(1+\epsilon)^{\Delta_t/u_e} \\ &= \frac{\delta}{u_e}(1+\epsilon)^{\sum_{t=1}^{L}\Delta_t/u_e} \\ &= \frac{\delta}{u_e}(1+\epsilon)^{\kappa}\end{aligned}$$

whence

$$\kappa < \log_{1+\epsilon} \frac{1+\epsilon}{\delta}.$$

Second, consider any link e after failure of some other link $\tilde{f}$ and associated weight $w(e,\tilde{f})$. The value of $w(e,\tilde{f})$ is updated by multiplying with the quantity $(1+\frac{\epsilon\Delta'(e,\tilde{f})}{u_e})$ where $\Delta'(e,\tilde{f})$ is the traffic on this link after failure of link $\tilde{f}$ corresponding to an iteration. Let the sequence of such traffic values $\Delta'(e,\tilde{f})$ for a given link e and failure link $\tilde{f}$ over all iterations be $\Delta'_1,\Delta'_2,\ldots,\Delta'_L$. Let $\sum_{t=1}^{L}\Delta'_t = \kappa' u_e$, i.e., the total traffic routed on link e after failure of link $\tilde{f}$ exceeds its capacity by a factor of κ'.

As established in the first part of the proof, we have $D(z_L,w_L) < 1+\epsilon$. Since the weight $w(e,\tilde{f})$, with coefficient u_e, is one of the summing components of $D(z,w)$, we have $u_e w_L(e,\tilde{f}) < 1+\epsilon$. Also, the value of $w_L(e,\tilde{f})$ is given by

$$w_L(e,\tilde{f}) = \frac{\delta}{u_e}\prod_{t=1}^{L}\left(1+\frac{\Delta'_t}{u_e}\epsilon\right).$$

Using the same inequality as for the first argument, we have

$$\begin{aligned}\frac{1+\epsilon}{u_e} > w_L(e,\tilde{f}) &\geq \frac{\delta}{u_e}\prod_{t=1}^{L}(1+\epsilon)^{\Delta'_t/u_e}\\ &= \frac{\delta}{u_e}(1+\epsilon)^{\sum_{t=1}^{L}\Delta'_t/u_e}\\ &= \frac{\delta}{u_e}(1+\epsilon)^{\kappa'}\end{aligned}$$

whence

$$\kappa' < \log_{1+\epsilon}\frac{1+\epsilon}{\delta}.$$

■

Proof (of Theorem 3.3): Using Lemma 1.1 and the inequality $1+x \leq e^x$ for all $x \geq 0$, we have

$$\begin{aligned}D(z_t,w_t) &\leq \frac{m^2\delta}{2}\prod_{j=1}^{t} e^{\frac{\epsilon(1+\xi)}{\theta}(A_j - A_{j-1})}\\ &= \frac{m^2\delta}{2}e^{\epsilon(1+\xi)A_t/\theta}.\end{aligned}$$

The simplification in the above step uses telescopic cancellation of the sum $(A_j - A_{j-1})$ over j. Since the algorithm terminates after iteration L, we must have $D(z_L,w_L) \geq 1$. Thus

$$1 \leq D(z_L,w_L) \leq \frac{m^2\delta}{2}e^{\epsilon(1+\xi)A_L/\theta}$$

whence

$$\frac{\theta}{A_L} \leq \frac{\epsilon(1+\xi)}{\ln\frac{2}{m^2\delta}}. \tag{21}$$

From Lemma 1.2, the objective function value of the feasible primal solution after scaling is at least

$$\frac{A_L}{\log_{1+\epsilon}\frac{1+\epsilon}{\delta}}.$$

The approximation factor for the primal solution is at most the gap (ratio) between the dual and primal solutions. Using the lower bound on the primal solution and inequality (21), this is at most

$$\frac{\theta}{\frac{A_L}{\log_{1+\epsilon}\frac{1+\epsilon}{\delta}}} \leq \frac{\epsilon(1+\xi)\log_{1+\epsilon}\frac{1+\epsilon}{\delta}}{\ln\frac{2}{m^2\delta}} = \frac{\epsilon(1+\xi)}{\ln(1+\epsilon)}\frac{\ln\frac{1+\epsilon}{\delta}}{\ln\frac{2}{m^2\delta}}.$$

The quantity $\ln\frac{1+\epsilon}{\delta}/\ln\frac{2}{m^2\delta}$ equals $\frac{1}{1-\epsilon}$ for $\delta = (1+\epsilon)/\left[(1+\epsilon)\frac{m^2}{2}\right]^{1/\epsilon}$. Using this value of δ, the approximation factor is upper-bounded by $\frac{\epsilon(1+\xi)}{(1-\epsilon)\ln(1+\epsilon)}$. The quantity $\frac{\epsilon}{(1-\epsilon)\ln(1+\epsilon)}$ is at most $1+2\epsilon$ for $\epsilon \leq 0.25$. Setting $\epsilon = \frac{\epsilon'}{2(1+\xi)}$, we get the desired approximation ratio of $1+\epsilon'$. ■

Proof (of Theorem 3.4): We first consider the running time of each iteration of the algorithm during which node $\bar{k}$ and associated primary–backup paths pairs P_i, Q_j are chosen to send traffic. Computation of $\min_{P\in\mathcal{PB}_{ij}}\Psi(z,w,P)$ for all $i,j \in N$ requires $n(n-1)$ calls to the oracle for the SBPR-DISJOINT-PATHS problem. This takes a total of $n(n-1)T_{\mathrm{DP}}(n,m) = O(n^2 T_{\mathrm{DP}}(n,m))$ time. It can be verified that the time taken for all other computations is $O(n^2 m)$. This is subsumed by the time taken for the computation by the oracle since we can assume that $T_{\mathrm{DP}}(n,m) = \Omega(m)$. Thus, the running time per iteration is $O(n^2 T_{\mathrm{DP}}(n,m))$.

We next estimate the number of iterations before the algorithm terminates. Recall that in each iteration, traffic is sent along primary–backup path pairs P_i, Q_j corresponding to the maximum value of intermediate node split ratio α such that both the working traffic $\Delta(e)$ for link e under normal (no-failure) conditions and the traffic $\Delta'(e,\tilde{f})$ on link e after failure of some other link $\tilde{f}$ corresponding to that iteration are at most u_e.

Thus, for at least one link e, either the value $\Delta(e)$ or the value $\Delta'(e,\tilde{f})$ equals u_e and the weight $z(e)$ or $w(e,\tilde{f})$ respectively increases by a factor of $1+\epsilon$. Accordingly, with each iteration, we can associate a weight $z(e)$ or $w(e,\tilde{f})$ that increases by a factor of $1+\epsilon$.

Consider the weight $z(e)$ for fixed $e \in E$. Since $z_0(e) = \frac{\delta}{u_e}$ and $z_L(e) < \frac{1+\epsilon}{u_e}$ (as deduced in the proof of Lemma 1.2), the maximum number of times that this weight can be associated with any iteration is

$$\log_{1+\epsilon}\frac{1+\epsilon}{\delta} = \frac{1}{\epsilon}\left(1+\log_{1+\epsilon}\frac{m^2}{2}\right) = O\left(\frac{1}{\epsilon}\log_{1+\epsilon}\frac{m^2}{2}\right).$$

Using the same reasoning, it follows that the maximum number of times that the weight $w(e,\tilde{f})$ (for fixed $e,f \in E, \tilde{e} \neq \tilde{f}$) can be associated with any iteration is the same quantity.

Since there are a total of $m + m(\frac{m}{2}-1)$ weights $z(e)$ and $w(e,\tilde{f})$, hence the total number of iterations is upper-bounded by

$$O\left(\frac{1}{\epsilon}\left(m+m\left(\frac{m}{2}-1\right)\right)\log_{1+\epsilon}\frac{m^2}{2}\right) = O\left(\frac{1}{\epsilon}m^2\log_{1+\epsilon}\frac{m^2}{2}\right).$$

Multiplying this by the running time per iteration, we obtain the overall algorithm running time as

$$O\left(\frac{1}{\epsilon}n^2m^2T_{\mathrm{DP}}(n,m)\ \log_{1+\epsilon}\frac{m^2}{2}\right) = O\left(\frac{1}{\epsilon}n^2m^2T_{\mathrm{DP}}(n,m)\log_{1+\epsilon}m\right).$$

Since $\ln(1+\epsilon)=\Theta(\epsilon)$, this is $O\left(\frac{1}{\epsilon^2}n^2m^2T_{\mathrm{DP}}(n,m)\log m\right)$. ∎

REFERENCES

[1] D. Applegate, L. Breslau, and E. Cohen, "Coping with network failures: Routing strategies for optimal demand oblivious restoration," in *Proc. ACM SIGMETRICS/Performance*, Jun. 2004, pp. 270–281.

[2] D. Applegate and E. Cohen, "Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs," in *Proc. ACM SIGCOMM*, Aug. 2003, pp. 313–324.

[3] "Configuring OSPF," Cisco Systems, San Jose, CA, 2008 [Online]. Available: http://www.cisco.com/en/US/docs/ios/iproute_ospf/configuration/guide/iro_cfg.html

[4] B. T. Doshi, S. Dravida, P. Harshavardhana, O. Hauser, and Y. Wang, "Optical network design and restoration," *Bell Labs Tech. J.*, Jan.-Mar. 1999.

[5] N. G. Duffield, P. Goyal, A. G. Greenberg, P. P. Mishra, K. K. Ramakrishnan, and J. E. van der Merwe, "A flexible model for resource management in virtual private network," in *Proc. ACM SIGCOMM*, Aug. 1999, pp. 95–108.

[6] T. Erlebach and M. Rüegg, "Optimal bandwidth reservation in hose-model VPNs with multi-path routing," in *Proc. IEEE INFOCOM*, Mar. 2004, vol. 4, pp. 2275–2282.

[7] J. A. Fingerhut, S. Suri, and J. S. Turner, "Designing least-cost nonblocking broadband networks," *J. Algor.*, vol. 24, no. 2, pp. 287–309, 1997.

[8] N. Garg and J. Könemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," in *Proc. 39th FOCS*, 1998.

[9] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*. New York: Springer-Verlag, 1988.

[10] A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener, "Algorithms for provisioning virtual private networks in the hose model," in *Proc. ACM SIGCOMM*, Aug. 2001, pp. 135–146.

[11] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Efficient and robust routing of highly variable traffic," in *Proc. HotNets-III*, Nov. 2004.

[12] M. Kodialam, T. V. Lakshman, and S. Sengupta, "A versatile scheme for routing highly variable traffic in service overlays and IP backbones," in *Proc. IEEE INFOCOM*, Apr. 2006, pp. 1–12.

[13] M. Kodialam, T. V. Lakshman, J. B. Orlin, and S. Sengupta, "Preconfiguring IP-over-optical networks to handle router failures and unpredictable traffic," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 5, pp. 934–948, Jun. 2007.

[14] M. Kodialam, T. V. Lakshman, J. B. Orlin, and S. Sengupta, "Resilient routing of variable traffic with performance guarantees," in *Proc. IEEE ICNP*, Nov. 2009, pp. 213–222.

[15] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Throughput guaranteed restorable routing without traffic prediction," in *Proc. IEEE ICNP*, Nov. 2006, pp. 137–146.

[16] E. L. Lawler, "A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem," *Manage. Sci.*, vol. 18, pp. 401–405, 1972.

[17] C. Li, S. T. McCormick, and D. Simchi-Levi, "Finding disjoint paths with different path costs: Complexity and algorithms," *Networks*, vol. 22, pp. 653–667, 1992.

[18] S. Sengupta, "Efficient and robust routing of highly variable traffic," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., MIT, Cambridge, MA, 2005.

[19] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with Rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, Feb. 2004.

[20] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet indirection infrastructure," in *Proc. ACM SIGCOMM*, Aug. 2002, pp. 73–86.

[21] R. Zhang-Shen and N. McKeown, "Designing a predictable Internet backbone network," in *Proc. HotNets-III*, Nov. 2004.

[22] "JANET," Janet, Harwell Oxford, U.K. [Online]. Available: http://www.ja.net

[23] "ABILENE," Internet2, Ann Arbor, MI [Online]. Available: http://abilene.internet2.edu

[24] "GEANT," DANTE, Cambridge, U.K. [Online]. Available: http://www.geant.net

M. Kodialam (M'99–A'00) received the Ph.D. degree in operations research from the Massachusetts Institute of Technology, Cambridge, in 1991.

He has been with Bell Laboratories, Murray Hill, NJ, since October 1991. He currently is with the High Speed Networks Research Department, working on resource allocation and performance of communication systems including routing in MPLS systems, topology construction and routing in ad hoc wireless networks, and reliable routing in optical networks.

Dr. Kodialam is a member of INFORMS.

T. V. Lakshman (M'85–SM'98–F'05) received the Master's degree in physics from the Indian Institute of Science, Bangalore, in 1984, and the Ph.D degree in computer science from the University of Maryland, College Park, in 1986.

He is currently the Director of the Communication Protocols and Networking Research Department, Bell Laboratories, Murray Hill, NJ. His research interests and contributions span a spectrum of networking topics including switch architectures, network design, TCP performance, traffic management, video transmission over packet networks, and network security.

Dr. Lakshman is a Fellow of the Association for Computing Machinery (ACM). He was an Editor of the IEEE/ACM TRANSACTIONS ON NETWORKING from 1996 to 2002. He is currently an Editor of the IEEE TRANSACTIONS ON MOBILE COMPUTING. He has received several Best Paper Awards from the ACM and the IEEE.

James B. Orlin received the B.A. degree from the University of Pennsylvania, Philadelphia, in 1974, the Master's degrees from the California Institute of Technology, Pasadena, and the University of Waterloo, Waterloo, ON, Canada, in 1976, and the Ph.D. degree from Stanford University, Stanford, CA, in 1981.

He is the Edward Pennell Brooks Professor of Operations Research with the Massachusetts Institute of Technology (MIT) Sloan School of Management, Cambridge. He has served as a Fulbright Fellow to the Netherlands, as a National Science Foundation (NSF) Presidential Young Investigator, and as a co-director of the MIT Operations Research Center. With two colleagues, he has written a graduate-level text, *Network Flows: Theory, Algorithms, and Applications* (Prentice-Hall, 1993). This text was the winner of the 1993 Lanchester Prize for the best English language publication in Operations Research. He specializes in network and combinatorial optimization. He is also interested in applications of network optimization and combinatorial optimization to logistics and vehicle routing.

Sudipta Sengupta (M'01–SM'07) received the B.Tech. degree in computer science from the Indian Institute of Technology (IIT), Kanpur, India, and the M.S. and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge.

He is currently with Microsoft Research, Redmond, WA, where he is working on data center systems and networking, peer-to-peer applications, wireless networking, nonvolatile memory for cloud/server applications, and data deduplication. Previously, he spent five years at Bell Laboratories, the R&D Division of Lucent Technologies, Murray Hill, NJ, where he worked on Internet routing, optical switching, network security, wireless networks, and network coding. Before that, he was with Tellium, Oceanport, NJ, an optical networking pioneer that grew from an early-stage startup to a public company during his tenure there. He has published 65+ research papers in some of the top conferences, journals, and technical magazines. He has authored 40+ patents (granted or pending) in the area of computer systems and networking.

Dr. Sengupta was awarded the President of India Gold Medal at IIT-Kanpur for graduating at the top of his class across all disciplines. He won the IEEE Communications Society Leonard G. Abraham Prize for 2008 for his work on oblivious routing of Internet traffic. At Bell Labs, he received the President's Teamwork Achievement Award for technology transfer of research into Lucent products. His work on peer-to-peer based distribution of real-time layered video received the IEEE ICME 2009 Best Paper Award. At Microsoft, he received the Gold Star Award that recognizes "important career milestones of people leaders, thought leaders, and customer leaders as they take on roles to increase their contribution to Microsoft's long term success."