On the Feasibility and Efficacy of Protection Routing in IP Networks

Kin-Wah Kwong*, Lixin Gao†, Roch Guerin* and Zhi-Li Zhang‡
*University of Pennsylvania, †University of Massachusetts, ‡University of Minnesota
kkw@seas.upenn.edu, lgao@ecs.umass.edu, guerin@ee.upenn.edu, zhzhang@cs.umn.edu

***Abstract*—With network components increasingly reliable, routing is playing an ever greater role in determining network reliability. This has spurred much activity in improving routing stability and reaction to failures, and rekindled interest in centralized routing solutions, at least within a single routing domain. Centralizing decisions eliminates uncertainty and many inconsistencies, and offers added flexibility in computing routes that meet different criteria. However, it also introduces new challenges; especially in reacting to failures where centralization can increase latency. This paper leverages the flexibility afforded by centralized routing to address these challenges. Specifically, we explore when and how standby backup forwarding options can be activated, while waiting for an update from the centralized server after the failure of an individual component (link or node). We provide analytical insight into the feasibility of such backups as a function of network structure, and quantify their computational complexity. We also develop an efficient heuristic reconciling protectability and performance, and demonstrate its effectiveness in a broad range of scenarios. The results should facilitate deployments of centralized routing solutions.**

I. Introduction

Intra-domain routing in IP networks has traditionally relied on distributed computations among routers, with the concatenation of individual forwarding decisions eventually resulting in packet delivery. In spite of their inherent adaptability and scalability, distributed computations can make troubleshooting harder, because of the many sources of inconsistencies they allow. This has renewed interest in centralized routing solutions [6], [8], [22] for IP networks, at least in settings where scalability is less of a concern, *e.g.*, intra-domain routing. Centralizing decisions not only guarantees full visibility into the forwarding state of individual routers (now essentially cheap *forwarding engines* or FEs), it also affords added flexibility in computing paths that meet different requirements.

In spite of its advantages and even when scalability is not an issue, centralizing decisions has disadvantages. Of particular concern for reliability is latency in reacting to failures, *i.e.*, the central server needs to be notified, react to the failure, and communicate updated forwarding information to all affected FEs. This can result in non-negligible "gaps" after failures, during which FEs have no valid forwarding states for some destinations, and translate into substantial packet losses. A natural approach to the problem is through preventive mechanisms, *e.g.*, by having the central server pre-compute forwarding decisions for common (most) failure scenarios, *and* pre-load those in the FEs so that updated forwarding state is locally available. However, even such solutions have their limitations. For one, the sheer volume of alternate forwarding states across failure scenarios will likely require that it be stored in "slow" memory to keep costs low. As a result, updating data path forwarding tables could take time. More importantly, even if the central server does not have to download updated forwarding state, it remains responsible for coordinating when and which FEs switch-over to the new state. As discussed in [14], failure to do so can introduce forwarding loops, whose effect can be worse than failures.

Ensuring uninterrupted (or minimally interrupted) packet forwarding in the presence of failures remains, therefore, a significant challenge in centralized routing systems. Our goal in this paper is to explore a possible solution to this problem, and in the process take centralized routing one step closer to offering an effective alternative for intra-domain routing. Furthermore, because a corollary of centralized routing is simplified FEs, we seek to realize this goal with no or minimal impact on data plane complexity. In particular, we want to avoid either encapsulation-based solutions that require additional packet manipulations, as well as packet marking and interface specific forwarding solutions that often call for significant expansion to the size (and therefore cost) of forwarding tables. Instead, our goal is to allow all (most) FEs to have, for each destination present in their forwarding tables, a pre-configured next-hop to which packets for that destination can be forwarded in case of failure of the primary next-hop(s). The trigger to switch to backup forwarding is entirely local (*i.e.*, unavailability of the primary next-hop(s)), and forwarding loops should be precluded.

In other words, we consider an IP network where (intra-domain) routing is under the responsibility of a central server, so that routers (FEs) are only responsible for (destination-based) packet forwarding. Because of the use of a central server, path computation is not restricted to shortest paths based on a common set of link weights. Instead, each destination prefix is associated with an "independently" computed (primary) forwarding tree (more generally a directed acyclic graph, or DAG), rooted at the egress node associated with the destination. Our goal is then to compute a set of primary routing trees (or DAGs), one for each destination, so that all

This work was supported by NSF grants CNS-0627004, CNS-0626617, and CNS-0626808.

nodes in the tree, or when not feasible[1] as many nodes as possible, have a standby alternate next-hop available when the primary next-hop becomes unreachable. We term such a routing, *protection routing*, and introduce it more formally in Section III. Protection routing is readily realized when each node in the DAG has two or more independent next-hops towards the destination, *e.g.*, as sought in [19], [15]. Its simplicity not withstanding, this is easily shown not to be simultaneously feasible for all nodes (at least one node is limited to only one next-hop). Furthermore, it ignores the option for two nodes to mutually protect each other, and exploring the benefits this affords is one of the motivations for this paper. In addition, while standby protection to failures is desirable, its impact on operational performance should also be accounted for. Incorporating this aspect when computing protection routing is another goal of the paper.

The concept of protection routing as just defined bears similarities with a number of related concepts, and we expand on this in Section II. The paper nevertheless makes a number of novel contributions and in particular:

1) It offers new insight into network topological properties that ensure the feasibility of protection routing;
2) It establishes that computing a protection routing is an NP-hard problem;
3) It develops a heuristic for computing a routing that reconciles the often conflicting goals of protectability and performance;
4) It demonstrates the heuristic's ability to realize an effective trade-off between protectability and performance across a range of network topologies.

The rest of the paper is structured as follows. Section II reviews related works and contrasts the approach and findings of the paper against them. Section III introduces the concept of protectability more formally and defines protection routing. Section IV is devoted to an analytical investigation of protection routing, while Section V leverages insight from this analysis to develop a heuristic for computing protection routings. The heuristic favors protectability, while trying to minimize its impact on performance. The underlying trade-off is further investigated in Section VI, which develops a modified heuristic that allows relaxed protectability goals for the sake of improving performance. Section VII evaluates the efficacy of the heuristics in several different scenarios, and Section VIII summarizes the paper's findings.

II. Related Works

This paper considers a centralized routing system similar to that proposed in [6], [8], [22]. In those works, the primary motivation for centralizing path computation was manageability. In [14], an efficient message-dissemination solution was proposed to minimize signaling overhead and avoid the formation of transient loops in such an environment. This paper builds on these earlier works by assuming a centralized routing solution, but differs in its focus. Its aim is to overcome problems associated with the potential for increased latency after failures, because of the system's reliance on a central server responsible for coordinating updates to the forwarding states of FEs. Our motivations and general approach for handling this issue are similar in principle to those behind many of the IP fast re-routing (IPFRR) schemes that have been proposed (see [20] for a generic introduction to IPFRR and its goals). We expand below on specific differences between our solution and individual IPFRR mechanisms, but an important contributor to those differences comes from our ability to exploit the flexibility afforded by centralized path computations to produce routing solutions that are difficult, if not impossible, to realize in the traditional, distributed environment assumed by most IPFRR solutions.

IPFRR's main goal is to ensure fast (sub-50ms) convergence of intra-domain routing protocols, as soon as failures have been detected. Current proposals fall in either one of two categories: those that can operate with an unmodified IP forwarding plane; and those that involve the use of a different (usually more complex) forwarding paradigm. The former category is the more relevant to this paper, which also seeks to offer protection to failure while preserving the simplicity and scalability of IP forwarding. In particular, one of our goals is to maximize the fast re-routing "coverage" achievable in any network by taking advantage of the flexibility of centralized routing in computing paths and controlling local forwarding decisions at each FE.

Examples of IPFRR mechanisms belonging to the first category include Loop-free alternate (LFA) [1], O2 [19], [17], [16], DIV-R [15] and MARA [13]. The LFA proposal of [1] is aimed primarily at IP networks that run distributed, shortest-path-based routing algorithms. Furthermore, it relies on a criterion for ensuring loop freedom when selecting next-hop alternates (backups) (see [1][Inequality 1]) similar to the invariant of [15]. As alluded to earlier, imposing such a requirement prevents neighboring nodes from backing each other up (the criterion enforces an ordering among nodes, so that only one is eligible as a backup for the other). Both factors limit the coverage that the scheme is able to provide. This limitation is not present in O2 [19], which is not restricted to using shortest paths and that introduces the concept of "joker" links specifically for the purpose of allowing mutual backups. These similarities make the O2 body of work [19], [17], [16] the most relevant to this paper, and it is, therefore, important to articulate differences in both scope and contributions.

O2 shares with this paper its applicability to (or more precisely, need for) a centralized routing system, and the goal of maximizing the number of nodes that are protected against any single link or node failure. In O2, this is realized by ensuring that every node has an "out-degree" (number of next-hops) of two - hence the name O2 - with one of them available as a backup in case of failures. This is similar to our goal as stated in Section I, with the difference that we do not seek to impose a limit of two on the out-degree, and will often allow more, especially when trying to reconcile the need for load-

[1]It is easy to construct network graphs for which no matter what routing is chosen, one or more nodes have no alternate next-hop.

balancing with protectability. As a matter of fact, exploring the trade-off that exists between protectability and performance is one of the important differences between our work and O2. This difference is further reflected in the path computation algorithm we propose to jointly optimize protectability and performance. We demonstrate in Section VII the benefits of our algorithm in terms of both performance and protectability, when compared to O2 algorithms [16]. Another difference between our work and the O2 contributions is our focus on identifying specific conditions for the feasibility of a protection routing, and conversely the complexity of finding one when it exists. In particular, we formally establish in Section IV that the problem of computing a protection routing is NP-hard, and provide several characterizations of network topology that affect the feasibility of protection routing.

The DIV-R algorithm of [15] and the several MARA algorithms of [13] have similar goals as O2 and this paper, but differ in their approaches. DIV-R proposes a distributed algorithm to maximize a metric that reflects the number of next-hops available to each node. This may be effective against link failures, but as shown in Section VII, less so when considering node failures. The MARA algorithms consider several path computation problems aimed at improving minimum connectivity and fully utilizing all available links; hence affording greater resilience to failures (MARA's all-to-one maximum connectivity problem is the most relevant, and similar in spirit to DIV-R). As with DIV-R, protection against node failures is not explicitly taken into account and neither is the trade-off between performance and protectability.

The second category of IPFRR works includes [23], [9], [21], [10], which seek to deliver protectability irrespective of network topological limitations at the cost of possible changes to packet forwarding. For example, [23] considers the use of interface-specific forwarding tables to handle packet re-routing after failures while preventing loops. Multiple "topologies" are used in [9], each covering different failures, with routers switching from one to another upon detecting a given failure and marking packets according to the topology to be used to overcome it. In [21], protection is achieved by using tunnels to detour packets around failures; hence requiring packet encapsulation and decapsulation. Finally, [10] proposes carrying root-cause failure information in packets to allow routers to diagnose problems and select alternate paths.

III. Model and Problem Formulation

We model the network as a directed graph $G = (V, E)$, with V the node set, E the link set, and $|V| = n$. A directed link from node i to node j is denoted by (i, j). $N_G(i) = \{j \in V | (i, j) \in E\}$ is the neighbor set of node i in G. As discussed in Section I, we assume that information such as network topology and link bandwidth is available to a central server for the purpose of path computation. We further assume that packet forwarding is destination-based without reliance on packet marking or encapsulation even in the presence of failures, *i.e.*, the standard IP forwarding paradigm.

For a destination[2] $d \in V$, let $R_d = (V, E_d)$ be a routing for traffic destined to d, where $E_d \subseteq E$. R_d is a directed acyclic graph (DAG) rooted at d and defines a destination-based routing. In R_d, every node $i \in V \setminus \{d\}$ has at least one outgoing link. A node j is called a primary next-hop (PNH) of node i if $(i, j) \in E_d$, and the link (i, j) is called a primary link of node i. If a node has multiple PNHs, traffic is split evenly across them. One advantage of centralized routing is that R_d's can be computed independently of each other. In contrast, a standard IGP such as OSPF computes routings that are coupled by a common set of link weights. Thus, without loss of generality, in the remainder of this section we focus on a single destination d.

When computing R_d, our goal is to preserve uninterrupted packet forwarding in the presence of any single "component" (link or node) failure, except for that of d itself.

Definition 3.1: After a single component failure f, the resulting network and routing for destination d are denoted by G^f and R_d^f, respectively. G^f and R_d^f are constructed by removing the failed component (node and/or link(s)) associated with f from G and R_d respectively.

Definition 3.2: Node i is said to be upstream of node j in a routing R_d if there exists a path from node i to node j in R_d. Conversely, node j is then downstream of node i.

Definition 3.3: In a routing R_d, node $i \neq d$ is said to be protected (with respect to d), if after any single component failure f that affects node i's PNH(s), there exists a node $k \in N_{G^f}(i)$ such that the following two conditions are satisfied:

1) Node k is not upstream of node i in R_d^f.
2) Node k and all its downstream nodes (except d) have at least one PNH in R_d^f.

Node k is called a secondary next-hop (SNH) of node i for f and d. By convention, destination d is always protected.

Definition 3.3 is inspired by LFA but does not mandate the use of shortest paths, nor does it require [1][Inequality 1] to prevent loops. The two conditions of Definition 3.3 imply that when the PNH of node i fails and packets are rerouted to node k: (i) routing loops never form (condition (1)); and (ii) packets are delivered to d through node k and its downstream nodes in R_d^f (condition (2)). Examples illustrating the feasibility or infeasibility of these conditions are provided in Section III-A.

Definition 3.4: R_d is said to be a protection routing if every node $i \in V$ is protected in R_d.

Definition 3.5: A graph $G = (V, E)$ is said to be protectable if a protection routing exists $\forall\, d \in V$.

By Definition 3.4, if R_d is a protection routing, packet forwarding (and delivery) to d can proceed uninterrupted in the presence of any single component failure (besides that of d itself). The main challenges are in *identifying* when such routings are feasible, and in *computing* them, or when not feasible, computing routings that maximize the number of protected nodes. We discuss these in Sections IV, V, and VI, but proceed first with some illustrative examples.

[2]For simplicity, we associate each node with a single destination, while in practice this would encompass all prefixes for which a node is the egress.

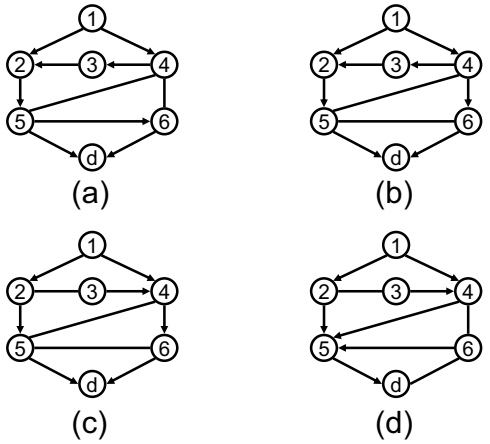


Fig. 1. Different R_d's (denoted by arrows) of a network.

A. Discussion

Fig. 1 illustrates on a simple network topology how different routing choices affect protectability for a destination d. The routing of Fig. 1(a) does not protect nodes 1, 2, 3 and 6 under Definition 3.3. For example, although node 1 has two PNHs, it is not protected against a failure of node 2. This is because its other PNH, node 4, is itself upstream of the failed node 2 (this violates condition (2) of Definition 3.3). Similarly, node 6 is not protected against a failure of link $(6, d)$, as its two neighbors, nodes 4 and 5, are both upstream of itself (this violates condition (1) of Definition 3.3). The routing of Fig. 1(b) succeeds in protecting node 6 against the failure of link $(6, d)$, because node 5 is now a valid SNH. However, according to condition (2) of Definition 3.3, node 1 is still not protected against a failure of node 2, as even if node 4 now has a PNH (*i.e.*, node 6) that does not rely on node 2, it will still forward some packets destined to d towards node 3 (node 4 is unaware of the failure of node 2 and load-balances across its two PNHs) that remains unprotected. This last issue is resolved in Fig. 1(c), where all nodes are now protected. Note that to ensure protectability, more links are left unused during normal operations, so that they are available for mutual backups after failures. This illustrates the tension that exists between performance and protectability, and is one of the issues we explore further in Sections VI and VII.

Fig. 1(d) illustrates a subtle issue that arises from the choice of conditions in Definition 3.3, and in particular condition (2) that calls for backup paths to only use PNHs. Fig. 1(d) gives an example of a routing that node 2 is not protected according to Definition 3.3, but that is still able to deliver packets to d after a failure of node 5. This is because, when node 5 fails, node 2 forwards packets to its SNH, node 3, which passes them to its PNH, node 4. Node 4's PNH, however, was also node 5, so that it must also forward packets to its own SNH, node 6, which finds itself in a similar situation and forwards packets to its own SNH, namely, d. This does ensure delivery of packets to d, but violates condition (2) of Definition 3.3. An intuitive "fix" might seem to simply relax condition (2) to allow packet forwarding using both PNH and SNH. This is unfortunately not possible, as such a relaxation could allow the formation of loops. In general, instances where backup paths such those of Fig. 1(d) improve protectability appear to be limited. Furthermore, systematically exploring them can add significant computational complexity, as all possible combinations of PNHs and SNHs need to be considered. Section V-D introduces a compromise based on an algorithm that iterates over possible SNH assignments once a choice of PNHs has been finalized, and allows the discovery of paths such as those of Fig. 1(d).

IV. Analysis

In this section, we model a network as an undirected graph $G = (V, E)$, so that finding a protection routing is equivalent to identifying an orientation for a subset of links such that every node is protected, *i.e.*, an ordering among nodes that makes re-routing possible without creating loops. Its existence depends on routing choices *and* the topological structure of the network. The goal of this section is to analyze what topological properties are sufficient to ensure protectability and characterize the algorithmic complexity of finding it. Graph terminology not defined in the paper can be found in [5].

A. Does a simple sufficient condition exist?

We first consider the existence of sufficient conditions for protectability. A necessary condition for a graph to be protectable is for every node to have two neighbors. Thus, it is natural to ask if the node degree of a graph can be used to characterize protectability. A simple sufficient condition for a graph to be protectable is as follows.

Theorem 4.1: Every graph with $n \geq 5$ nodes and minimum degree at least $\lceil n/2 \rceil$ is protectable.

Proof: See Appendix A. ■

Theorem 4.1 cannot be improved in that we cannot replace the bound of $\lceil n/2 \rceil$ with $\lfloor n/2 \rfloor$, as there exists a 1-node-connected graph[3] with minimum degree $\lfloor n/2 \rfloor$, which is obviously not protectable.

Theorem 4.1 implies that in the absence of any global graph property, a high minimum degree is needed to guarantee protectability. A natural next step is to explore if introducing global graph properties such as link- and node-connectedness can yield less stringent sufficient conditions for protectability. Intuitively, k-link-connectedness, for k large enough, would seem sufficient to ensure protectability. Surprisingly, this is not true in general, no matter how large k is. The result is summarized as follows.

Theorem 4.2: For any given $k \in \mathbb{Z}^+$, there exists a k-link-connected graph that is unprotectable.

Proof: See Appendix B. ■

Theorem 4.2 establishes that even with arbitrarily many link-disjoint paths, a protection routing is not guaranteed to exist. A similar question can be asked using the stronger condition of

[3]If n is odd, such a graph can be constructed by taking the union of two copies of complete graph $K^{\lceil n/2 \rceil}$ connected at one node.

node-connectedness. In this setting, we only have the weaker result of Theorem 4.3 and a conjecture. Specifically,

Theorem 4.3: For $k = 2, 3$, there exists a k-node-connected graph that is unprotectable.

Proof: See Appendix B. ∎

Conjecture 4.1: For any given $k \geq 4$, there exists a k-node-connected graph that is unprotectable.

The reason the relatively strong properties of Theorems 4.2 and 4.3 fail to ensure protectability is because destination-based routing induces an *ordering* among nodes; something that is not present when, for example, computing node-disjoint paths. Hence, even if each node *individually* has several disjoint paths to a destination, this need not hold when coupling them through a common destination-based routing.

B. On random graphs

The previous results showed that even graphs with very rich connectivity, *e.g.*, large degree or connectedness, were not guaranteed to be protectable. However, the proofs of these results involved graphs with very specific structure. A natural question is whether such graphs are the norm or the exception. To explore this question, we rely on a family of graphs with little or no special structure, *i.e.*, random graphs, and investigate what can be said about their protectability. We use Erdös-Rényi random graphs $G(n, p)$, where n denotes the number of nodes and p is the link probability, and analyze under what conditions such graphs are protectable as n becomes large. This calls for finding routings for each destination such that all nodes have a suitable SNH to reroute traffic after failures. The random and relatively homogeneous structure of random graphs makes it possible to establish the following result.

Theorem 4.4: Let $G \in G(n, p)$ and $p = (4 + \varepsilon) \log n/n$ where $\varepsilon > 0$ is any constant and $\log$ is the natural logarithm. Then asymptotically almost surely G is protectable.

Proof: See Appendix C. ∎

Theorem 4.4 implies that a mean degree that grows like $O(\log n)$ is sufficient to ensure that a random graph is protectable with probability tending to 1 as $n \to \infty$. In other words, in the absence of structure explicitly aimed at defeating it, the level of connectivity required to ensure protectability is significantly lower than that required by Theorem 4.1. Although random graphs are not representative of all network topologies, this provides some hope that protectability is feasible in many practical networks with reasonable connectivity. The next section is devoted to assessing how difficult a task computing such protection routings is.

C. NP-completeness of protection routing

This section analyzes the algorithmic complexity of computing a protectable routing. For that purpose, we formulate the **PR problem** as follows.

- Instance: Given an undirected graph $G = (V, E)$ and a destination node $d \in V$.
- Question: Does a protection routing destined to d exist?

Theorem 4.5: The PR problem is NP-complete.

Proof: See Appendix D. ∎

Theorem 4.5 indicates that in an arbitrary graph there is no known polynomial-time algorithm to solve the PR problem unless P=NP. The proof is based on a reduction from the 3SAT problem. Heuristics are, therefore, required to compute protection routings.

V. Heuristic Design

Our goal is to compute n routings (one for each destination) to maximize protectability while realizing good network performance (*e.g.*, congestion) in normal (failure-free) situations. Computing a routing to minimize the number of unprotected node for a destination is NP-hard because the decision version of the problem is NP-complete. Because all n routings contribute to link loads, adding the dimension of performance introduces a coupling that only makes the problem harder. Practical solutions must, therefore, rely on heuristics.

A. Heuristic outline

Our heuristic seeks routings $R_d, \forall\, d \in V$, that minimize the number Ω_d of unprotected nodes for their respective destination, and that together minimize network congestion in the absence of failure. Network congestion is measured through a cost function Φ. For illustration purposes, we select the function $\Phi = \sum_{l \in E} \Phi_l$ of [7], where Φ_l denotes the congestion cost of link l as a function of its load. Other expressions for Φ can be readily used.

To keep computational complexity low and preserve the ability to independently compute routings that minimize Ω_d for each $d \in V$ while accounting for performance (congestion), we design a two-phase heuristic. Phase 1 allows independent computations of protection routings for each destination, while Phase 2 considers them jointly and attempts to modify them to optimize performance without hurting protectability.

B. Phase 1 - Greedy Search

Phase 1 uses a greedy search with a cost function F_d, $d \in V$, that focuses on Ω_d but remains congestion aware. Congestion is not explicitly accounted for in F_d to preserve independent computations across destinations. It is used to influence the greedy exploration of the solution space.

Specifically, prior to Phase 1, a standard traffic optimization routine, *e.g.*, [7], is run to assess the best network congestion cost Φ^{opt} in the absence of protectability considerations. This provides routings, $R_d^{opt}, d \in V$, that achieve Φ^{opt}, as well as a benchmark against which to compare network congestion costs under protection routing. Each routing R_d^{opt} can be computed using a shortest path algorithm with appropriate link weights. These link weights are used in Phase 1 to compute a deviation $||R_d - R_d^{opt}||$ between a proposed protection routing R_d and R_d^{opt}. This deviation is measured[4] using $\Gamma_d = \sum_{i \in V} \Gamma_{i,d}$, where $\Gamma_{i,d}$ denotes the distance from node i to destination d under R_d, with distances computed using the link weights of R_d^{opt}. The smaller Γ_d, the "closer" R_d is to R_d^{opt}. This metric guides the selection of solutions during Phase 1 as follows.

[4]Other measures can easily be accommodated.

The cost function F_d is defined as $F_d = \langle \Omega_d, \Gamma_d \rangle$ where $\langle a_1, b_1 \rangle > \langle a_2, b_2 \rangle$ if and only if $a_1 > a_2$, or $a_1 = a_2$ and $b_1 > b_2$. This gives precedence to protectability, while favoring solutions with lower congestion costs (as measured through Γ_d) when it does not affect protectability. The optimization carried out in Phase 1 is then of the form

$$\forall\, d \in V \quad \underset{R_d}{\text{minimize}}\ F_d = \langle \Omega_d, \Gamma_d \rangle \ . \tag{1}$$

Note that although Γ_d in F_d accounts for congestion, computations for different destinations are still decoupled. This is because Γ_d is computed based on a fixed reference point (*i.e.*, the link weights that produced R_d^{opt}). This also avoids evaluating the cost function Φ for each candidate routing, an operation that in itself has a significant computational cost.

A "Greedy-search" heuristic (see Appendix E for details) is used to minimize Eq. 1. It was inspired by approximation algorithms for the 3SAT problem from which the NPC of the PR problem is reduced, and operates on routings limited to trees (*i.e.*, each node except the destination has only one PNH). There are two motivations for the latter. First, assigning multiple PNHs to a node may affect the protectability of other nodes as discussed in Section III-A. Second, the sheer number of possible combinations involving multiple PNHs makes it computationally impractical to consider them all. Allowing multiple PNHs can obviously reduce congestion through better load-balancing. This aspect is considered separately in Phase 2.

The heuristic starts with an initial routing $R_d = T\left(R_d^{opt}\right)$ obtained by extracting a tree from R_d^{opt} (when multiple next-hops are available, one is randomly selected). The main loop uses local feasibility checks to explore improvements in F_d when swapping the PNH of node $i \in V \backslash \{d\}$. This process repeats until F_d shows no improvement for all nodes. A diversification step is then executed, and generates a new random shortest path tree rooted at d. Unlike the first tree based on R_d^{opt}, the new tree is generated using random link weights uniformly selected in $[1, 1000]$. This ensures that after exploring the neighborhood of R_d^{opt}, the search restarts at a different point of the solution space[5]. The heuristic stops after P diversifications without improvement to F_d.

C. Phase 2 - Load-Balancing

The routing trees R_d^*, $\forall d \in V$, of Phase 1 are used as inputs to Phase 2. Phase 2 seeks to assign multiple PNHs to nodes to better distribute traffic (load-balance), subject to the constraint that the number of unprotected nodes cannot increase.

Its main loop (see Appendix E for details) examines each node i in decreasing order[6] of its congestion, and tries to assign it multiple PNHs to better load-balance traffic and reduce its congestion cost. Note that Phase 2 involves evaluating Φ for each candidate routing, and this is where the bulk of its computational cost lies. The heuristic stops when Φ cannot be further reduced through new PNH assignments.

[5]Other diversification methods were tried, *e.g.*, shuffling a subset of PNHs in the tree, generating increasingly perturbed versions of R_d^{opt}, etc. The more diverse starting points of a random diversification consistently resulted in a better exploration of the solution space.

[6]Other orders, *e.g.*, random, fixed, were tried and found to perform worse.

D. SNH assignment

The first two phases of the heuristic produce a set of routings that maximize the number of protected nodes while minimizing congestion cost by load-balancing across multiple PNHs, as long as it does not affect protectability. By definition of protectability, all protected nodes have at least one SNH they can use in case of failure to forward packets on an alternate path that delivers packets to the destination solely through PNH forwarding. As discussed earlier, the restriction to PNH forwarding imposed by Definition 3.3 precludes backup paths involving multiple SNHs, which could improve protectability. Allowing such paths, however, requires some care to avoid loops. In this section, we describe an algorithm that assigns SNH (when a choice is available) to allow backup paths involving multiple SNHs, while ensuring the absence of loops. The algorithm is outlined for a given d with details in Appendix E.

$R_d = (V, E_d)$ denotes the routing for d produced by Phases 1 and 2, where $E_d \subseteq E$ is the set of primary links. After failure f, $R_d^f = (V^f, E_d^f)$ denotes the residual routing after removing the failed component(s). Let $S_d^f : V \rightarrow V \cup \{\emptyset\}$ denote the SNH assignment mapping for failure f, with $S_d^f(i) \in V \cup \{\emptyset\}$ the SNH assigned to node i. An empty assignment, *i.e.*, $S_d^f(i) = \emptyset$, implies that there is either no need to assign an SNH to node i because its PNH is not affected by f, or no suitable SNH can be found. Our goal is to explore SNH assignments that maximize protectability when allowing backup paths that involve multiple SNHs.

Let $H_d^f = \left(V^f, E_d^f \bigcup_{i \in V, S_d^f(i) \neq \emptyset} (i, S_d^f(i))\right)$ be a routing under failure f. Note that H_d^f combines R_d^f and S_d^f, and hence permits the use of multiple SNHs. This calls for additional precautions when assigning SNHs. Specifically, assume that node k is a candidate SNH for node i after failure f. Node k can be selected if the following two conditions are satisfied: (H1) H_d^f remains a DAG after the addition of link (i, k); (H2) Node k and all its downstream nodes (except d) have an out-degree of at least one in H_d^f. Condition (H1) ensures that loops are avoided, while Conditions (H1) and (H2) together guarantee packets delivery to d. Using these two conditions, SNHs can be assigned to improve protectability of nodes affected by failure f and with initially (after Phases 1 and 2) no feasible SNH, *i.e.*, $S_d^f(i) = \emptyset$. This continues until no SNH assignment satisfying conditions (H1) and (H2) is found. Note that the fact that PNHs remain fixed is in part what keeps computational complexity manageable. Additional details on the computational complexity of the heuristic can be found in Appendix E.

VI. Trading Protectability for Performance

The cost function F_d gives strict precedence to protectability. A natural question is whether this can be relaxed to trade-off protectability for performance. Such a trade-off can be

formulated using the following optimization:

$$\underset{R_d, d \in V}{\text{minimize}} \ \Phi \tag{2}$$

subject to

$$\Omega_d \leq (1 + \varepsilon_d)\, \Omega_d^* \qquad \forall d \in V \tag{3}$$

where Ω_d^* denotes the smallest possible number of unprotected nodes for destination d, and $\varepsilon_d \geq 0$ controls how much protectability can be traded-off for performance.

In realizing such a trade-off, computational complexity is again the main concern. Our proposed solution is based on two observations: (i) computing Ω_d^*, $\forall d \in V$, as required by Eq. 3, calls for performing Phase 1; and (ii) a large number of routings are examined during Phase 1. A natural option is to take advantage of the availability of those routings. Specifically, we keep *all* routings examined during Phase 1, and at the end of Phase 1 we identify those that satisfy Eq. 3. We then select for each destination d, the routing that minimizes Γ_d. Those routings can subsequently be further improved by invoking Phase 2.

This approach leverages the computational tractability of the previous heuristic (it has the same computational complexity, and avoids most expensive computations of the cost function Φ), and the additional memory it requires to store the routings examined during Phase 1 is relatively small. Intelligently discarding routings whenever they fail to satisfy Eq. 3 based on the current estimate of Ω_d^* can further reduce this memory.

VII. Evaluation

This section assesses the extent to which our heuristic can find efficient protection routings, and explores the trade-off between performance and protectability. It starts with a review of the environment in which this evaluation is conducted.

A. Evaluation settings

1) Network topologies: Both real and synthesized topologies are used.

- *RN*: Random topology of given average node degree.
- *PL*: Power-law topology based on the preferential attachment model [2].
- *AS*: Real topologies from the Rocketfuel project [18] and labeled by their AS numbers[7].

Link capacities are all set equal to unity with traffic demand (see below) used to generate heterogeneous load levels because heterogeneous loads can be generated either by varying traffic demand or link capacity.

2) Traffic matrix: The traffic matrix $M = [r(s,t)]_{|V|\times|V|}$ is generated using a gravity model [11], [3], [12] as follows: Traffic volume from node s to node t is defined as $r(s,t) = b_s \frac{e^{a_t}}{\sum_{i \in V \setminus \{s\}} e^{a_i}}$ where b_s is the total traffic originating at node s, and is given by

$$b_s = \begin{cases} \text{Uniform(10, 50)}, & \text{with prob 0.6} \quad (4a) \\ \text{Uniform(80, 130)}, & \text{with prob 0.35} \quad (4b) \\ \text{Uniform(150, 200)}, & \text{with prob 0.05} \quad (4c) \end{cases}$$

[7]Nodes isolated from the giant component are removed.

Uniform(a,b) denotes a random variable uniformly distributed in $[a,b]$, a_t is the "mass" of node t which is proportional to the number of links it has and $\sum_{i \in V} a_i = 1$. The larger a node's mass, the more traffic it attracts. Using b_s, it generates three different levels of heterogeneous load. Finally, M is scaled to produce a reasonable link utilization in the network.

3) Parameter settings: Our heuristic involves only one parameter, P, used as the stopping criterion of Phase 1. We set $P = 10$, so that Phase 1 is stopped if there is no improvement after 10 diversification rounds. This value was chosen as it balances solution quality and computational time in our experiments.

4) Comparison: We use the proposed two-phase heuristic and the SNH assignment algorithm to compute protection routing solutions, and the results are denoted by PR. Our solutions are compared to the following previous works:

- *SP*: Routings computed by the OSPF optimization in [7].
- *DIVR*: Routings computed by DIV-R from [15].
- *O2*: Routings computed by the pattern-based algorithm[8] of [16].

We believe that this provides a reasonable coverage of both the heuristic's performance across different networks, and its comparison to other alternatives. SP is commonly used for intra-domain routing in large ISP's, *e.g.*, [12], and focuses solely on performance. DIVR, like [13], seeks to maximize the number of PNHs at each node but without considering the use of SNHs after failures. O2 optimizes for protectability, but is oblivious to performance. In the experiments, a node is said to be protected with respect to a destination if it has a valid re-routing option for that destination after any single component failure.

B. Benefits of protection routing

We first investigate the effectiveness of PR on synthesized topologies with mean degree varying from three to five. Fig. 2 shows the numbers of protected nodes across destinations, with the x-axis showing destination IDs sorted in ascending order of the number of protected nodes under SP. The results illustrate that PR significantly improves protectability when compared to other solutions. Moreover, the results show that the gap is still present even in richly connected topologies for which, as indicated by Theorems 4.1 and 4.4, a protection routing is more likely to exist. Hence, even in those topologies, protection routings remain difficult to find unless an efficient heuristic such as PR is used.

It should be noted that the relatively poor performance of DIVR can, as mentioned earlier, be partly attributed to its focus on link failures that makes it more susceptible to node failures. Another finding from the figure is that a mean degree of 4 (*i.e.*, 70 nodes and 140 links) appears sufficient to realize near 100% protectability with PR. This indicates that protectability should be feasible in practice under reasonable connectivity.

The results of another set of evaluations carried out on real ISP topologies are shown in Fig. 3, where the mean degrees

[8]This algorithm is chosen among several O2 heuristics, because, as reported in [16], it provides better protectability.

TABLE I
NETWORK PERFORMANCE ACROSS TOPOLOGIES.

Topology [# nodes, # links]	RN [70,105]	RN [70,140]	RN [70,175]	PL [70,105]	PL [70,140]	PL [70,175]	AS 1221	AS 1755	AS 3967
Avg link load (PR)	0.20	0.31	0.22	0.24	0.21	0.29	0.11	0.26	0.15
Avg link load (SP)	0.18	0.28	0.20	0.21	0.19	0.28	0.10	0.23	0.13
Avg link load (DIVR)	0.22	0.44	0.36	0.25	0.28	0.43	0.12	0.31	0.18
Avg link load (O2)	0.18	0.32	0.25	0.22	0.23	0.35	0.11	0.24	0.14
Max link load (PR)	0.86	0.66	0.55	0.66	0.53	0.74	0.93	0.98	0.91
Max link load (SP)	0.66	0.66	0.55	0.67	0.62	0.84	0.93	0.90	0.89
Max link load (DIVR)	0.90	1.36	1.23	1.00	1.56	2.92	1.13	1.57	1.17
Max link load (O2)	1.12	1.18	1.29	1.33	1.53	2.60	1.30	1.23	1.18
Increase in Φ under PR (%)	48.06	11.79	0.12	15.12	10.60	-3.98	3.73	19.72	32.51
Increase in Φ under DIVR (%)	55.05	4690	3284	91.86	13203	56690	492	3619	889
Increase in Φ under O2 (%)	495	978	1358	4455	11112	44917	3792	1246	1667

of AS1221, AS1755 and AS3967 are 2.90, 3.70 and 3.72, respectively. The figure offers similar conclusions, namely, PR is effective in computing protection routings, and its advantage over other solutions remains even in richly connected ASes such as AS3967.

Table I shows network performance metrics across topologies, where comparisons with SP reflect the cost of protectability. In the case of Φ, increases relative to SP are reported for PR, DIVR and O2. Under PR, network performance typically degrades slightly compared to SP. This is expected because PR leaves some links unused under normal conditions to ensure they are available for protection after failures. This cost is, however, small, especially in comparison to that incurred by DIVR and O2, which often result in very high levels of congestion. This is in part because both are oblivious to performance goals when computing routings, and demonstrates that PR is successful at reconciling both.

C. Trading protectability for performance

Following the discussion of Section VI, we study whether it is possible to improve performance if we are willing to sacrifice some protectability. For simplicity, we assume that in Eq. 3, $\varepsilon_d = \varepsilon$, $\forall d \in V$.

Table II illustrates the trade-off between protectability and performance for two topologies. The table uses results for $\varepsilon = 0$ (*i.e.*, no trade-off) as a benchmark for the decrease[9] in Φ realized by an increase in $\overline{\Omega}_d$, the average number of unprotected nodes across all destinations for different ε's. For reference, we also give $\overline{\Omega}_d$ in the table. The traffic matrices used in the experiments produce roughly 70% maximum link utilization when $\varepsilon = 0$.

The main observation from the results of Table II is that the proposed heuristic successfully realizes different trade-offs between protectability and performance. As a result, it provides network operators with a tunable solution for selecting a routing that provides the desired balance between protectability and performance. In addition, since the solution has essentially the same computational complexity as the base heuristic, it can be readily used in practice as we discuss next.

D. Computational complexity

To support our claim of computational efficiency, we report computational times for some large topologies. Specifically, run times were 1.7 hours, 1.63 hours, and 0.79 hours for the RN [70,175], PL [70,175], and AS 1221 topologies, respectively. These results are obtained with a Pentium Xeon 2.66 GHz machine. Note that the computation times are realized without trying to explicitly take advantage of the inherent parallelism of the computations (the n routings of Phase 1 are independent and can be computed in parallel). The other metric of importance when assessing computational cost is memory consumption. None of the experiments required more than 300MB of memory.

[9]The relative decrease of Φ is at most 100% which corresponds to $\Phi = 0$.

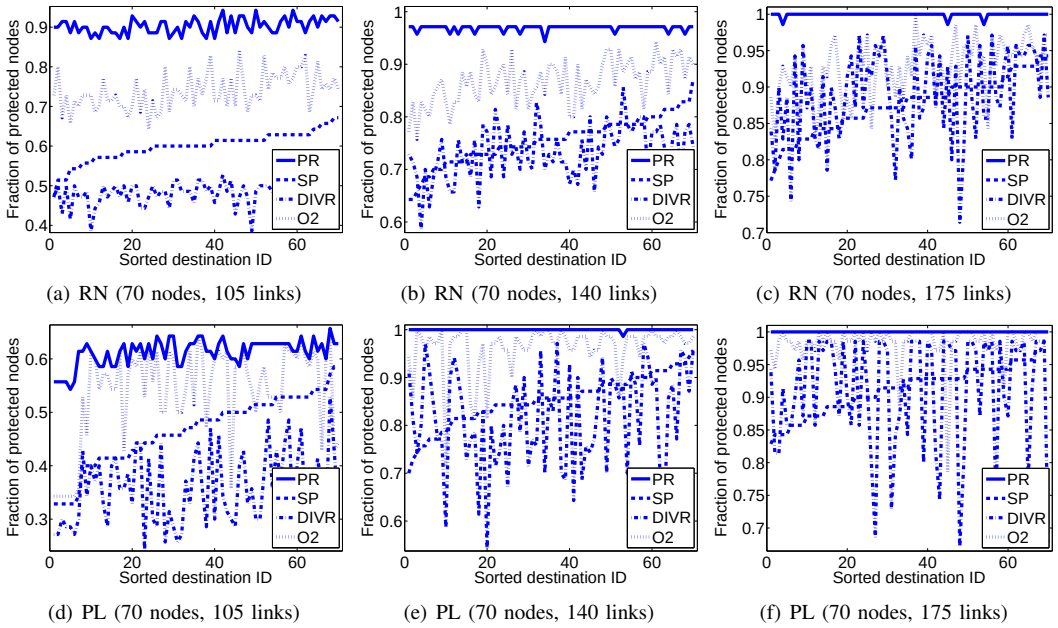
TABLE II
TRADEOFF BETWEEN PROTECTABILITY AND PERFORMANCE.

ε	0	0.2	0.5	1	1.5	2
Random topology (70 nodes, 105 links)						
Decrease in Φ (%)	0	9.74	19.85	24.81	28.09	30.90
Increase in $\overline{\Omega}_d$ (%)	0	14.81	47.25	78.81	127.40	176.59
$\overline{\Omega}_d$ (in nodes)	6.75	7.75	9.94	12.07	15.35	18.67
AS3967 (79 nodes, 147 links)						
Decrease in Φ (%)	0	10.17	16.99	19.29	20.77	22.68
Increase in $\overline{\Omega}_d$ (%)	0	10.45	35.67	57.81	87.08	126.07
$\overline{\Omega}_d$ (in nodes)	8.13	8.98	11.03	12.83	15.21	18.38

VIII. CONCLUSION

This paper has investigated the feasibility of protection routing in a centralized routing system, which displays heightened sensitivity to failures due to latency in responses from the central server. The paper identified topological properties that affect the feasibility of protection routing and established that computing protection routings is NP-hard. It developed an efficient heuristic to compute routings that not only optimize protectability, but also minimize its performance cost. The heuristic was shown to out-perform earlier proposals, and its efficacy demonstrated for a range of topologies.

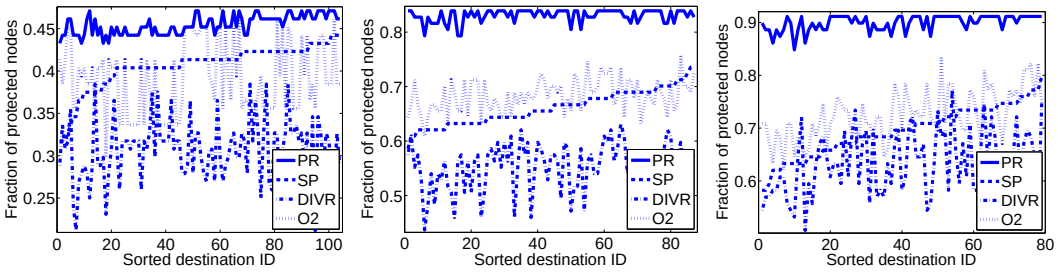
There are many directions in which this work can be extended or built on. The first is to demonstrate the feasibility of protectability in a centralized routing system through an implementation. Another direction of interest is to develop "weighted" protectability solutions, *i.e.*, to account for the fact that certain nodes are more important than others. Yet another area is to develop update mechanisms at the central server that are aware of which nodes have protection and which do not, and select update orderings based on this information and the need to avoid loops when updating forwarding states.



(a) RN (70 nodes, 105 links) (b) RN (70 nodes, 140 links) (c) RN (70 nodes, 175 links)

(d) PL (70 nodes, 105 links) (e) PL (70 nodes, 140 links) (f) PL (70 nodes, 175 links)

Fig. 2. Protectability across synthesized topologies.



(a) AS 1221 (104 nodes, 151 links) (b) AS 1755 (87 nodes, 161 links) (c) AS 3967 (79 nodes, 147 links)

Fig. 3. Protectability across real topologies.

REFERENCES

[1] A. Atlas and A. Zinin, "Basic specification for IP fast reroute: Loop-free alternates," IETF RFC 5286, September 2008.

[2] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, October 1999.

[3] S. Bhattacharyya, N. Taft, J. Jetcheva, and C. Diot, "POP-level and access-link-level traffic dynamics in a Tier-1 POP," in *ACM IMW*, 2001.

[4] B. Bollobas, *Random Graphs*, 2nd ed. Cambridge U. Press, 2001.

[5] R. Diestel, *Graph Theory*, 3rd ed. Springer, 2005.

[6] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, "The case for separating routing from routers," in *Proc. ACM SIGCOMM FDNA workshop*, 2004.

[7] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. IEEE INFOCOM*, 2000.

[8] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4D approach to network control and management," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, October 2005.

[9] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne, "Fast IP network recovery using multiple routing configurations," in *IEEE INFOCOM*, 2006.

[10] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica, "Achieving convergence-free routing using failure-carrying packets," in *Proc. ACM SIGCOMM*, 2007.

[11] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: Existing techniques and new directions," in *ACM SIGCOMM*, 2002.

[12] A. Nucci, S. Bhattacharyya, N. Taft, and C. Diot, "IGP link weight assignment for operational Tier-1 backbones," *IEEE/ACM Transactions on Networking*, vol. 15, no. 4, pp. 789–802, August 2007.

[13] Y. Ohara, S. Imahori, and R. V. Meter, "MARA: Maximum alternative routing algorithm," in *Proc. IEEE INFOCOM*, 2009.

[14] H. Peterson, S. Sen, J. Chandrashekar, L. Gao, R. Guerin, and Z.-L. Zhang, "Message-efficient dissemination for loop-free centralized routing," *ACM SIGCOMM Comp. Comm. Rev.*, vol. 38, no. 3, July 2008.

[15] S. Ray, R. Guerin, K.-W. Kwong, and R. Sofia, "Always acyclic distributed path computation," *To appear in IEEE/ACM Transactions on Networking*, 2009.

[16] C. Reichert, Y. Glickmann, and T. Magedanz, "Two routing algorithms for failure protection in IP networks," in *Proc. ISCC*, 2005.

[17] C. Reichert and T. Magedanz, "Topology requirements for resilient IP networks," in *Proc. 12th GI/ITG Conf. on Meas., Mod. & Eval. of Comp. & Comm. Sys*, Dresden, Germany, September 2004.

[18] Rocketfuel project. [Online]. Available: www.cs.washington.edu/research/networking/rocketfuel

[19] G. Schollmeier, J. Charzinski, A. Kirstädter, C. Reichert, K. Schrodi, Y. Glickman, and C. Winkler, "Improving the resilience in IP networks," in *Proc. HPSR 2003*, Torino, Italy, June 2003.

[20] M. Shand and S. Bryant, "IP fast reroute framework," Internet Draft, June 2009, (work in progress).

[21] M. Shand, S. Bryant, and S. Previdi, "IP fast reroute using Not-Via addresses," Internet draft, July 2009, (work in progress).

[22] H. Yan, D. A. Maltz, T. S. E. Ng, H. Gogineni, H. Zhang, and Z. Cai,

"Tesseract: A 4D network control plane," in *Proc. USENIX Symposium on Networked Systems Design and Implementation*, 2007.

[23] Z. Zhong, S. Nelakuditi, Y. Yu, S. Lee, J. Wang, and C.-N. Chuah, "Failure inferencing based fast rerouting for handling transient link and node failures," in *Proc. IEEE Global Internet*, 2005.

APPENDIX A
PROOF OF THEOREM 4.1

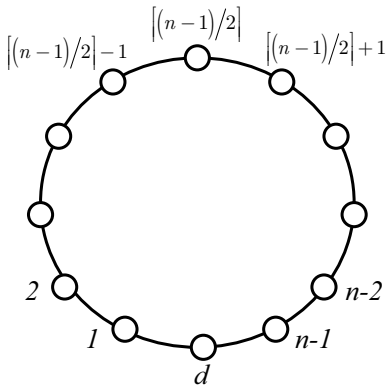


Fig. 4. A labeling used in the proof of Theorem 4.1

Let $n \geq 5$ be the number of nodes in a graph. Let us first consider how to build a protection routing for a destination d. Since the graph has minimum degree at least $\lceil n/2 \rceil$, it has a Hamilton cycle by Theorem 10.1.1 on p. 276 in [5]. We label the nodes as shown in Fig. 4. Note that destination d must have at least $\lceil n/2 \rceil - 2$ link(s) connected to some nodes other than nodes 1 and $n-1$. Suppose that one of the links from destination d lands on node $j \in [2, n-2]$. If $j \leq \lceil (n-1)/2 \rceil$, we build a 3-branch spanning tree rooted at node d as follows where the arrows mean the orientation of the routing: (1) $j-1 \rightarrow j-2 \rightarrow \cdots \rightarrow 1 \rightarrow d$, (2) $\lceil (n-1)/2 \rceil \rightarrow \lceil (n-1)/2 \rceil - 1 \rightarrow \cdots \rightarrow j \rightarrow d$, and (3) $\lceil (n-1)/2 \rceil + 1 \rightarrow \lceil (n-1)/2 \rceil + 2 \rightarrow \cdots \rightarrow n-1 \rightarrow d$. If $j \geq \lceil (n-1)/2 \rceil + 1$, we build a 3-branch spanning tree rooted at node d as follows: (1) $j+1 \rightarrow j+2 \rightarrow \cdots \rightarrow n-1 \rightarrow d$, (2) $\lceil (n-1)/2 \rceil \rightarrow \lceil (n-1)/2 \rceil + 1 \rightarrow \cdots \rightarrow j \rightarrow d$, and (3) $\lceil (n-1)/2 \rceil - 1 \rightarrow \lceil (n-1)/2 \rceil - 2 \rightarrow \cdots \rightarrow 1 \rightarrow d$. Since each node has degree at least $\lceil n/2 \rceil$, it is easy to see that every node has at least one link connected to another tree branch based on the above construction, and hence the routing is protected. We can apply the same technique to find a protection routing for every destination, and as a result, the graph is protectable.

APPENDIX B
PROOF OF THEOREMS 4.2 AND 4.3

We first show that for any given $k \in \mathbb{Z}^+$, there exists a k-link-connected graph that is unprotectable. Our construction is as follows.

Let d be the destination. Let x_i, $i = 1, 2, \ldots, k$, be a node directly connected to destination d. Let $G_i = (V_i, E_i)$ be a fully connected subgraph where $V_i = \{y_{i,1}, y_{i,2}, \ldots, y_{i,2k}\}$. k links are established from node x_i to nodes $y_{i,1}, y_{i,2}, \ldots, y_{i,k}$ for $i = 1, 2, \ldots, k$. Then node z is created so that it connects

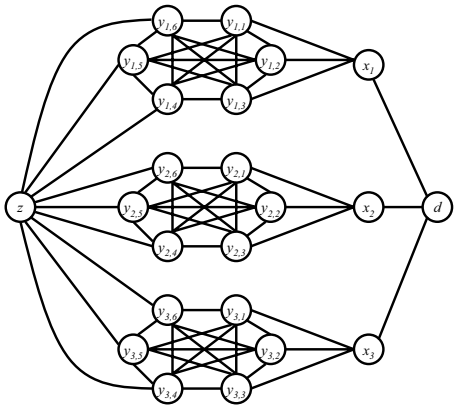


Fig. 5. An example of an unprotected 3-link-connected graph.

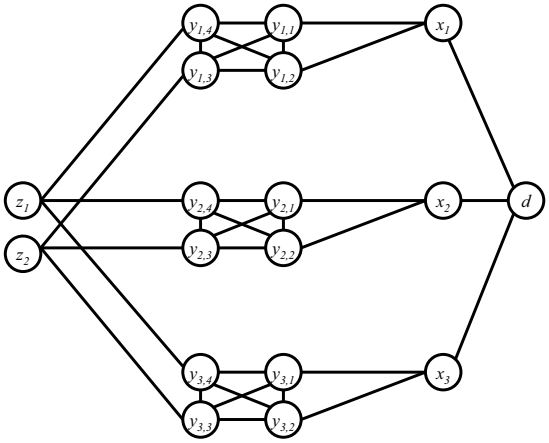


Fig. 6. An example of an unprotected 3-connected graph.

to nodes $y_{i,k+1}, y_{i,k+2}, \ldots, y_{i,2k}$ for $i = 1, 2, \ldots, k$. We use H_k to denote the final graph. The example for H_3 is shown in Fig. 5. It is easy to see that H_k is k-link-connected and 2-node-connected. We can argue that H_k is not protectable as follows. Note that node z has to select at least one PNH. Without loss of generality, suppose that its PNH is a node in G_1. Since the graph is 2-node-connected and when the link (x_1, d) is failed, the traffic originated at node x_1 has to use one of the nodes $x_2, x_3, \ldots, x_k$ to reach the destination, and the traffic needs to go through node z. As a result, a loop is formed under the destination-based routing.

To prove that there exists a 3-node-connected graph that is unprotectable, we give a counter-example as shown in Fig. 6 where the destination is denoted by d. Suppose that a protection routing for destination d exists. First note that if link (x_1, d) is failed, a protection routing needs one PNH from either node $y_{1,3}$ or $y_{1,4}$ to either node z_1 or z_2. It is the same if link (x_2, d) or (x_3, d) is failed. Therefore, a protection routing requires 3 PNHs pointing to nodes z_1 and z_2. Additionally, nodes z_1 and z_2 totally need 2 outgoing PNHs and 2 SNHs in order to protect themselves. As a result, $3 + 4 = 7$ links are required from z_1 and z_2 but they only have 6 links, and hence a contradiction happens.

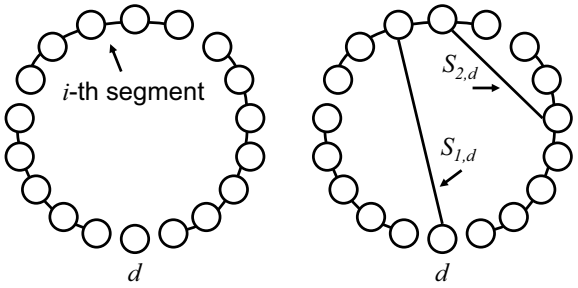


Fig. 7. Construction of a protection routing in a random graph.

APPENDIX C
PROOF OF THEOREM 4.4

In the following, we assume that nodes in a random graph $G(n,p)$ are labeled by $\{1, 2, \ldots, n\}$, and we use the notation $a_n \sim b_n$ to represent $\lim_{n\to\infty} a_n/b_n = 1$. A property A of $G(n,p)$ is said to be existence asymptotically almost surely (a.a.s.) if $P(A) \to 1$ as $n \to \infty$. Unless otherwise specified, $\log$ denotes the natural logarithm.

The proof is based on a constructive method to find a protection routing for every destination in a random graph. The main difficulty in the proof is how to construct routings for every destination such that each node is protected in the sense that they have a suitable SNH to reroute traffic when their PNHs are failed. To overcome this difficulty, we use a two-round exposure (*e.g.*, see [4]) to construct a protection routing for every destination. The two-round exposure means that if $G_1(n,p_1)$ and $G_2(n,p_2)$ are generated independently on the same node set, then $G_1(n,p_1) \cup G_2(n,p_2)$ is just distributed as $G(n,p)$ where $p = p_1 + p_2 - p_1 p_2$, and any multiple links between nodes in $G_1(n,p_1) \cup G_2(n,p_2)$ are just replaced by a single link. Based on this fact, our method is to generate two independent, "suitable" random graphs, $G_1(n,p_1)$ and $G_2(n,p_2)$, such that a protection routing can be constructed accordingly.

Let $G_1 \in G_1(n,p_1)$ and $p_1 = (1+\varepsilon_1)\log n/n$ where $\varepsilon_1 > 0$ is any constant. By a well-known result (*e.g.*, see [4]), a.a.s. G_1 has a Hamilton cycle. In the following, we let $p_2 = a \log n/n$ where $a > 0$ is some constant determined later, and generate an independent graph $G_2 \in G_2(n,p_2)$ such that a protection routing can be constructed based on a given Hamilton cycle in G_1.

Suppose that a Hamilton cycle exists in G_1. For a destination d, we divide the Hamilton cycle into $m \geq 2$ segments (exclude node d) where the value of $m \in \mathbb{N}$ is determined later, and each i-th segment, $i = 1, 2, \ldots, m$, has g_i nodes where

$$\frac{n-1}{m} - 1 \leq g_i \leq \frac{n-1}{m} + 1 \ . \tag{5}$$

An example is shown on the left hand side of Fig. 7. Now, we introduce the two additional structures in G_2:

- $S_{1,d}$: Destination d has at least one link connected to a node in each segment.
- $S_{2,d}$: Each node in each segment has at least one link connected to a node in another segment.

The examples of $S_{1,d}$ and $S_{2,d}$ are shown on the right hand side of Fig. 7. If $S_{1,d}$ exists, we can construct a m-branching routing tree rooted at destination d. On the other hand, if $S_{2,d}$ exists, it implies that each node has a neighbor in another segment for the protection of a failure situation. As a result, a protection routing destined to d can be constructed accordingly.

Let $X_{i,d} = 1$ if destination d has no link connected to i-th segment, and $X_{i,d} = 0$ otherwise. Let $X_d = \sum_{i=1}^{m} X_{i,d}$, and note that if $X_d = 0$, then $S_{1,d}$ exists. Because $E(X_{i,d}) = P(X_{i,d} = 1)$ and hence

$$\begin{aligned} E(X_d) &= \sum\nolimits_{i=1}^{m} E(X_{i,d}) && (6)\\ &= \sum\nolimits_{i=1}^{m} (1-p_2)^{g_i} && (7)\\ &\sim m(1-p_2)^{n/m} && (8)\\ &\leq m\exp\left(-p_2\frac{n}{m}\right) && (9)\end{aligned}$$

Let $Y_d = \sum_{i=1, i\neq d}^{n} Y_{i,d}$ where $Y_{i,d} = 1$ if node $i \neq d$ has no link connected to a node in another segment, and $Y_{i,d} = 0$ otherwise. If $Y_d = 0$, then $S_{2,d}$ exists. First note that the total number of nodes in any $m-1$ segments is at least $\{(n-1)/m - 1\}\{m-1\}$, so the probability that a node $i \neq d$ has no link connected to a node in another segment is at most $(1-p_2)^{\{(n-1)/m-1\}\{m-1\}}$. Because $E(Y_{i,d}) = P(Y_{i,d} = 1)$ and hence

$$\begin{aligned} E(Y_d) &= \sum\nolimits_{i=1, i\neq d}^{n} E(Y_{i,d}) && (10)\\ &\leq n(1-p_2)^{\{(n-1)/m-1\}\{m-1\}} && (11)\\ &\leq n\exp\{-p_2(n-1)(1-1/m) + p_2 m\} && (12)\\ &\sim n\exp\{-p_2 n(1-1/m)\} && (13)\end{aligned}$$

Now, we apply the above construction, $S_{1,d}$ and $S_{2,d}$, for all $d \in V$. In G_2, let $Z_d = 1$ if at least $S_{1,d}$ or $S_{2,d}$ does not hold, and $Z_d = 0$ otherwise. Define $Z = \sum_{d=1}^{n} Z_d$ which counts the number of destinations without a protection routing based on the above construction. Note that by Markov inequality,

$$\begin{aligned} P(Z_d = 1) &= P(X_d + Y_d \geq 1) && (14)\\ &\leq E(X_d + Y_d) && (15)\\ &= E(X_d) + E(Y_d) && (16)\end{aligned}$$

and then, by using Markov inequality, Eqs. 9 and 13, we have

$$\begin{aligned} P(Z = 0) &= 1 - P(Z \geq 1) && (17)\\ &\geq 1 - E(Z) && (18)\\ &= 1 - \sum\nolimits_{d=1}^{n} E(Z_d) && (19)\\ &= 1 - nP(Z_d = 1) && (20)\\ &\geq 1 - nE(X_d) - nE(Y_d) && (21)\\ &\geq 1 - nm\exp\{-p_2 n/m\} \\ &\quad -n^2\exp\{-p_2 n(1-1/m)\}. && (22)\end{aligned}$$

To ensure that every destination has a protection routing, *i.e.*, $P(Z = 0) \to 1$ as $n \to \infty$, Eq. 22 needs the following two conditions as $n \to \infty$:

$$\begin{aligned} \log n - p_2 n/m \to -\infty &\quad\Rightarrow\quad a > m && (23)\\ 2\log n - p_2 n(1-1/m) \to -\infty &\quad\Rightarrow\quad a > \frac{2}{1-1/m} && (24)\end{aligned}$$

Because the value of a should be kept as small as possible, we pick $m = 3$ and hence $a = 3 + \varepsilon_2$ where $\varepsilon_2 > 0$ is any constant. As a result, G_2 should be generated using $p_2 = (3+\varepsilon_2)\log n/n$.

Therefore, by using the two-round exposure, a.a.s. $G(n,p) = G_1(n,p_1) \cup G_2(n,p_2)$, which $p = (4+\varepsilon)\log n/n$ and $\varepsilon > 0$ is any constant, has a protection routing for all the n destinations.

APPENDIX D
PROOF OF THEOREM 4.5

This section shows that the PR problem is NP-complete by a reduction from the 3SAT problem. In the following, we use d to denote the destination. First note that one can verify whether a given routing destined to node d is protected in a polynomial time. Thus, the PR problem is in NP.

Assume the 3SAT problem consists of n boolean variables, $u_1, u_2, \ldots, u_n$, in which their complements are denoted by $\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_n$ respectively. Let $U = \{u_1, \bar{u}_1, \ldots, u_n, \bar{u}_n\}$. The boolean equation of the 3SAT problem consists of m clauses, *i.e.*,

$$B = C_1 C_2 \cdots C_m \quad (25)$$

where $C_j = x_j + y_j + z_j$ and $x_j, y_j, z_j \in U$ for all $j = 1, 2, \ldots, m$. We build a graph G' to represent B, and show that the 3SAT problem has a solution if and only if G' has a protection routing destined to node d. The construction of G' involves a combination of the components called squares, switches and connectors, and they are detailed in the following.

For each clause $C_j, j = 1, 2, \ldots, m$, we use the four-node mesh topology called C_j-square, as shown in Fig. 8, for its representation where each node has a link going outside. One of those links is connected to destination d and other three links are labeled as x_j, y_j and z_j respectively.

Observation D.1: C_j-square is protected if it has at least two outgoing PNHs.

The function of C_j-square is to mimic the situation that clause C_j is true or not. In order to make clause C_j to be

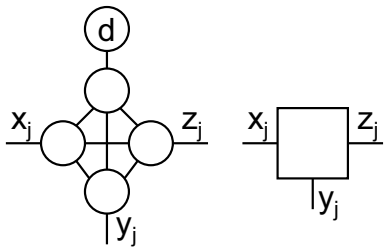


Fig. 8. Structure of C_j-square and its schematic representation.

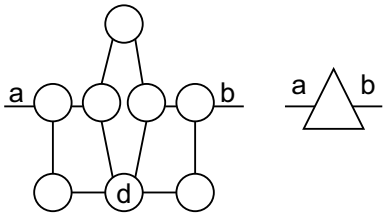


Fig. 9. Structure of u_i-switch and its schematic representation.

true, at least one of its boolean variables must be true which corresponds to an outgoing PNH from link x_j, y_j or z_j.

Next we introduce a graph component called u_i-switch where $i = 1, 2, \ldots, n$. The structure of u_i-switch is shown in Fig. 9 where links a and b are outgoing from the switch. The function of u_i-switch is to represent boolean variable u_i.

Observation D.2: To make u_i-switch protected, the routing must result in a configuration that at least link a or b is an outgoing PNH.

The key point of Observation D.2 is that, by introducing this constraint, we can map between the boolean assignment of u_i and the routing configuration in u_i-switch.

Next we introduce connector which is structured as Fig. 10. In particular, the connector has three outgoing links e, f and g. We have the following two observations about the connector.

Observation D.3: If link e is an incoming PNH to a connector, then in order to protect the connector, we need a routing that link f can not be an incoming PNH to the connector and

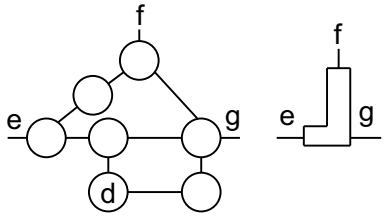


Fig. 10. Structure of connector and its schematic representation.

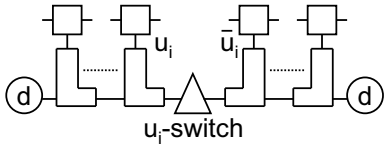


Fig. 11. An example of a graph formed by squares, switch and connectors for boolean variables u_i and $\bar{u}_i$.

link g must be an outgoing PNH from the connector.

Observation D.4: If link f is an incoming PNH to a connector, then we can find a routing to protect the connector such that link e is an outgoing PNH from the connector and link g is not necessary to be used.

The key point of Observations D.3 and D.4 is that link e and link f can be used to control the direction of a protection routing such that we can impose a certain routing pattern for the NPC reduction as discussed shortly.

Now we are ready to use squares, switches and connectors to build G' for the NPC reduction. For each C_j-square, there are three links corresponding to x_j, y_j and z_j in C_j clause. For each of them, it is connected to a connector under the corresponding switch of that boolean variable. Fig. 11 shows an example where on the left hand side all the squares consist of u_i boolean variable and on the right hand side all the squares consist of $\bar{u}_i$ boolean variable. Therefore, given the 3SAT problem, the corresponding G' can be constructed accordingly in a polynomial time.

Suppose that a 3SAT solution exists, *i.e.*, each clause must have at least one boolean variable resulting in true value. Now we find a protection routing using the following method. In each C_j-square, we assign the link corresponding to a true boolean variable as an outgoing PNH. If the clause only has one true boolean variable, we can assign the link connected to destination d as an outgoing PNH. As a result, each C_j-square has at least two outgoing PNHs. An example is given below.

Consider u_i-switch, $i = 1, 2, \ldots, n$, and suppose that u_i is true. We assign the routing as shown in Fig. 12 (by Observations D.2, D.3, and D.4) where the arrows indicate the orientation of the routing. It is similar if $\bar{u}_i$ is true (*i.e.*, u_i is false). In this case, the orientation of the routing is from right to left as opposed to Fig. 12. As a result, a protection routing can be found accordingly.

Now suppose that a protection routing exists on G'. First, we have the following observation.

Observation D.5: If a protection routing exists, the situation shown in Fig. 13 can not happen. In other words, it is impossible that links u_i and $\bar{u}_i$ from their squares point their PNHs to their connectors.

Observation D.5 can be explained as follows. If a protection routing exists, the switch must have at least one outgoing PNH

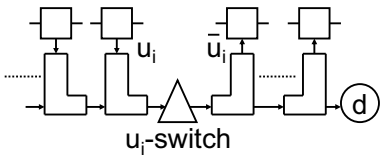


Fig. 12. An example of a routing configuration if u_i is true.

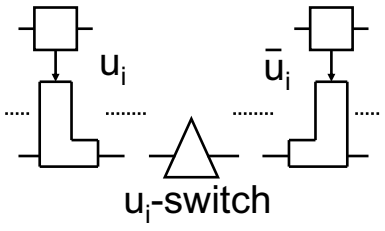


Fig. 13. An impossible configuration under u_i-switch.

(by Observation D.2) which points to link e of a connector. By Observation D.3, link f of that connector can not be an incoming PNH from a square, and link g of that connector must be an outgoing PNH. Since the connectors are connected in serial, the situation shown in Fig. 13 can not happen.

Given a protection routing, we use the following method to find a 3SAT solution. We look at each C_j-square, $j = 1, 2, \ldots, m$, and note that there must exist an outgoing PNH via link x_j, y_j or z_j due to Observation D.1. For example, if link x_j is an outgoing PNH, then we set x_j to be true. By Observation D.5, x_j can be assigned true or false consistently. After looking at all C_j-squares, if there are any remaining, unassigned boolean variables, they can be assigned true or false arbitrarily. As a result, each clause is satisfied and hence $B = C_1 C_2 \cdots C_m$ is satisfied.

APPENDIX E
HEURISTIC DETAILS

This section describes the details of our proposed two-phase heuristic and SNH assignment algorithm, as well as its computational complexity.

The main loop of our two-phase heuristic is shown in Algorithm 1. In Phase 1 (from lines 1 to 3), the main focus is to find a R_d to minimize F_d for each $d \in V$. In Phase 2 (line 4), the main focus is to minimize Φ by further modifying the routings obtained from Phase 1 without hurting protectability.

The details of Phase 1 are shown in Algorithm 2. F_d in the pseudocode denotes the cost function value based on the current R_d. Phase 1 starts by extracting a tree from R_d^{opt} (when multiple next-hops are available, one is randomly selected), and this operation is denoted by $T\left(R_d^{opt}\right)$. The main loop is from lines 5 to 17. Its function is to locally change the PNH of each node to try to improve F_d. This method is inspired by those approximation algorithms for the 3SAT problem. After visiting all the nodes and no improvement in F_d is achieved, the diversification is executed in line 22 and generates a new random shortest-path tree such that the search can restart from it and explore a different region of the solution space. Phase 1 stops if there is no improvement in F_d after P diversification rounds.

The details of Phase 2 are shown in Algorithm 3. The routing trees $R_d^*, \forall d \in V$, of Phase 1 are used as inputs to Phase 2. In the pseudocode, Φ, $\varphi_i, \forall i \in V$, and $\Omega_d, \forall d \in V$, denote their respective values based on the current $R_d^*, \forall d \in V$. The main loop is from lines 2 to 17, and tries to assign multiple PNHs to each node to better load balance over the network without hurting protectability. The order of visiting nodes is in decreasing order of its congestion defined by $\varphi_i, i \in V$ (line 3). Phase 2 stops if no further PNH assignment is allowed to reduce Φ.

After Phase 2, we use Algorithm 4 to assign SNHs to nodes (if possible) for any single component failure in routings $R_d, \forall d \in V$. This algorithm can systematically explore backup paths involving multiple SNHs to improve protectability and prevent from creating loops. The algorithm continues until no SNH assignment satisfying conditions (H1) and (H2) is found. A factor affecting computational complexity is the number of iterations required before no further SNH assignment improves protectability. This number depends in part on the order in which nodes in A_f are visited (line 8 of Algorithm 4). There are, however, no simple ordering that yields a consistently better outcome.

In our heuristic, complexity is dominated by the evaluation of the costs Ω_d and Φ. A change in R_d can affect the protectability of many nodes and hence the cost Ω_d needs to be re-evaluated. A node is protected if it has at least two independent PNHs such that a rerouting option is available when a single component failure affects its PNHs. Checking this condition can be done using a modified breadth-first search, and its time complexity is $O\left(|V| + |E|\right)$ (resp. $O\left(|V|\left(|V| + |E|\right)\right)$) in sparse graphs (resp. dense graphs). To compute Φ, we need to evaluate the load of each link. The load of the links contributed by R_d (*i.e.*, traffic destined to d) can be computed in $O\left(|V| + |E|\right)$ time. Therefore, the total time to evaluate Φ is $O\left(|V|\left(|V| + |E|\right)\right)$. In Algorithm 4 to assign SNHs in R_d, checking the conditions (H1) and (H2) can be easily done using a modified breath-first search which takes $O\left(|V| + |E|\right)$ time.

Algorithm 1: A proposed two-phase heuristic

Input: Network topology $G = (V, E)$ and traffic matrix M
Result: R_d^{**} for each $d \in V$
1 Phase 1: **foreach** *destination* $d \in V$ **do**
2 | $(R_d^*, \Omega_d^*) \leftarrow$ Greedy Search on d
3 **end**
4 Phase 2: $\left(R_d^{**}\right)_{d \in V} \leftarrow$ Load-Balancing $\left(\left(R_d^*, \Omega_d^*\right)_{d \in V}, M\right)$

Algorithm 2: Greedy Search

Input: $G=(V,E)$, destination d and R_d^{opt} link weights
Result: R_d^* and $F_d^*=\langle \Omega_d^*, \Gamma_d^* \rangle$

$R_d \leftarrow T\left(R_d^{opt}\right)$
$R_d^* \leftarrow R_d$, $F_d^* \leftarrow F_d$
repeat
  $F_d^{\text{temp}} \leftarrow F_d$
  foreach *node* $i \in V \setminus \{d\}$ **do**
    foreach $j \in N_G(i)$, *j is not upstream of i in R_d and link (i,j) is not in R_d* **do**
      Let node k denote the current PNH of node i
      Remove link (i,k) from R_d
      Add link (i,j) to R_d
      if $F_d < F_d^{temp}$ **then**
        $F_d^{\text{temp}} \leftarrow F_d$
      else
        Remove link (i,j) from R_d
        Add link (i,k) to R_d
      end
    end
  end
  if $F_d < F_d^*$ **then**
    $F_d^* \leftarrow F_d$, $R_d^* \leftarrow R_d$
  end
  if F_d^{temp} *has no improvement* **then**
    Diversification on R_d
  end
until F_d^* *has no improvement after P diversifications*

Algorithm 3: Load-Balancing

Input: $G=(V,E)$, traffic matrix M, $(R_d^*, \Omega_d^*)_{d\in V}$
Result: R_d^{**} for each $d \in V$

$\Phi^* \leftarrow \Phi$
repeat
  Compute $\varphi_i = \sum_{(i,j)\in E} \Phi_{(i,j)}$ for each node i
  Sort each φ_i such that $\varphi_{\sigma(1)} \geq \varphi_{\sigma(2)} \geq \ldots \geq \varphi_{\sigma(n)}$
  for $i=1$ **to** n **do**
    foreach $d \neq \sigma(i)$ **do**
      foreach *node $j \in N_G(\sigma(i))$, j is not upstream of $\sigma(i)$ in R_d^* and link $(\sigma(i),j)$ is not in R_d^** **do**
        Add link $(\sigma(i),j)$ to R_d^*
        if $\Omega_d > \Omega_d^*$ *or* $\Phi > \Phi^*$ **then**
          Remove link $(\sigma(i),j)$ from R_d^*.
        else
          $\Phi^* \leftarrow \Phi$
        end
      end
    end
  end
until Φ^* *can not be further reduced*
$R_d^{**} \leftarrow R_d^*$ for each $d \in V$

Algorithm 4: SNH assignment algorithm for R_d

Input: R_d
Result: SNH assignment S_d^f for every single failure f

foreach *node i and failure f* **do**
  $S_d^f(i) \leftarrow \emptyset$
end
repeat
  foreach *failure f* **do**
    Let A_f denote the set of nodes whose PNHs are affected by failure f
    Let $H_d^f = \left(V^f, E_d^f \bigcup_{i\in V, S_d^f(i)\neq\emptyset} (i, S_d^f(i))\right)$
    foreach *node $i \in A_f$* **do**
      if $S_d^f(i) = \emptyset$ **then**
        foreach $k \in N_{G^f}(i)$ **do**
          Add link (i,k) to H_d^f
          if H_d^f *satisfies conditions (H1) and (H2)* **then**
            $S_d^f(i) \leftarrow k$
            Break
          else
            Remove link (i,k) from H_d^f
          end
        end
      end
    end
  end
until *no further SNH assignment is allowed*