On critical service recovery after massive network failures

N. Bartolini, S. Ciavarella, T. La Porta, and S. Silvestri

***Abstract*—This paper addresses the problem of efficiently restoring sufficient resources in a communications network to support the demand of mission critical services after a large scale disruption. We give a formulation of the problem as an MILP and show that it is NP-hard. We propose a polynomial time heuristic, called Iterative Split and Prune (ISP) that decomposes the original problem recursively into smaller problems, until it determines the set of network components to be restored. ISP's decisions are guided by the use of a new notion of demand based centrality of nodes. We performed extensive simulations by varying the topologies, the demand intensity, the number of critical services, and the disruption model. Compared to several greedy approaches ISP performs better in terms of total cost of repaired components, and does not result in any demand loss. It performs very close to the optimal when the demand is low with respect to the supply network capacities, thanks to the ability of the algorithm to maximize sharing of repaired resources.**

***Index Terms*—Network recovery, flow restoration, massive network disruption.**

I. Introduction

NATURAL disasters or intentional attacks can severely disrupt critical infrastructures such as communication, power, and emergency control networks at a large scale.

Because our society heavily depends on communication networks to support mission critical services, especially in times of emergency, it is important that such infrastructures be repaired quickly, at least to the point where mission critical services are restored.

In 2011, the "great east Japan earthquake" hit a large part of the north-east of Japan. The earthquake was just the start of a widespread disaster, which also included a huge tsunami and the nuclear failure at Fukushima. The tsunami destroyed most terrestrial communication infrastructures including many of the wired communication networks and emergency municipal radio communication systems [1], [2]. The communication outage consequent to the disaster hampered the assessment of residents' safety. It also precluded efficient rescue operations by government and public organizations, such as distribution of medical aid and emergency supplies. The restoration of the communication infrastructure and its related services took months, a time that is far from meeting the requirements of critical services or normal local communications of people living in the affected areas.

N. Bartolini and S. Ciavarella are with the Department of Computer Science, Sapienza University of Rome, Italy. E-mail: {bartolini,ciavarella}@di.uniroma1.it

T. La Porta is with the Department of Computer Science and Engineering, Pennsylvania State University, USA. E-mail: tlp@cse.psu.edu

S. Silvestri is with the Department of Computer Science Missouri University of Science and Technology, USA. E-mail: silvestris@mst.edu

In the aftermath of a disaster it is a government priority to let local leadership and the business community work together to develop a recovery plan based on a common knowledge of damages and estimate of the critical service demand.

We focus on the *communication network* and the mission critical applications it supports. The latter represents critical services such as communication between government offices, police stations, fire stations, power plants, gas-duct control centers and hospitals, that rely on the communication network for control and cooperation. These services are critical for first responders and typically show increased rate of requests as a consequence of the occurred incidents [3].

We address the problem of *sufficiently recovering* the communication network infrastructure so that it may support mission critical applications in the shortest time and with minimum interventions.

In this paper we give an original formulation of the recovery problem in terms of mixed integer linear programming (MILP). The problem looks for the best strategy that recovers the damaged infrastructure and deploys new links and nodes in order to minimize the cost of the recovery actions under the constraints on network capacity and demand flows satisfaction.

We show that the problem is NP-hard and propose a novel heuristic called *Iterative Split and Prune* (ISP) to recover the network efficiently in polynomial time with a solution close to the optimal. ISP is based on a new metric called *demand based centrality*, specifically designed to measure the importance of a node with respect to multiple demand flows of interest. ISP makes use of this metric to determine the most important nodes to be repaired. In particular, ISP iteratively selects the node with the highest centrality, repairs it if damaged, and *splits* some demand flows to force them to pass through the selected node. This way, ISP minimizes the repairs by concentrating flows towards the areas of the network already repaired. Additionally, it *prunes* the demand flows which can be satisfied by the currently repaired network.

ISP returns both a recovery strategy and a routing solution for the demand flows.

We also consider other greedy heuristics as benchmarks. We compare the performance of ISP and the other heuristics against the optimal solution under a variety of scenarios, including both real and synthetic network topologies, geographically correlated failures, as well as different demand requirements and heterogeneous cost setting.

Results show that ISP always outperforms the other heuristics in the number of repairs. Moreover, we show scenarios in which the execution time of ISP in complex scenarios is in the order of 5 minutes, whereas the optimal solution takes more than 27 hours.

The original contribution of the paper is the following:

- We formulate the recovery problem, hereby called MinR, as an MILP and show its NP-hardness.
- We introduce a metric of demand based centrality, to measure the importance of a node of an instance of MinR.
- We propose a heuristic called ISP, which uses the new centrality metric to guide recovery decisions.
- We propose several greedy heuristic variants and a shortest path heuristic as baseline solutions.
- We evaluate the proposed solutions through simulations under a wide variety of scenarios. Results show that ISP performs close to the optimal, while other heuristics incur a much higher cost to accommodate the demand flows.

II. Related work

As most of today's critical infrastructures and services rely on the support of the communication network, the problem of network recovery after major disruptions is receiving increasing attention. Numerous works address the case of sparse, or small scale failures through the provision of alternative paths, provided either proactively, as in the work of Todimala et. al [4], Hansen et al. [5], [6], [7], Medard et al. [8], or reactively, as in the work of Zheng et al. [9]. Suchara et. al [10] jointly address the problems of recovery and traffic engineering, to minimize congestion after a failure. A taxonomy of previous work on the design of survivable networks providing pre-planned routing recovery plans is given by Kerivin et al. [11] and, with more emphasis on optical communication networks, by Habib et al. [12]. Another work by Gardner et al. [13] propose a pre-planned rerouting approach in the case of geographic scale disruptions.

Neumayer et al. [14] characterize the minimum number of failures that would cause the disconnection of given terminals.

Our paper addresses the case of massive failures from a different perspective and tackles the problem of repairing the network elements so that at least the critical services can be provided within given quality of service requirements.

Yang et al. [15] addressed a related problem. They formulated the problem of repairing links as an MILP, where repair interventions are performed in a way that optimizes a weighted throughput function. They proposed a Knapsack based heuristic in which links are repaired according to priorities based on the values of the shadow prices of the link capacity constraints of the MILP problem. A similar approach, based on a shadow prices technique, is proposed by Wan et al. [16]. In this work, the authors study the impact of recovery actions in terms of improved throughput over time. Their work aims at formulating a schedule of repair interventions under limited daily budget, so as to optimize the total accumulative throughput over time. The authors proposed a greedy heuristic for solving the problem in multiple stages by analyzing the shadow prices of the related optimization problem. Both the works [15], [16] aim at prioritizing repair of the edges that have the highest potential for contributing to the objective function. They are both based on the assumption that broken links have a non null heterogeneous minimum bandwidth. This is a necessary assumption to prevent situations in which the shadow prices of all links are zero, which is likely to happen in the case of a massive disruption, where many links and nodes are broken. In such a case, in fact, the performance of the algorithm would be related to the particular tie breaking rule to be adopted when no single link repair intervention would provide an immediate improvement in terms of routed flow.

Unlike these works which aim at optimizing throughput over time, we aim at optimizing the recovery costs under constraints on quality of service. Our algorithm considers the more realistic case in which both links and nodes can be disrupted. Moreover, our algorithm also produces a routing solution that guarantees that the demand flows are actually routed through the repaired nodes and links.

Some works focus on the *rent or buy multi-commodity problem*, which aims at installing possibly unlimited capacities on the edges of a network so that a prescribed amount of flow can be routed between several pairs of terminals. Unlike our problem, the rent or buy problem assumes that each edge can obtain unlimited capacity at a given cost. The works by Kumar et al. [17] and Fleischer et al. [18] address this problem and propose polynomial time heuristics with a given approximation of the optimal solution.

Other works address the problem of service restoration in the case of heterogeneous non-telecommunication networks. Among these, the work of Lee et al. [19] addresses the problem of restoring service in an interconnected network by creating new links. They propose a formulation of the problem in terms of a high complexity optimization model. Other works [20], [21] address the problem of recovery beyond the field of telecommunications with solutions tailored to the specific type of network being considered.

The present paper extends a previous conference version [22] with proofs and new experiments.

III. The Network Recovery Problem

In this section we give an original formalization of the problem of minimizing the cost to repair broken nodes and links so as to restore the necessary network capacity to meet a given demand. We call it the *Minimum Recovery* (MinR) problem and we formulate it as a mixed integer linear optimization problem. The formulation refers to the notation and nomenclature given in Table I.

We model the communication network as an undirected graph $G = (V, E)$, called the *supply graph*, where V and E represent nodes and links of the network, respectively. Each edge $(i, j) \in E$ has capacity c_{ij}. We also model the critical service demand as a *demand* graph $H = (V_{\text{H}}, E_{\text{H}})$, where $V_{\text{H}} \subseteq V$, and $E_{\text{H}} \subseteq V_{\text{H}} \times V_{\text{H}}$. Each pair $(s_h, t_h) \in E_{\text{H}}$ has a source s_h, a destination t_h and an associated demand flow d_{s_h,t_h}. For sake of simplicity, we write $h \in E_{\text{H}}$, when $(s_h, t_h) \in E_{\text{H}}$, and we shortly use the notation d_h for d_{s_h,t_h} when the context allows. In order to model the network failure, we define the sets $V_{\text{B}} \subseteq V$ and $E_{\text{B}} \subseteq E$ of damaged vertices and edges, respectively. We denote with k_i^v the cost of repairing vertex $i \in E_{\text{B}}$ and with k_{ij}^e the cost of repairing

Notations	Descriptions
$G = (V, E)$	supply graph
$G^{(n)} = (V^{(n)}, E^{(n)})$	supply graph at iteration n
$H = (V_{\text{H}}, E_{\text{H}})$	demand graph
$H^{(n)} = (V_{\text{H}}^{(n)}, E_{\text{H}}^{(n)})$	demand graph at iteration n
c_{ij}	capacity of edge $(i, j) \in V$
$d_h = d_{s_h,t_h}$	demand flow of edge $(s_h, t_h) \in E_H$
$c_{ij}^{(n)}$, $d_{s_h t_h}^{(n)}$	value of c_{ij} and d_h at the n-th iteration
$V_{\text{B}} \subseteq V$ and $E_{\text{B}} \subseteq E$	broken vertices and edges
$V_{\text{B}}^{(n)}$ and $E_{\text{B}}^{(n)}$	V_{B} and E_{B} at iteration n
$h \in E_{\text{H}}$,	demand pair $(s_h, t_h) \in E_{\text{H}}$
k_i^v, k_{ij}^e	cost of vertex i and edge (i, j)
f_{ij}^h	quantity of flow h from i to j
δ_{ij}	decision to use edge $(i, j) \in E$,
δ_i	decision to use vertex $i \in V$
Δ	maximum degree of the network
b_i^h	flow h generated at node i
$\ell(p)$, $l(e_i)$	length of path p, length of edge e_i
$c(p)$	capacity of path p: $\min_{e \in p} c_e$
$\mathscr{P}(i, j)$	paths in G between i and j
$\mathscr{P}^*(i, j)$	shortest paths necessary to route d_{ij}
$\mathscr{P}_{ij}^*\|_v$	set of paths in $\mathscr{P}_{ij}^*$ that include v
$c_d(v)$	demand based centrality, see equation (3)
$v_{\text{BC}}^{(n)}$	node with highest centrality at iteration n
$\mathcal{C}^{(n)}(v_{\text{BC}}^{(n)}) \subseteq E_{\text{H}}^{(n)}$	demand pairs that contributed to the centrality of $v_{\text{BC}}^{(n)}$, updated at iteration n
$R(n)$	set of repairs, updated at iteration n

TABLE I
NOMENCLATURE AND NOTATION

the edge $(i, j) \in E_{\text{B}}$[1]. The recovery costs are heterogeneous and dependent on the location and on the technology in use.

We introduce the decision variables $f_{ij}^h \in \mathbb{R}$, with $f_{ij}^h \geq 0$, to represent the fraction of the demand flow h that will be routed through the link $(i, j) \in E$, going from vertex i to vertex j. We also define the binary variables δ_{ij} and δ_i. The variable δ_{ij} represents the decision to use link $(i, j) \in E$, therefore $\delta_{ij} = 1$ if link (i, j) is used, and $\delta_{ij} = 0$ otherwise. If the link $(i, j) \in E_{\text{B}}$, the decision to use this link implies that it must be recovered. Similarly, δ_i represents the binary decision to use the node $i \in V$, which has to be recovered if it is broken, that is if $i \in V_{\text{B}}$.

The objective of the MinR, expressed in Equation 1(a), is the minimization of the cost of repairing the only broken elements (vertices and edges) that are used.

The capacity constraint of Equation 1(b) implies that the total amount of flow traversing the edge (i, j) in both directions does not exceed the maximum capacity of the link.

Notice that if an edge (i, j) is used, the corresponding endpoints i and j are also used, which implies that $\delta_i \geq \delta_{ij}$, $\forall i, j \in V$. To express this constraint in a compact form, with fewer equations, we consider that the degree of each vertex is lower than or equal to the maximum degree Δ of the network. Therefore the relationship between δ_i and δ_{ij} can be expressed by the constraint given by Equation 1(c).

We consider a flow balance constraint, in the form expressed by Equation 1(d). In this equation $b_i^h = d_h$ if $i = s_h$, $b_i^h = -d_h$ if $i = t_h$, and $b_i^h = 0$ otherwise.

[1]Notice that this model can also be *adopted as is* to support decisions to replace broken links with new links of higher capacity, or to deploy and connect new nodes, by formulating a related decision space. These additional choices may be considered in the model as parts of the sets E_{B} and V_{B} and included in the correspondent supply graph G.

Finally, Equations 1(e) and Equation 1(f) give the domains of the decision variables f_{ij}^h and δ_{ij}, δ_i.

The MinR problem is the following:

$$
\begin{array}{lll}
\min \sum_{(i,j) \in E_{\text{B}}} k_{ij}^e \delta_{ij} + \sum_{i \in V_{\text{B}}} k_i^v \delta_i & & (a) \\
c_{ij} \cdot \delta_{ij} \geq \sum_{h=1}^{|E_{\text{H}}|} (f_{ij}^h + f_{ji}^h) & \forall (i, j) \in E & (b) \\
\delta_i \cdot \Delta \geq \sum_{j:(i,j) \in E} \delta_{ij} & \forall i \in V & (c) \\
\sum_{j \in V} f_{ij}^h = \sum_{k \in V} f_{ki}^h + b_i^h & \forall (i, h) \in V \times E_{\text{H}} & (d) \\
f_{ij}^h \geq 0 & \forall (i, j) \in E,\ h \in E_{\text{H}} & (e) \\
\delta_i, \delta_{ij} \in \{0, 1\} & \forall i \in V,\ (i, j) \in E & (f)
\end{array}
\tag{1}
$$

Theorem III.1. *The problem MinR is NP-Hard.*

Proof. See the Appendix for the proof of the theorem. □

IV. Iterative Split and Prune

The algorithm ISP (Iterative Split and Prune) is the first polynomial approach to the minimization of the cost of repairs after massive failures. It iteratively selects the best candidate nodes and links for repair, then simplifies the demand by either removing (pruning) or reducing it in smaller segments (split), so as to consider simpler instances of the problem at every iteration. The termination condition is the complete removal of the demand or the achievement of an instance whose demand is routable through the currently working links.

We give the pseudo-code of ISP, while more details on the individual activities can be found in the following sections.

Algorithm Iterative Split and Prune (ISP)

Input: Supply graph G, demand graph H, broken nodes V_{B} and broken edges E_{B}

```
while routability test fails do
    while pruning condition do
        Prune demands satisfying pruning condition;
        Update G and H;
    if there are repairable links then
        Repair broken repairable links;
        Update G and E_B;
    else
        Find best candidate v_BC for split;
        Repair v_BC if broken;
        Find best demand d to split on v_BC;
        Calculate the maximum splittable amount d_x;
        Split amount d_x of demand d on v_BC;
        Update G, H, V_B;
```

A. Routability test

At the basis of the algorithm is the use of flow balance equations and capacity constraints to determine the feasibility of an action or the termination condition. The algorithm terminates whenever there is no demand left, or the current demand can be routed without additional repairs.

For some specific topologies of both supply and demand graphs, as discussed by Schrijver in [23], the question whether a demand can be routed through the links of the supply graph can be answered by verifying the so called *cut condition*, namely whether for every cut the total capacity crossing the cut is not lower than the total demand crossing it. While the cut condition is always necessary to ensure the routability of a set of demand flows through a supply graph, it is not always

sufficient, for example when the graphs G and H admit an *odd p-spindle* as a minor as motivated by Chakrabarty, Fleischer and Weible in [24], or a *bad-k4-pair* as discussed in the already mentioned work by Schijver [23].

The specific instances of graph pairs G and H of a multi-commodity flow problem for which the verification of the cut condition is a necessary and sufficient condition for the routability are called *cut-sufficient* instances. In this work we are *not* assuming cut-sufficiency as we address general graph instances.

Without assuming any structural property of the supply and demand graph, the routability of the demand over the supply graph can be determined by solving the following set of inequalities, to which we will refer under the name of *routability conditions*:

$$\begin{cases} \sum_{h\in E_{\mathrm{H}}}(f_{ij}^h + f_{ji}^h) \le c_{ij} & \forall (i,j)\in E \\ \sum_{j\in V} f_{ij}^h = \sum_{k\in V} f_{ki}^h + b_i^h & \forall (i,h)\in V\times E_{\mathrm{H}} \\ f_{ij}^h \ge 0 & \forall (i,j)\in E, h\in E_{\mathrm{H}} \end{cases} \quad (2)$$

If the constraint system given by the routability conditions determines a non empty region, then we can assert that the supply graph G has enough capacity to ensure the routability of the considered demand H. Any feasible solution of the above system is a routing policy that can be adopted to satisfy the demand H with routes in G.

Notice that at any iteration, the demand graph H and the residual capacities of the edges of graph G are updated as a consequence of either prune, or split actions. The sets V_{B} and E_{B} are also updated after any repair decision.

For this reason we define the *supply graph at iteration* n as $G^{(n)} = (V^{(n)}, E^{(n)})$, with link capacities $c_{ij}^{(n)}$, and where $V^{(n)} = V \setminus V_{\mathrm{B}}^{(n)}$, and $E^{(n)} = (E \setminus E_{\mathrm{B}}^{(n)}) \setminus \{(i,j) \text{ s.t. } |\{i,j\} \cap V_{\mathrm{B}}^{(n)}| \ge 1\}$. Analogously, we consider the demand graph $H^{(n)}$, updated at iteration n. When necessary, the routability test is performed on the problem instance defined at iteration n, with supply graph $G^{(n)}$ and demand graph $H^{(n)}$.

B. Centrality based ranking

The actions of ISP rely on a ranking among nodes which reflects their relevance in routing the given demand. To this purpose we introduce a new metric of *demand based centrality* in a capacitated network with demands. Unlike previous definitions of node centrality [25], [26], [27], [28], our metric takes account of the ability of each node to route the demand flows throughout the network. Our metric generalizes the notion of *betweenness centrality* [25] which considers only connectivity through shortest paths.

We denote a path p in G with a list of edges $p = < e_1, e_2, \ldots, e_n >$. For shortness of notation, we will also say that a vertex $v \in p$ when v is an endpoint of an edge belonging to p. We denote with $\ell(p)$ the length of the path p, therefore $\ell(p) = \sum_{e_i\in p} l(e_i)$, where $l(e_i)$ is the length of the edge e_i. The capacity of a path is denoted by $c(p)$ and is equal to the minimum capacity of the links in p, therefore $c(p) = \min_{(i,j)\in p} c_{ij}$.

We denote with $\mathscr{P}(i,j)$ the set of acyclic paths in G connecting nodes i, j, such that $(i,j) \in E_{\mathrm{H}}$. We also denote with $\mathscr{P}^*(i,j) \subseteq \mathscr{P}(i,j)$ the set of the first shortest paths necessary to ensure the routability of the demand (i,j), when considered independently of the other demands.

The demand pair $(i,j) \in E_{\mathrm{H}}$ contributes to the centrality of a node v with all the paths $p \in \mathscr{P}_{ij}^*|_v$, where $\mathscr{P}_{ij}^*|_v \triangleq \{p | v \in p \wedge p \in \mathscr{P}_{ij}^*\}$. In particular, for each path $p \in \mathscr{P}_{ij}^*|_v$, the pair (i,j) contributes to the centrality of v with a fraction of the demand d_{ij} equal to the ratio between the capacity of p, $c(p)$, and the sum of the capacities of all the paths in $\mathscr{P}_{ij}^*$.

Given the supply graph G (including broken elements) and the demand graph H, the *demand based centrality* $c_d(v)$ of node v is defined as:

$$c_d(v) \triangleq \sum_{(ij)\in E_{\mathrm{H}}} \left(\frac{\sum_{p\in \mathscr{P}_{ij}^*|_v} c(p)}{\sum_{p\in \mathscr{P}_{ij}^*} c(p)} \cdot d_{ij} \right). \quad (3)$$

If a static distance metric is adopted to calculate the path length, $\mathscr{P}^*(i,j)$ can be calculated offline for any demand pair $(i,j) \in E_{\mathrm{H}}$, and therefore it does not affect the complexity of ISP. Nevertheless, as we discuss in Section IV-D, a dynamic notion of path length which takes account of whether the considered network elements are working or not, may be used to attract more flow to repaired elements.

If the adopted distance metric is dynamic, the centrality of a node may vary significantly during the unfolding of the algorithm, according to the actions provided by ISP. In this case the centrality cannot be calculated offline and needs to be updated at every iteration. In order to have a low complexity, we calculate an *estimated set of paths* $\hat{\mathscr{P}}_{ij}^*$ as follows. We use Dijkstra's algorithm to find the shortest path p between nodes i and j. If $c(p) \ge d_{ij}$ this path is sufficient, otherwise we consider the residual graph in which we reduce the capacity of p by $c(p)$, and we calculate the next shortest path to satisfy a demand $d_{ij} - c(p)$, if available. For each demand d_{ij}, with endpoints $(i,j) \in E_{\mathrm{H}}$, we calculate iteratively the estimated sets of shortest paths $\hat{\mathscr{P}}_{ij}^*$. For each shortest path in $\hat{\mathscr{P}}_{ij}^*$, we can update the centrality of its nodes in linear time with respect to the path length. As a result of this procedure, we obtain an estimate $\hat{c}_d(v)$ of the centrality of each node v, using the Equation 3, where we replace $\mathscr{P}_{ij}^*$ with $\hat{\mathscr{P}}_{ij}^*$.

Notice that, the calculation of the centrality based ranking is performed at each iteration considering the supply graph $G^{(n)}$ (including broken elements), the current demand graph $H^{(n)}$ and the current values of link capacities which may vary iteration by iteration as a consequence of pruning actions.

C. Split of the demand

At the n-th iteration, ISP selects the node $v_{\mathrm{BC}}^{(n)} \in V$ with highest demand based centrality. The centrality ranking does not take account of disruptions, but of the potentiality of a node to contribute to an efficient routing. Hence, the centrality calculation considers the original complete supply graph G (including the broken elements), with updated residual capacities, and the current demand graph $H^{(n)}$. If $v_{\mathrm{BC}}^{(n)} \in V_{\mathrm{B}}^{(n)}$, then $v_{\mathrm{BC}}^{(n)}$ is virtually repaired at the current iteration, therefore it is removed from the set $V_{\mathrm{B}}^{(n)}$ and it is added to the *set of items*

to be repaired, hereby denoted with $R(n)$. Notice that once an element is inserted in the set $R(n)$ it is thereafter considered as if it were already repaired (more details on this set can be found in Section IV-E).

The next step of the algorithm ISP is the split of a demand flow over the node $v_{\text{BC}}^{(n)}$. Let us consider a split action occurring at the n-th iteration. Let us consider also a demand pair $(s_h, t_h) \in E_{\text{H}}^{(n)}$ of value $d_h^{(n)}$. *Splitting d_x units of the demand* $d_h^{(n)}$, with $d_x \leq d_h^{(n)}$ is the action of removing d_x units from the demand associated to the couple (s_h, t_h) and creation of two new demand edges of d_x units of flow on the node couples $(s_h, v_{\text{BC}}^{(n)})$ and $(v_{\text{BC}}^{(n)}, t_h)$.

Figure 1 illustrates the described split action.

The set of demand couples $E_{\text{H}}^{(n)}$ will be updated as follows

$$E_{\text{H}}^{(n+1)} = \{(s_h, v_{\text{BC}}^{(n)}), (v_{\text{BC}}^{(n)}, t_h)\} \cup E_{\text{H}}^{(n)}. \tag{4}$$

The demand flows associated to the edges of $E_{\text{H}}^{(n+1)}$ will be the same as in the previous iteration, with the exception of the split pair and the two new derived pairs. Therefore,

$$d_{zw}^{(n+1)} = d_{zw}^{(n)},\ \forall (z,w) \neq (s_h, t_h), \tag{5}$$

while

$$d_{zw}^{(n+1)} = d_{zw}^{(n)} - d_x,\ \text{if } (z,w) = (s_h, t_h) \tag{6}$$

and the new demand pairs have the following flows:

$$d_{zw}^{(n+1)} = d_x \text{ if } (z,w) = (s_h, v_{\text{BC}}^{(n)}) | (v_{\text{BC}}^{(n)}, t_h). \tag{7}$$

Notice also, that whenever a split action creates a new demand over an already existing demand pair, a unique demand link is created by summing the new demand to the previous.

The split action implies a routing decision, by imposing that d_x units of the split demand between s_h and t_h be routed across the intermediate node $v_{\text{BC}}^{(n)}$ through which the demand has been split. Although this action requires the existence of a set of paths that can be used to route the demand, the only routing decision implied by the split action is the traversal of the node $v_{\text{BC}}^{(n)}$ with d_x units of the original demand $d_h^{(n)}$.

When performing a split action on the selected node $v_{\text{BC}}^{(n)}$, ISP makes two decisions regarding: (1) the demand $h^{(n)}$, and (2) the amount d_x of flow to split.

Decision (1): Let $\mathcal{C}^{(n)}(v_{\text{BC}}^{(n)}) \in E_{\text{H}}^{(n)}$ be the set of demand pairs that positively contributed to the centrality value of the node $v_{\text{BC}}^{(n)}$ at the current iteration, that is:
$\mathcal{C}^{(n)}(v_{\text{BC}}^{(n)}) = \bigcup_{(i,j) \in E_{\text{H}}^{(n)}} \{(i,j) \text{ s.t. } \mathscr{P}^*(i,j)|_{v_{\text{BC}}^{(n)}} \neq \emptyset\}$.

The algorithm ISP selects the demand pair $h^{(n)} \in \mathcal{C}^{(n)}(v_{\text{BC}}^{(n)})$ to be split as the one that can less likely be routed elsewhere, which can be roughly estimated by taking the demand which, if split onto v_{BC}, would more likely use the major portion of the maximum flow between its endpoints. Therefore

$$h^{(n)} = \arg \max_{(i,j) \in E_{\text{H}}^{(n)}} \frac{\min\{d_{ij}^{(n)}, \sum_{p \in \mathscr{P}^*(i,j) | v_{\text{BC}}^{(n)}} c^{(n)}(p)\}}{f^*(i,j)} \tag{8}$$

where $f^*(i,j)$ is the maximum flow between nodes i and j on the complete supply graph G (including broken components) with currently updated capacities $c^{(n)}(\cdot)$, while $\min\{d_{ij}^{(n)}, \sum_{p \in \mathscr{P}^*(i,j) | v_{\text{BC}}^{(n)}} c^{(n)}(p)\}$ is the part of demand $d_{ij}^{(n)}$ that can be routed across node $v_{\text{BC}}^{(n)}$ in case of no conflicts with other demand pairs.

Decision (2): ISP decides the actual amount of demand that can be routed across $v_{\text{BC}}^{(n)}$ by taking account of all potential conflicts with the other demands at the current iteration. Let d_x be such an amount, that is the part of $d_h^{(n)}$ that can be split on $v_{\text{BC}}^{(n)}$ without affecting the routability of the current iteration instance of the problem on the supply graph $G^{(n)}$. The amount d_x can be calculated by solving the linear programming problem to maximize d_x under constraints of $d_x \leq d_h^{(n)}$ and to the flow conservation and capacity constraints defined by equations (2), where the set $E_{\text{H}}^{(n)}$ is defined according to Equation (4), and the demand flows are defined according to Equations (5), (6) and (7).

D. On the use of a dynamic path metric

We use a measure of link length proportional to the cost of repairing the link or its endpoints, if any of these is broken, and inversely proportional to the link capacity. Such metric is updated every time a broken component is repaired or the residual capacity of a link is reduced due to a pruning action (see Section IV-F).

Formally, we define the length of the edge $e_{ij} = (i,j) \in E$ at iteration n as $l^{(n)}(e_{ij}) = [\mathit{const} + k_{ij}^e(n) + (k_i^v(n) + k_j^v(n))/2]/c_{ij}$, where the terms *const*, $k_i^v(n)$ and $k_{ij}^e(n)$ are as follows. The term *const* is a constant needed to account for the length of a working link. The terms $k_i^v(n)$ and $k_{ij}^e(n)$ are non null only if the corresponding elements are broken and not listed for repair in any previous iteration: therefore $k_i^v(n) = k_i^v$ if $i \in V_{\text{B}}^{(n)}$, and null otherwise. Similarly, $k_{ij}^e(n) = k_{ij}^e$ if $(i,j) \in E_{\text{B}}^{(n)}$ and null otherwise.

This path metric gives an extraordinary strength to the algorithm ISP because, if a decision to repair an element has been made, all successive actions will be performed accordingly. For instance, the nodes belonging to paths containing a repaired component will see an increased centrality measure after the repair, because they more likely belong to shortest paths. Henceforth, paths containing repaired components will more likely be selected for subsequent split and pruning actions.

E. Recovery of nodes and edges

The algorithm ISP works by virtually recovering network components during its execution until a sufficient number of edges and links are recovered to route the entire demand. These progressive recovery decisions alter the problem instance at any iteration. ISP considers a *set of items to be repaired* $R(n)$, which is updated at any new repairing decision.

At any iteration n of the algorithm, if the best candidate v_{BC} is broken, that is $v_{\text{BC}} \in V_{\text{B}}^{(n)}$, it is added to the current set of repairs, so $R(n+1) = R(n) \cup \{v_{\text{BC}}\}$, and the set of broken vertices is updated as follows: $V_{\text{B}}^{(n+1)} = V_{\text{B}}^{(n)} \setminus \{v_{\text{BC}}\}$. Moreover, we repair a broken link in the supply graph if such a link directly connects two endpoints of a demand, and such a demand cannot be satisfied by the current repairs. Formally, if

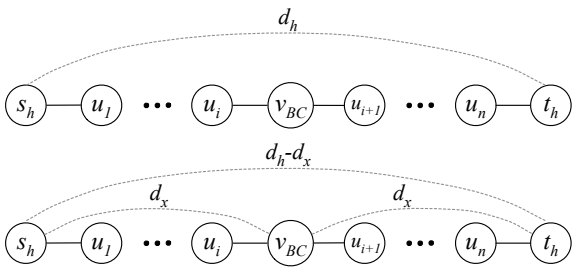


Fig. 1. Split of d_x units of demand

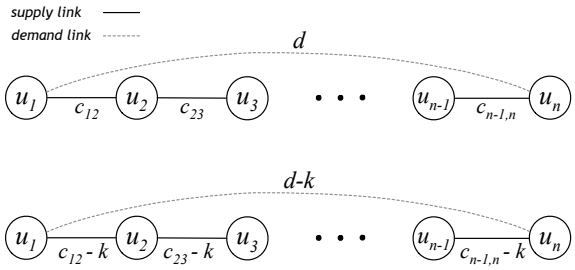


Fig. 2. Pruning of k units of demand

at any iteration n there is a demand $(s_h, t_h) \in E_{\text{H}}^{(n)}$ that cannot be satisfied by any working path (including the links in $R(n)$), and there is also a supply broken edge $(s_h, t_h) \in E \cap E_{\text{B}}^{(n)}$ with the same endpoints, then the supply edge (s_h, t_h) is added to the set of repairs, that is $R(n+1) = R(n) \cup \{(s_h, t_h)\}$. The set of broken edges at the current iteration is also updated accordingly $E_{\text{B}}^{(n+1)} = E_{\text{B}}^{(n)} \setminus \{(s_h, t_h)\}$.

F. Pruning

The algorithm ISP executes the *pruning activity* to simplify the problem instance, when some units of demand can be routed over working paths. This may occur at the beginning of the algorithm execution or during its unfolding, after some split or repair actions.

According to ISP, k units of the demand flow d between a pair $(u_1, u_n) \in E_{\text{H}}^{(n)}$, with $k \leq d$, can be pruned at iteration n only if there is a working path p between u_1 and u_n in the supply graph with capacity at least k. This is only a necessary condition for a demand to be *prunable*, and it does not imply that it will certainly be pruned. Figure 2 illustrates the pruning action. More formally, given the demand pair $(u_1, u_n) \in E_{\text{H}}^{(n)}$ with a demand flow d, k units of this demand ($k \leq d$) can be pruned on path p if (1) $p \subseteq E^{(n)}$, and (2) $c(p) \geq k$. The pruning action consists in the removal of k units from the demand edge $(u_1, u_n) \in E_{\text{H}}^{(n)}$ and routing these k units on a selected path p, thus subtracting the related capacity from any of the composing edges. Therefore, after the pruning action of k units, $d_{u_1,u_n}^{(n+1)} \leftarrow d_{u_1,u_n}^{(n)} - k$, and for any edge of the selected path $(i,j) \in p$, $c_{ij}^{(n+1)} \leftarrow c_{ij}^{(n)} - k$. If a demand is completely pruned, the demand pair is removed from $E_{\text{H}}^{(n)}$. Moreover, if one or both of its endpoints do not belong to any other demand pair, then such endpoints are removed from $V_{\text{H}}^{(n)}$.

It must be noted that, like the splitting action, the pruning action implies a routing decision which may possibly lead to an infeasible solution of the problem. In the following, we give a sufficient condition for pruning to be feasible. Given a demand h between the pair (s_h, t_h), the set $S_h \subset V$ is a *bubble for* h if it contains only vertices that cannot be reached by any demand node in V_{H} without traversing either s_h or t_h. More formally, we give the following definition.

Definition IV.1 (Bubble)**.** *Given a supply graph $G = (V, E)$ and a demand graph $H = (V_H, E_H)$, a set $S_h \subseteq V$, is a* bubble *for demand $h \in E_H$ if $S_h \cap V_H = \{s_h, t_h\}$, and $\forall (i,j) \in \delta_G(S_h)$, it holds that $|\{i,j\} \cap \{s_h, t_h\}| = 1$, where $\delta_G(S_h) = \{(i,j) \in E,\ s.t.\ |\{i,j\} \cap S_h| = 1\}$ is the supply cut of S_h.*

Theorem IV.1 (Prune conditions)**.** *Consider a supply graph G and a demand graph H, which satisfy the routability conditions given by equations (2). Let us consider a demand $h \in E_H$ between the pair (s_h, t_h) and flow d_h. If there is a set of working paths $\mathscr{P}(s_h, t_h)$ with maximum flow $f^*(\mathscr{P}(s_h, t_h))$ that can satisfy the demand, such that the set of vertices S_h forming the paths of $\mathscr{P}(s_h, t_h)$ is a bubble for the demand h, then the demand between s_h and t_h can be pruned on the paths of $\mathscr{P}(s_h, t_h)$ for an amount equal to $k_h \triangleq \min \{f^*(\mathscr{P}(s_h, t_h)), d_h\}$ without compromising the routability of the demand and without worsening the final solution in terms of recovered components.*

Proof. See the Appendix for the proof of the theorem □

Notice that, in order to find demand bubbles, ISP adopts a modified breadth first search visit starting from one of the demand endpoints, and discarding all paths that lead to any endpoint of another demand. As the purpose of ISP is to minimize the number of repairs and not to find an efficient routing of the demand, any of the feasible assignments of a demand to one or several paths of one of its bubbles can be used for pruning. Moreover the pruning action must be performed by routing on the selected path the maximum amount of demand that is prunable, that is k_h which is the minimum between the maximum flow $f^*(\mathscr{P}(s_h, t_h))$ of the set of paths from s_h to t_h and the demand d_h.

G. Properties of ISP

Theorem IV.2. *ISP terminates in a finite time.*

Proof. See the Appendix for the proof of the theorem. □

Theorem IV.3. *ISP has polynomial time complexity.*

Proof. See the Appendix for the proof of the theorem. □

V. A discussion on future extensions of ISP

We underline that the MinR problem is centralized in nature. It deals with the restoration of critical services, typically involving governmental entities and emergency service providers. For example, in the USA the Federal Emergency Management Agency (FEMA) is in charge, according to the Stafford act, of providing disaster relief and emergency assistance in the territory of the USA. The Agency recognizes the communication infrastructure as critical for the community

and includes it in the list of the infrastructures to be repaired to restore critical communication services during an emergency, with utmost urgency. Despite the fact that multiple private businesses may own different parts of the communication infrastructure, FEMA promotes a holistic approach to disaster recovery providing financial and physical assistance. The example of FEMA holds for the USA, but almost every country that recognizes the relevance of the communication network as a critical infrastructure adopts identical policies. This highlights the governments' priority to collaborate with the business community during emergency and motivates the use of a centralized approach to the MinR problem even in a multi-domain network, to enable the formulation of a unique fast recovery plan.

Nevertheless, once critical services are completely restored, the next step will be the restoration of the other second priority services. In such a case we envision the adoption of a distributed version of ISP in which multiple domain owners cooperate loosely to restore the functionality of the global network. Similarly we believe that a distributed version of ISP may be useful when non-critical communication networks incur a large scale failure.

As an example of a distributed multi-domain implementation, ISP can be adopted as the local solution of a hierarchical approach in which demand flows are restored with local repairs when the endpoints belong to a single domain. After the local intra-domain execution of ISP, some demands will be pruned (only if their endpoints lie on a single domain network) but some others will be left unaddressed, including demands with endpoints in different domains, or demands with endpoints in the same domain but for which the local network capacity is insufficient. These demands will be addressed at a higher level of the hierarchical solution, and through the limited cooperation of the involved multiple domain owners. This and other distributed approaches to the MinR problem will be addressed in a future work.

VI. Heuristics

A. Drawbacks of a multi-commodity based approach

Considering the MinR problem of Equation (1), we can see some similarities with a classic problem known as the *Multi-commodity flow problem* (MCFP) [29]. MCFP aims at finding a flow routing scheme that maximizes the flow routed between given demand endpoints, under link capacity constraints (the flow across a link cannot exceed its capacity) and flow balance constraints (the flow that enters a node is the same that exits the same node, with the exception of the endpoints of each demand). With this consideration, it may seem reasonable to extend the multi-commodity flow problem, to consider as a new objective the minimization of the amount of flow crossing broken links. If this extension were possible, we could use classic approaches of operations research and optimization theory, known for the multi-commodity flow problem, to address the MinR problem. The corresponding formulation in terms of MCFP is:

$$\begin{array}{lll} \min \sum_{(i,j)\in E_{\mathrm{B}}} k^e_{ij} \cdot \sum_{h\in E_{\mathrm{H}}} f^h_{ij} & & (a) \\ \sum_{h\in E_{\mathrm{H}}} (f^h_{ij} + f^h_{ji}) \leq c_{ij} & \forall (i,j) \in E & (b) \\ \sum_{j\in V} f^h_{ij} = \sum_{k\in V} f^h_{ki} + b^h_i & \forall (i,h) \in V \times E_{\mathrm{H}} & (c) \\ f^h_{ij} \geq 0 & \forall (i,j) \in E, h \in E_{\mathrm{H}} & (d) \end{array} \tag{9}$$

Translating the solution of problem (9) in terms of recovery decisions, implies the decision to repair all the broken links and vertices that are actually used by the optimal solution.

Under this formulation, the problem is no longer NP-hard, but has polynomial time complexity, being it solvable efficiently with LP methods [30].

Nevertheless, this formulation has a wide range of equally optimal solutions which vary significantly in the number of repaired edges and vertices. We denote with MCB and MCW the best and the worst of these solutions, respectively, in terms of number of repaired elements. Figure 3 illustrates the performance of MCB and MCW, versus the optimal solution of MinR and the trivial solution of repairing all broken elements (OPT and ALL in the figure, respectively). The results of Figure 3 are obtained considering a real fiber physical topology, the Palmetto topology [31], [32], whose characteristics are given in details in Section VII, where we describe the simulation settings. In the figure we show how the number of repairs varies by increasing the number of demand pairs, under the experimental setting explained in Section VII-A1. The results show that the MCFP approach has a wide

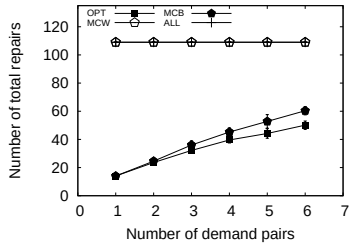


Fig. 3. Total number of repairs of multi-commodity based solutions

solution space, which includes solutions close to the optimum as well as solutions close to the worst possible one that is to repair all broken elements. Notice that the optimal solution of MinR repairs fewer network elements than MCB because it takes account of both vertex and edge repairs. We underline that in order to obtain MCB among the wide set of solutions of problem (9) we have to solve an NP-hard problem, being it an instance of MinR. For the above mentioned reasons we do not include the multi-commodity approach in our results.

B. Shortest Path Heuristic (SRT)

This heuristic is based on a very intuitive approach to the MinR problem, that is to consider all the demands (s_i, t_i, d_i) in decreasing order of demand flow d_i, and repair all the shortest paths that are sufficient to meet the demand requirements, considering the individual demands, one by one.

The path length is calculated according to the dynamic notion of distance introduced for ISP in section IV-D. Hence

the shortest path between two endpoints is the path that is either already working or has the lowest cost of repair.

Let S_i be the set including the first shortest paths for the $i-th$ demand, such that the maximum flow traversing the sub-graph formed by the only paths in S_i is at least d_i.

SRT considers all the demand in decreasing order and for each d_i it repairs all broken nodes and edges in S_i.

This heuristic has polynomial time complexity, as it considers the demand pairs one at a time without considering potential conflicts with other demand pairs. For each demand pair, it requires to calculate the shortest paths to be repaired iteratively on a residual graph. Paths are selected until they are sufficient to meet the demand in a single flow scenario.

It is important to notice that the sets of shortest paths of different demands may overlap on some links, therefore the repaired links may be insufficient to route all flows and there can be some demand loss.

C. Greedy Heuristics

We developed two other heuristics based on a mapping between paths of the MinR problem and objects of an instance of a Knapsack problem. According to this mapping, we create a knapsack object for each path connecting a demand pair in H. The cost of repairing such a path is the weight of the corresponding knapsack object, while the path capacity is the object value. Both the greedy heuristics make use of the set $P(H,G)$ of all simple paths between the endpoints of the demand pairs in H. Notice that the number of paths in $P(H,G)$ is potentially exponential in the graph size, hence these heuristics can only be adopted if paths are pre-computed offline. Thanks to the described mapping, we can formulate two different heuristics based on the greedy approach to the Knapsack problem [33].

1) Greedy Commitment (GRD-COM)

The first heuristic, called *Greedy Commitment* (GRD-COM), assigns to each path $p \in P(H,G)$ a weight $w(p) = \frac{cost(p)}{capacity(p)}$, where $cost(p)$ is the sum of the costs of repairing the edges composing p, while $capacity(p)$ is the residual capacity of p. GRD-COM sorts the paths in $P(H,G)$ in ascending order of weight, and iteratively repairs paths following this order. Let p be the path repaired at the current iteration, and (s_i,t_i) the demand pair whose endpoints are connected by p. GRD-COM assigns the maximum possible quantity of such demand to p, and updates the residual capacities of edges and the residual demand accordingly. It then verifies if also some other demands may be routed through the current graph, considering all the paths already repaired including p. The algorithm proceeds to the next iteration, selecting the next path in the order. GRD-COM terminates as soon as all demands are satisfied, or there are no more paths to repair.

Note that considering the residual graph capacities allows a lower amount of repairs with respect to the following greedy heuristics GRD-NC, but as in the case of SRT, there is no guarantee that all the demands can be satisfied due to the possibility to have wrong routing decisions, which may create inhibiting flow allocations, even if the capacity of the repaired edges is enough to route the demand.

Algorithm GRD-COM

Input: G, H, V_{B}, and E_{B}
Calculate (offline) $P(H,G)$;
while $\exists$ *unsatisfied demands and available paths* **do**
  for $p \in P(H,G)$ **do** $w(p) = \frac{cost(p)}{capacity(p)}$;
  Sort paths according to their weight;
  Let p be the next path, (s_i,t_i) its demand pair;
  Repair p;
  Assign a quantity of demand $\min\{d_i, capacity(p)\}$ to p;
  Update G and H;
  for *each routable demand flow* (s_k,t_k,d_k), $k \neq i$ **do**
    Assign the maximum quantity of demand;
    Update G and H;

2) Greedy with No Commitment (GRD-NC)

The second heuristic is called *Greedy with No Commitment* (GRD-NC). It is also inspired by the Knapsack heuristics, and similarly to GRD-COM, it makes use of the set of all paths $P(H,G)$ and path weights $w(\cdot)$.

GRD-NC repairs paths one by one following the ascending order of weights, but it does not provide a routing assignment of flows to paths. On the contrary, it evaluates the routability of the overall demand, given the current repaired paths, using the routability test described in Section IV-A. GRD-NC terminates as soon as all demands are routable with the current repairs.

Note that GRD-NC does not provide an update of the path capacity at each step, since there is no routing assignment after the repairs. As a consequence, this heuristic can repair more edges and vertices than GRD-COM, but it has the advantage that if the demand is routable in the original graph before the disruption, the heuristic finds a solution with no demand loss.

Algorithm GRD-NC

Input: G, H, V_{B}, and E_{B}
Calculate (offline) $P(H,G)$;
for $p \in P(H,G)$ **do** $w(p) = \frac{cost(p)}{capacity(p)}$;
Sort paths according to their weight;
while *routability test fails* **do**
  Repair the next path p;

VII. Simulation study

In order to provide an evaluation study of the proposed approaches we realized a simulator using the Python language. We use the Gurobi [34] package to calculate the optimal solution of the MinR problem of Equation (1).

In our simulation study we consider both real and synthetic topologies of various size to highlight different aspects of the performance of the algorithms.

The first scenario of our simulations is designed to give a thorough comparison of the algorithms being analyzed in a variety of experimental settings. We considered a real physical layer topology of small size, which corresponds to the fiber network connectivity of the Palmetto network, between North and South Carolina. We considered such a small size network in order to be able to compute the optimal solution in a

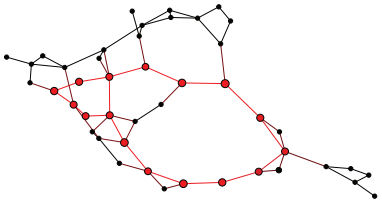


Fig. 4. Palmetto physical layer topology (with 45 nodes and 64 edges).

reasonable time for the various operational settings, despite the NP-hardness of the problem.

By considering various MinR instances of different size, we recognized that the critical aspect affecting computation time of the optimal solution is the size of the supply network and in particular the average degree of the nodes. To underline the scalability issues of the optimal approach, in the second scenario of our simulations, we stress the algorithms by using synthetic topologies of increasing complexity. While the topologies being considered in this scenario do not reflect the properties of any real physical networks, these simulations provide a scalability study. We will evidence the poor scalability of the optimal approach, motivating the need to resort to heuristic solutions.

In the last experimental scenario we considered another real example of physical layer network, that is the fiber network of Minnesota, whose size is much larger than the one considered in the first scenario. The purpose of this scenario is to evidence the good approximation of ISP to the optimal solution even with a large problem size.

In all the following simulations, where not otherwise stated, we average the results over 20 runs.

A. First scenario

In this set of simulations we consider the Palmetto physical layer topology of Figure 4, taken from the Internet Topology Zoo [31], [32] collection.

This graph reflects the fiber connectivity of the area between North and South Carolina. It is composed of 45 nodes and 64 edges. Detailed information on the edge capacity of this topology was provided by CenturyLink [35]. In Figure 4, the 10 Gb/s GbE backbone lines correspond to the red links, while all the remaining OC-48 edges with 2.5 Gb/s are drawn in black. Where not otherwise stated, we use a homogeneous unitary repairing cost for damaged nodes and edges.

We build the demand graph $H = (V_H, E_H)$ as follows. We select the demand pairs to be far apart in the supply graph. In particular, we randomly select the demand pairs among those having a hop distance greater than or equal to half the diameter of the network. In most of the experiments we varied the demand either in terms of number of demand pairs or demand flow of each pair. We start this evaluation with a low load scenario, and gradually increase the demand until the problem instance is no longer feasible.

We perform four sets of simulations. In the first set (Section VII-A1) we fix the flow per pair, and increase the number of pairs in the demand graph. In the second set (Section VII-A2), we fix the number of demand pairs and increase the demand

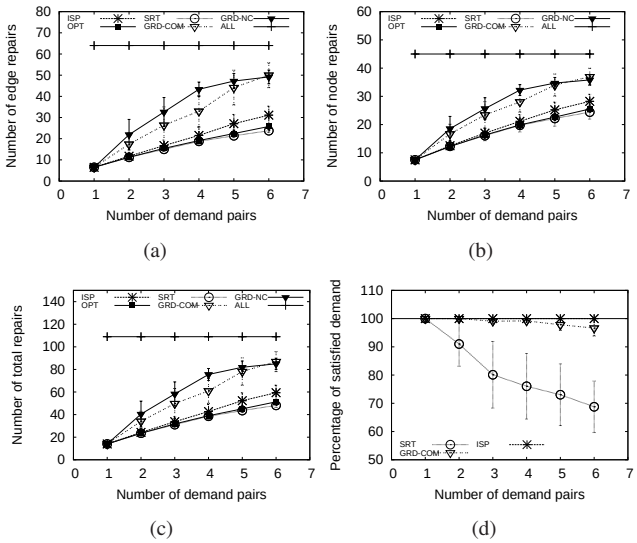


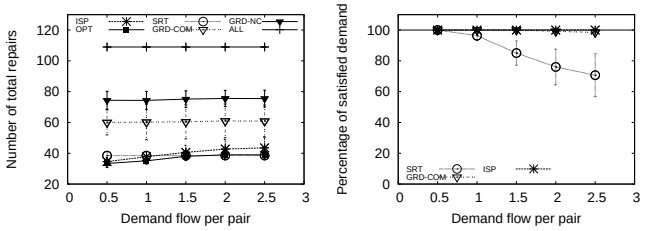
Fig. 5. Palmetto topology. Varying number of demand pairs (2 Gb/s for each pair). Repaired edges (a), repaired nodes (b), total repairs (c) and demand loss (d).

flow per pair. In both these sets of simulations, we considered a complete destruction of the supply graph, in order to have the maximum range of potential solutions. On the contrary, the third set of simulations (Section VII-A3) considers different failure scenarios according to a geographically correlated failure model. Finally, the fourth set of simulations considers the case of heterogeneous costs of repair.

1) Variation of the number of demand pairs: In these simulations we increase the number of demand pairs from 1 to 6, where each demand pair corresponds to an aggregated flow of 2 Gb/s. Figures 5(a) and 5(b) show the number of edges and nodes repaired by the considered approaches. Figure 5(c) shows the cumulative number of repairs. In the figures, the line ALL refers to the total number of destroyed nodes and links. In these simulations we consider a total destruction of the network components.

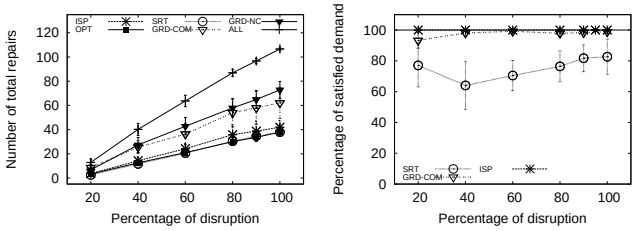
The results shown in Figures 5(a)-(c) highlight that by linearly increasing the number of demand pairs, the number of repaired edges and vertices also grows.

We notice that SRT repairs the smallest number of network components, in some cases even smaller than the optimal. This happens because the heuristic SRT, as well as GRD-COM, does not guarantee that all demand flows will be satisfied. In particular, SRT repairs the minimum number of shortest paths that are necessary to satisfy each demand, without considering the other demands. As the number of demand pairs increases, the paths selected by SRT for each demand are more likely to traverse links shared with other demands. Therefore when these shared links reach their capacity limit, the policy SRT is no longer able to satisfy all demands, as is evidenced by Figure 5(d). In these simulations, we see that even with only two pairs, SRT shows a demand loss of the 10% of the total demand. In fact, since the minimum capacity link is 2.5 Gb/s, a single demand flow of 2 Gb/s can be routed on a single path. Nevertheless if there are two or more demand flows, they may conflict in the choice of a same link in their shortest path,



(a) (b)

Fig. 6. Palmetto topology. Varying the intensity of demand flow (4 demand pairs). Total repairs (a), demand loss (b).



(a) (b)

Fig. 7. Palmetto topology. Varying the extent of destruction (4 demand pairs, 2Gb/s per pair). Total repairs (a) and demand loss (b).

causing demand loss also with only two demand pairs. This explains the growing demand loss shown in Figure 5(d) and in the analogous figures of the following sets of simulations.

The heuristic GRD-COM also shows demand loss, as can be seen in Figure 5(d), due to erroneous routing decisions. Nevertheless, GRD-COM takes account of the residual capacity of the links when selecting routing paths, hence is able to solve conflicts among demands better than SRT.

The heuristic GRD-NC has a similar behavior to GRD-COM as the demands are routed on paths selected with similar criteria (the weight function). Nevertheless, GRD-NC guarantees the 100% of demand satisfaction thanks to the routability test, but at the expense of some additional repairs with respect to GRD-COM.

Excluding SRT, which shows a considerable demand loss, ISP is the heuristic that performs the smallest number of repairs, close to the optimal solution. In the most critical setting, with 6 demand pairs, OPT repairs 51 network elements, ISP repairs 60 elements, while GRD-COM and GRD-NC repair about 83 network elements.

The superiority of ISP to the other heuristics evidenced by Figure 5 is due to its capability to either route the flow on already working paths, thanks to the pruning activity, or to concentrate the flow onto repaired portions of the network, thanks to the use of the demand based centrality metric to determine the best candidate nodes that will be traversed by the demands. Moreover, we highlight that the greedy solutions GRD-COM and GRD-NC are much more computationally expensive than ISP, due to the necessity to find all paths between any demand pairs. It is also worth noting that ISP better approximates the optimal solution when the demand requirements are low with respect to the available bandwidth in the network. This result is visible in Figures 5 (a)-(c), when the number of demand pairs is less than 3. For a higher number of demand pairs, ISP remains within 20% of the optimal, until the number of pairs becomes so high that the generated instances of MinR are infeasible.

2) Variation of the demand intensity: In these set of simulations we fix the number of demand pairs to 4, and we vary the intensity of demand per pair. We consider a total destruction of the network components, as evidenced by the line labeled ALL in all the referred figures. Figures 6(a) and 6(b) show the total number of repaired elements and the demand loss. We omit the figures on the number of node and edge repairs for space limitation. We observe a similar behavior to what we discussed for the previous set of simulations. In summary, SRT performs a small number of repairs at the expense of a considerable demand loss, because the shortest paths between the endpoints of different demands overlap on some links. The greedy heuristic GRD-COM has a negligible demand loss only when the demand flow per pair is 2.5 Gb/s and for few runs, thanks to the fact that it routes demand on paths which have enough residual capacity. GRD-NC guarantees the demand satisfaction but at the expense of more repairs than GRD-COM, thanks to the use of the routability test.

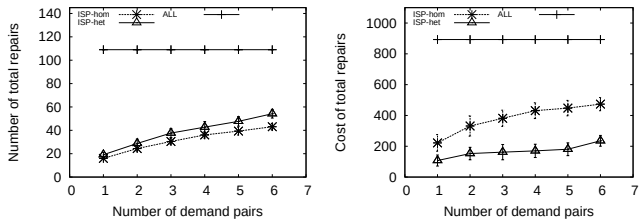
Also in this case, ISP is able to obtain maximum benefit from the performed repairs, aggregating flows on the same repaired links when possible, showing a number of repairs close to the optimal.

In this scenario, the demand pairs are fixed, and we only increase the flow per pair. All policies tend to reveal a smoother increase in the number of repairs with respect to the previous scenario. This is due to the fact that when the flow demand is low (e.g. 0.5 Gb/s) the paths repaired to connect the demand endpoints are underutilized and are sufficient to accommodate a further increase of demand flow for each pair.

3) Variation of the extent of destruction: In this set of simulations we consider the impact of the extent of the destruction. We consider a geographical failure model. We generated the disruption according to a bi-variate Gaussian distribution of the disruption probability of network components. The distribution has a central symmetry around an epicenter. Network components located in regions that are closer to the epicenter are more likely to be disrupted than others. In the simulations we gradually increase the variance of such a distribution and scale the probability accordingly to obtain larger failures with larger variance. The line labeled ALL shows how many edges or vertices of the 109 network elements of the Palmetto topology are disrupted in the considered instance of the problem.

Figures 7(a) and 7(b) show the total number of repaired elements and the percentage of demand loss, respectively, when increasing the variance of the disruption, and consequently the percentage of network components that are disrupted.

In these simulations we consider 4 demand pairs, each with a demand intensity of 2 Gb/s. The algorithms show the same behavior we described for the previous set of simulations. In particular, ISP performs close to the optimal, and when the network is almost completely destroyed ISP repairs only 41 elements, against the 39 elements repaired by the optimal solution, whereas GRD-COM requires 61 repairs and GRD-



(a) (b)

Fig. 8. Palmetto topology. Heterogeneous recovery costs. Total repairs (a), cost of repairs (b).

NC requires 72 repairs. It is interesting to notice that the SRT heuristic shows a local minimum of demand satisfaction when the destruction involves about the 40% of the network components. In fact, when the amount of destroyed elements is lower than 40%, in many cases the demand endpoints have working paths, hence SRT is likely to find disjoint paths to route the demands, on the basis of the hop count. The same happens when the disruption is large, as most of the network elements are destroyed, SRT tends to select disjoint paths, on the basis of the hop count. Nevertheless when the disruption percentage has intermediate values, SRT tends to select paths sharing the few existing working links. This causes more conflicts than in the two extreme situations, and more demand loss. For similar reasons the heuristic GRD-COM tends to lose less demand when the disruption is complete, as it does not attempt to concentrate the flow across the existing working links.

4) Heterogeneous costs: In this set of simulations, we modeled heterogeneous repair costs to highlight the ability of ISP to adapt its choices to reduce the total cost of repairs and not simply the number of repairs. To make this scenario more realistic, we set the cost of the links according to their capacity, and considered a complete disruption of the network. We considered different repair costs for the 10 Gb/s links (red links in Figure 4) and the 2.5Gb/s links (black links in Figure 4), respectively of 50 and 1 cost units, and unitary cost for node recovery. We considered an increasing number of demand pairs, all of 0.5 Gb/s. The adoption of a smaller demand value with respect to previous scenarios is motivated by the need to have more freedom of choice between low and high capacity links. The results are shown in Figure 8.

Since ISP utilizes the cost dependent metric of distance introduced in Section IV-D, the centrality ranking will prioritize the repair of nodes traversed by low cost paths.

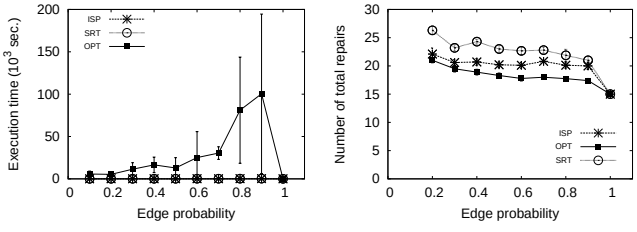
We compared the results of ISP under the described heterogeneous cost setting (in the pictures called *ISP-het*) with the execution of ISP over the same network but in a cost-blind scenario, where it assumes uniform costs (in the pictures named *ISP-hom*). In terms of number of repairs, shown in Figure 8(a), ISP-hom performs better than ISP-het, requiring fewer repair interventions. This is because ISP-hom, tends to prioritize the repair of higher capacity links, since it assumes homogeneous repair costs. Nevertheless, as repairing a backbone link costs more than repairing smaller capacity links, ISP-het is capable to repair sufficient capacity to meet the demand at a much smaller cost, although repairing a slightly higher number of links. When the two algorithms are compared in terms of the real heterogeneous costs, as shown in Figure 8(b), we can see that ISP-het performs better than ISP-hom as it reduces the total cost of repairs.

B. Second scenario

In this scenario, we compare the scalability of ISP and OPT. We consider synthetic network topologies of increasing complexity and we evaluate the performance and the computation time of the two algorithms. We considered a complete destruction of an Erdos-Renyi topology [36] with 100 nodes and varying number of edges. We underline that this type of random graph does not reflect the property of any real world physical layer network. The purpose of this set of simulations is to evidence the poor scalability of the optimal solution with respect to a growing number of edges in the network. We recall that in an Erdos-Renyi graph, any two nodes are connected through an edge with probability p (*edge probability*). In the simulations we varied this parameter.

As the purpose of this set of experiments is to evaluate the algorithm scalability, we consider a relatively simple problem instance in which we have only connectivity requirements between demand endpoints. For this purpose, we sized the demand flow and link capacity as follows: we considered 5 demand pairs, of one unit, while links have uniform capacity of 1000 units.

In Figure 9(a) we focus on the execution time of ISP and OPT. For these experiments we used a 20 core/40 thread architecture composed of 2 Intel(R) Xeon(R) CPU ES-2680 v2 (2.80GHz) and 64GB RAM, running Ubuntu 14.04. The experiments show that the optimal solution has a prohibitive execution time, which as expected grows significantly with the parameter p. For instance, we observe that when p=0.9 OPT requires 10^5 secs (about 27 hours), on average. The execution time of ISP is negligible and not affected by this parameter setting. When p=1 the problem becomes trivial, as the supply network is a clique, and the optimal solution consists in repairing the endpoints of each demand pair and the edge connecting them. When p=1 the number of repairs is 15 for all the three plotted algorithms, as the supply network is a clique and all the algorithms are able to find the trivial solution of repairing the endpoints of each demand pair and the links between them, for a total of 5 pairs.



(a) (b)

Fig. 9. Erdos-Renyi topology. Varying edge probability p. Execution time (a), number of total repairs (b).

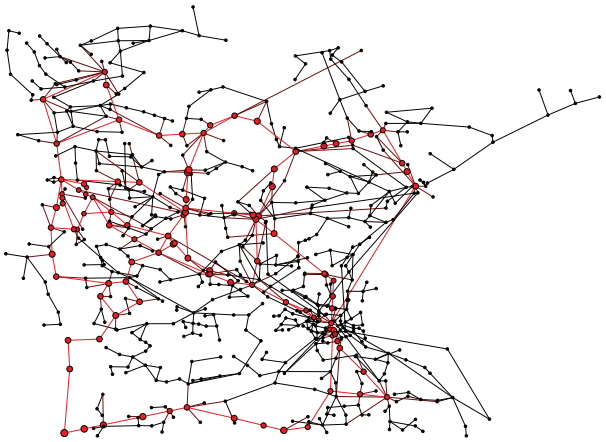


Fig. 10. Minnesota fiber network topology (681 nodes and 921 edges).

For these simulations, the link capacity is so high that none of the heuristics has any demand loss. Notice also that we do not plot the greedy heuristics that are based on the pre-computation of the list of all paths, because with high values of p they would require $O(N!)$ steps.

C. Third scenario

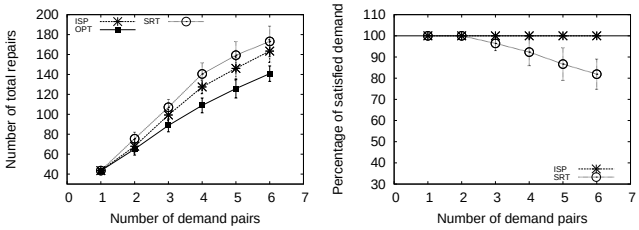
For this final scenario, we consider the map of the fiber network of Minnesota shown in Figure 10, made available by Aurora Fiber Optic Network [37], which results from the collaboration of more than 50 carriers in the state. The figure highlights the backbone lines in red. As in the first simulation scenario, the backbone is a 10Gb/s GbE line, while the other links are OC-48 lines with 2.5 Gb/s. The network is composed of 681 nodes and 921 edges. We consider the case of complete network destruction.

In these simulations we varied the number of demand pairs, for each considering an aggregate demand of 1.5 Gb/s. This study is qualitatively similar to the one of Section VII-A1 but is conducted on a much larger network and is meant to evidence that ISP performs similarly well also for larger networks. Due to the large size of the network we do not include the heuristics GRD-NC and GRD-COM in this evaluation, because of the computational cost to calculate all the paths between the demand endpoints.

In Figure 11(a), we increase the number of demand pairs until we lose feasibility of the problem. The corresponding increase in the number of repairs is such that the number of elements repaired by ISP remains within the 15% of the optimal solution, and allows the routing of the entire demand, as shown in Figure 11(b). On the opposite, SRT shows a considerable demand loss, despite the higher number of repaired elements.

It is interesting to notice that in this large network, unlike the observation in all the simulations for the first scenario, SRT repairs more network elements than ISP and shows a considerable demand loss.

This confirms the capability of ISP to find solutions in which repaired elements are shared among multiple demands, thanks to the use of the new concept of demand based centrality in



(a) (b)

Fig. 11. Minnesota fiber network topology. Varying the number of demand pairs (1.5 Gb/s per pair). Total repairs (a), demand loss (b).

selecting which components to repair and use for routing. In the large network considered in this scenario, ISP expresses its full potential with respect to simple policies such as SRT. While still showing a low computation time, ISP performs close to the optimal, and better than SRT in all the performance metrics.

VIII. Conclusions

In this paper we consider, for the first time, the problem of recovery of a communication network after large scale failures. We model this problem, named Minimum Recovery (MinR), as a Mixed Integer Linear Programming (MILP) problem, and show that it is NP-Hard. We propose ISP, an efficient heuristic to solve MinR, based on a novel demand based centrality metric. ISP makes use of this metric to iteratively select the best nodes for repair, and concentrates the flow on them by means of split actions. It additionally prunes demand flows if they can be satisfied by the currently repaired supply network. We also proposed several greedy heuristics as baselines for comparisons. Experimental results on real and synthetic topologies show that ISP outperforms other approaches in number of repairs and in execution time. In particular, it achieves a number of repairs close to the optimum without incurring any demand loss. A distributed variant of ISP is being considered as future work.

Appendix
Proofs of the Theorems

Theorem III.1. *The problem MinR is NP-Hard.*

Proof. Let us consider a generic instance of the Steiner Forest problem [38]. Given a graph $G_{\mathtt{sf}} = (V_{\mathtt{sf}}, E_{\mathtt{sf}})$, a set of node pairs $S_{\mathtt{sf}} = \{(s_1, t_1), \ldots, (s_n, t_n)\}$ and a cost function $c_{\mathtt{sf}} : E \to \mathbb{R}^+$, the goal of the Steiner Forest problem is to find a forest $F_{\mathtt{sf}} \subseteq E$ with minimum cost, such that for each pair (s_i, t_i), s_i and t_i belong to the same connected component in $F_{\mathtt{sf}}$.

We reduce this problem to an instance of MinR as follows. We consider a supply graph $G = (V, E)$ with $V = V_{\mathtt{sf}}$ and $E = E_{\mathtt{sf}}$. We consider $E_{\mathtt{B}} = E$ and $V_{\mathtt{B}} = \emptyset$. We create a unitary demand flow for each pair in $S_{\mathtt{sf}}$. For each edge in E we set the cost of repair equal to the cost of the corresponding edge in $G_{\mathtt{sf}}$, and its capacity equal to a value L that is sufficiently large that any link of E can accommodate the sum of all demand flows. Therefore, considering a requirement of one unit of flow for each demand pair, it is $L \gg |S_{\mathtt{sf}}|$.

Given such instance, MinR returns the set of nodes $V^* \subseteq V$ and edges $E^* \subseteq E$ to be repaired to accommodate all the demand flows. However, $V^* = \emptyset$, since no node is damaged. Additionally, since the capacity of each edge in E is large enough to accommodate an amount of flow exceeding the sum of all demand flows, for each demand pair (s_i, t_i) a single path from s_i to t_i is sufficient to accommodate the demand flow between s_i and t_i. As a result, the union of the links in E^* generates a Steiner forest, since any cycle would imply unnecessary repairs. This is also the forest with minimum cost, since MinR minimizes the costs of repairs. We can conclude the reducibility of the Steiner Forest problem to MinR, and consequently that the problem MinR is NP-Hard. □

Theorem IV.1 (Prune conditions)**.** *Consider a supply graph G and a demand graph H, which satisfy the routability conditions given by equations (2). Let us consider a demand $h \in E_H$ between the pair (s_h, t_h) and flow d_h. If there is a set of working paths $\mathscr{P}(s_h, t_h)$ with maximum flow $f^*(\mathscr{P}(s_h, t_h))$ that can satisfy the demand, such that the set of vertices S_h forming the paths of $\mathscr{P}(s_h, t_h)$ is a bubble for the demand h, then the demand between s_h and t_h can be pruned on the paths of $\mathscr{P}(s_h, t_h)$ for an amount equal to $k_h \triangleq \min\{f^*(\mathscr{P}(s_h, t_h)), d_h\}$ without compromising the routability of the demand and without worsening the final solution in terms of recovered components.*

Proof. As the paths of $\mathscr{P}(s_h, t_h)$ form a bubble, any potentially conflicting demand which requires capacity from the links of the paths of $\mathscr{P}(s_h, t_h)$ should traverse the endpoints s_h and t_h. Let us consider a potentially conflicting demand (s_q, t_q) requesting at least $f^*(s_h, t_h) - k_h + \epsilon$ units of flow, so that it is conflicting with demand (s_h, t_h) for an amount of capacity exactly equal to ϵ. Due to the hypothesis of routability of the overall demand, if the conflicting demand of ϵ of the couple (s_q, t_q) is routed in $\mathscr{P}(s_h, t_h)$, there is an alternative set of paths of capacity at least ϵ which goes from s_h to t_h traversing the nodes of $V \setminus S_h$. Therefore such an alternative path can equivalently be assigned to (s_q, t_q) without harming the routability of the demand. In terms of routability the two solutions, routing either one or the other of the two conflicting demands, are alike. Nevertheless in terms of resource consumption, the bandwidth consumed to route the demand d_h over its bubble is lower than the one potentially consumed by routing the conflicting demand d_q over the bubble of d_h. In fact, if d_q is routed over the bubble of d_h, this last demand will require the traversal of more edges than d_q to reach the alternative path. Hence routing d_h will result in the same or in a lower number of repairs than with the corresponding alternative solution. □

Theorem IV.2. *ISP terminates in a finite time (the number of steps is polynomial in the input size).*

Proof. At each iteration, ISP performs either a repair, a split or a prune action. The number of repairs is limited by the number of broken network elements in the supply graph, that is $|V_B| + |E_B|$.

Let us consider the case of split actions. When a demand d_h between the pair (s_h, t_h), is split on the node v, ISP produces two new demand pairs for a flow d_x, namely (s_h, v) and (v, t_h), and updates the original pair to a demand $d - d_x$.

Let us consider the case of a partial split, where d_x is strictly lower than d. In such a case, d_x is the maximum value of splittable demand under the constraints given by Equations 2, with the updated demands. Due to the linearity of the problem, at least one capacity constraint acts as *binding constraint* of the linear programming problem, and is met with an equality in correspondence to the optimal. New partial splits will have new binding capacity constraints. As there is a capacity constraint for every edge, it follows that the number of partial splits is limited to the number of edges of the supply graph, that is $|E|$. This also shows that split actions can never produce infinitesimal demand values. This property is necessary to prove that also complete splits (which do not create binding capacity constraints) and pruning actions are executed a finite and limited amount of times.

We recall that the *surplus* [39] of a set of vertices $U \subset V$ is defined as: $\sigma(U) = \sum_{(i,j)\in\delta_{\mathbb{G}}(U)} c_{ij} - \sum_{(i,j)\in\delta_{\mathbb{H}}(U)} d_{ij}$, where $\delta_{\mathbb{G}}(U) = \{(i,j) \in E,\ \text{s.t. } |\{i,j\} \cap U| = 1\}$ is a cut determined by U on the supply graph; similarly the cut on the demand is $\delta_{\mathbb{H}}(U) = \{(i,j) \in E_{\mathbb{H}},\ \text{s.t. } |\{i,j\} \cap U| = 1\}$. We denote with $\sigma^{(n)}(v)$ the surplus, at iteration n, of the set formed by the single vertex $v \in V$. By using the properties of cuts given in [24] we can prove that the algorithm actions affect the value of the surplus of single vertices as follows (details are omitted due to space limitation): a split action of d demand units over the intermediate vertex v decreases the surplus of v for a value of $2d$, while it leaves the other individual vertex cuts unaltered; a prune action of a demand amount of d along a path p causes a decrease of $2d$ in the surplus of the nodes belonging to p that are not endpoints of the pruned demand and leaves all other individual vertex cuts unaltered. As routability is a requirement for any action of ISP the action preserves the cut condition and all surplus will be non negative (cut condition). Therefore the number of split of any demand d on a node v is bounded by $\lfloor \sigma(v)/2d \rfloor$ which is finite and limited. Finally, let us consider the effect of pruning actions. A prune action of a demand d to a path p reduces the capacity of the edges of p of an amount equal to $\min\{d, c(p)\}$. As the capacity of each edge is limited, the number of prune actions is also limited, as d is also finite. □

Theorem IV.3. *ISP has polynomial time complexity.*

Proof. Theorem IV.2 shows that ISP terminates in a polynomial number of iterations. Let us consider the individual activities for each iteration.

Complexity of routability test. Notice that the execution of the routability test requires to decide the feasibility of the set of inequalities on continuous variables (2), which has polynomial complexity, as detailed in [40], [30].

Notice that this complexity can be further reduced by considering the only incremental modifications of the routability problem at each iteration of the algorithm.

Complexity to calculate the demand based centrality ranking. If a static distance metric is adopted to measure the path length, the centrality of the nodes can be calculated offline

with Equation 3, and therefore does not affect the complexity of ISP. If the distance metric is dynamic, as described in Section IV-D, the sets $\mathscr{P}^*_{ij}$ cannot be calculated offline, so the demand based centrality is determined using the *estimated sets* $\hat{\mathscr{P}}^*_{ij}$ described in Section IV-B. The resulting complexity is $O(|E_H^{(n)}| \times |E^{(n)}| \times (|E^{(n)}| + |V^{(n)}| \log(|V^{(n)}|))$, since at each iteration we compute the shortest path between nodes i and j (complexity of the Dijkstra algorithm), which is either sufficient to route the entire demand $d(i,j)$ or at least one edge will be removed from the residual graph and a new shortest path will be considered. For each selected shortest path, we update the centrality of its nodes in linear time with respect to the path length. Thanks to this procedure, we can use Equation (3) to obtain an estimate of the centrality of each node.

Complexity of the split action. Finding the best candidate requires $O(|V|)$ steps. In order to select the demand to be split (Decision 1), we rank all demands that contributed to the centrality of the best candidate on the basis of Equation (8). Calculating the demand rank costs $O(|E_{\mathrm{H}}^{(n)}|$ times the calculation of the max flow between any demand pair, which is also polynomial. We can then select the demand with highest rank in $O(|E_{\mathrm{H}}|^{(n)})$. Solving the linear programming problem to calculate d_x (Decision 2), has also polynomial complexity, using the interior point method [30] and, depending on the iteration, is performed on problem instances of decreasing size.

Complexity of the recovery action. For each demand pair $(u,v) \in E_{\mathrm{H}}$, ISP checks if there exists a destroyed edge $(u,v) \in E$. The overall complexity is then $O(|E_{\mathrm{H}}|)$, using an adjacency matrix for E.

Complexity of the prune action. We can find the set of paths that form a bubble for each demand pair (s_h, t_h) by using a modified BFS visit starting from s_h. Such visit discards all paths that lead to a demand endpoint which is not s_h or t_h. Since the pruning action of a demand on a path can be performed in linear time with respect to the path length, the complexity of the pruning activity is the complexity of the visits, i.e. $O(|E_{\mathrm{H}}| \times (|V| + |E|))$. □

REFERENCES

[1] T. Sakano, Z. Fadlullah, T. Ngo, H. Nishiyama, and others, Disaster-resilient networking: a new vision based on movable and deployable resource units. *Network, IEEE*, 27(4), 2013.

[2] Y. Nemoto, and K. Hamaguchi, Resilient ICT research based on lessons learned from the Great East Japan Earthquake. *Communications Magazine, IEEE*, 52 (3), 2014.

[3] S. Bailey, Disaster Preparedness and Resiliency, in *Guide to Reliable Internet Services and Applications,* Editors C. Kalmanek, et al., Springer London, 2010.

[4] A. Todimala and B. Ramamurthy. Approximation Algorithms for Survivable Multicommodity Flow Problems with Applications to Network Design. *IEEE INFOCOM,* 2006

[5] A. F. Hansen, A. Kvalbein, T. Čičić, and S. Gjessing. Resilient Routing Layers for Network Disaster Planning. *International Conference on Networking (ICN 2005)*.

[6] A. Kvalbein, A. F. Hansen, T. Čičić, S. Gjessing, O. Lysne. Fast IP Network Recovery Using Multiple Routing Configurations. *IEEE INFOCOM 2006*.

[7] A. Kvalbein, A. F. Hansen, T. Čičić, S. Gjessing, O. Lysne. Multiple Routing Configurations for Fast IP Network Recovery. *IEEE/ACM Transactions on Networking,* 17(2), 2009

[8] M. Medard, S. G. Finn, R. A. Barry, R. G. Gallager. Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs. *IEEE/ACM Transactions on Networking,* 7(5), 1999

[9] Q. Zheng, G. Cao, T. La Porta and A. Swami. Optimal Recovery from Large-Scale Failures in IP Networks. *IEEE ICDCS,* 2012

[10] M. Suchara and D. Xu and R. Doverspike and D. Johnson and J. Rexford. Network Architecture for Joint Failure Recovery and Traffic Engineering. *ACM SIGMETRICS,* 2011

[11] H. Kerivin, A. R. Mahjoub. Design of Survivable Networks: A survey. *Networks,* 46(1), 2005

[12] M. F. Habib, M. Tornatore, F. Dikbiyik, B. Mukherjee. Disaster survivability in optical communication networks. *Computer Communications,* 36(6), 2013

[13] M. T. Gardner, R. May, C. Beard, D. Medhi. A Geographic Multi-Topology Routing approach and its benefits during large-scale geographically correlated failures. *Computer Networks,* 82, 2015

[14] S. Neumayer, A. Efrat, E. Modiano. Geographic max-flow and min-cut under a circular disk failure model. *Computer Networks,* 77, 2015

[15] C. Yang, H. Yu and G. Sun. Network recovery from large-scale failures considering the recovery resources and the upper limit of traffic demand. *International Conference on Advanced Intelligence and Awareness Internet (AIAI 2011)*.

[16] J. Wang, C. Qiao, and H. Yu. On progressive network recovery after a major disruption. In *IEEE INFOCOM*, 2011.

[17] A. Kumar, A. Gupta, and T. Roughgarden. A constant-factor approximation algorithm for the multicommodity rent-or-buy problem. In *IEEE Foundations of Computer Science (FOCS)*, 2002.

[18] L. Fleischer, J. Könemann, S. Leonardi, and G. Schäfer. Simple cost sharing schemes for multicommodity rent-or-buy and stochastic steiner tree. In *ACM Symposium on Theory of Computing (STOC)*, 2006.

[19] E. E. Lee, J. E. Mitchell, and W. A. Wallace. Restoration of services in interdependent infrastructure systems: A network flows approach. *IEEE Trans. on Systems, Man, and Cybernetics, Part C*, 37(6), 2007.

[20] A. Arab, A. Khodaei, Z. Han, and S. Khator. Proactive recovery of electric power assets for resiliency enhancement. *Access, IEEE*, 3, 2015.

[21] H. Ho and A. Sumalee. Optimal recovery plan after disaster. *Journal of Transportation Engineering*, 140(8), 2014.

[22] N. Bartolini, S. Ciavarella, T. La Porta and S. Silvestri, Network recovery after massive failures. *IEEE/IFIP DSN 2016*.

[23] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24. Springer Verlag, New York, NY, USA, 2003.

[24] A. Chakrabarti, L. Fleischer, and C. Weibel. When the cut condition is enough: A complete characterization for multiflow problems in series-parallel networks. In *ACM STOC*, 2012.

[25] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1), 1977.

[26] P. Bonacich. Power and centrality. *American Journal of Sociology*, 92(5), 1987.

[27] P. Hage and F. Harary. Eccentricity and centrality in networks. *Social Networks*, 17(1), 1995.

[28] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking. In *ACM World Wide Web (WWW)*, 1998.

[29] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM J. Comput.*, 37(2), 2007.

[30] B. Guenin, J. Könemann, and L. Tunçel. *A gentle introduction to optimization*. Cambridge University Press, United Kingdom, 2014.

[31] The internet topology zoo, *http://www.topology-zoo.org/*. Last access on May 2015.

[32] S. Knight, H. X. Nguyen, N. Falkner, R. A. Bowden, and M. Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9), 2011.

[33] S. Martello and P. Toth. *Knapsack problems. Algorithms and Computer Implementation*. John Wiley & Sons, England, 1990.

[34] Gurobi. http://gurobi.com. Last access on Nov. 2016.

[35] CenturyLink, IP/MPLS Networking, *http://www.centurylink.com/business/asset/network-map/ip-mpls-network-nm090930.pdf*. Last access on Nov. 2016.

[36] P. Erdos and A. Renyi, *On Random Graphs*, Publicationes Mathematicae, 1959.

[37] Aurora Fiber Optic Networks, *http://www.aurorafonet.com/wp-content/uploads/2015/06/auroramap.png*. Last access on Nov. 2016.

[38] M. Hauptmann and M. Karpinski. A compendium on Steiner tree problems, *http://theory.cs.uni-bonn.de/info5/steinerkompendium/*. Last access on Nov. 2016.

[39] H. Okamura and P. Seymour. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory, Series B*, 31(1), 1981.

[40] N. Megiddo. On the complexity of linear programming. *Advances in Econ. Theory*, 1987.