Network recovery after massive failures

Novella Bartolini and Stefano Ciavarella
Department of Computer Science
University Sapienza of Rome, Italy
Email: {bartolini,ciavarella}@di.uniroma1.it

Thomas F. La Porta
Department of Computer Sc. and Eng.
Pennsylvania State University, USA
Email: tlp@cse.psu.edu

Simone Silvestri
Department of Computer Science
Missouri Science and Technology, USA
Email: silvestris@mst.edu

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. 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 number 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 [1] at a large scale. Because society has come to depend heavily on communication networks to support mission critical services, especially in times of emergency, it is critical that such infrastructures be repaired quickly, at least to the point where mission critical services can be supported.

A widespread collapse of critical infrastructures occurred after Hurricane Katrina hit the Gulf Coast of the United States, in 2005. The damage extended for an area of approximately 93,000 square miles. More than 2,000 cell towers were knocked out. The backbone conduit for landline service was flooded as well as many central switching centers [2], [3].

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 almost all terrestrial communication infrastructures including many of the wired communication networks and emergency municipal radio communication systems [4], [5].

In both cases, 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. For this reason, a major challenge in disaster management scenarios is to *sufficiently recover* the communication network infrastructure so that it may support mission critical applications in the shortest time and with minimum interventions.

In this paper 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. We address the problem of fulfilling the requirements of the communications network through the restoration of network components. Our goal is to optimize the recovery actions, in order to obtain the restoration of the required capacity to support mission critical services at minimum cost.

We model the mission critical services as a *demand graph* which takes account of the demand increase consequent to the occurred incidents [6]. This graph defines a set of *demand flows* on the communication network, to which we refer to as the *supply network*. We consider scenarios in which a major disruption of the supply network makes it unable to meet the capacity requirements of demand flows. Therefore, the flows must be accommodated by means of recovery actions or deploying new links and nodes.

We model the recovery problem in terms of mixed integer linear programming. 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 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 meant to measure the importance of a node in a supply graph given the demand flows. 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,

This work was supported by the Defense Threat Reduction Agency under the grant HDTRA1-10-1-0085. The work of Novella Bartolini was sponsored by the U.S. Army Research Laboratory under Agreement Number W911NF-14-0610.

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.

We formally prove that ISP terminates in a finite number of steps by returning both a recovery strategy and a routing solution for the demand flows.

We also propose other heuristics to the recovery problem, based on the standard multi-commodity approach as well as greedy approaches. We compare the performance of ISP and the other heuristics against the optimal solution under a variety of scenarios. Such scenarios include both real and synthetic network topologies, geographically correlated failures, as well as different demand requirements. Results show that ISP always outperforms all the other heuristics. In particular, it performs very close to the optimal when the demand is relatively low with respect to the network capacity. We also compare the algorithms in terms of execution time, showing that ISP provides solutions for complex cases in the order of 5 minutes, whereas the optimal solution takes on the order of 27 hours.

In summary the original contribution of the paper is the following:

- We formulate a recovery problem as an MILP and show its NP-hardness.
- We introduce a new metric of demand based centrality, specifically meant to measure the importance of a node in a supply graph of a multi-commodity problem instance.
- We propose a polynomial time heuristic called ISP, which uses the new centrality to address the recovery problem.
- We propose other heuristics based on the standard multi-commodity approach, as well as greedy heuristics and shortest paths repair approaches, as baseline solutions.
- We evaluate the proposed solutions through simulations on real and synthetic topologies, under geographically correlated failures. 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

While there is a considerable amount of research on recovery from single or sparse failures in a network, our paper addresses the problem of network recovery from large scale failures. Hence, in this section, we do not consider the previous work on the first problem, and describe only works that are related or are applicable to the case of massive failures.

The work by Wan, Qiao and Yu [32] introduced a problem related to ours. They 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 achieved throughput. The authors modeled the problem as an MILP and showed that is it NP-hard. They proposed a greedy heuristic for solving the problem in multiple stages by analyzing the shadow prices of the related optimization problem and using an iterative evaluation of these values to repair the edges with highest potential for contributing to the objective function. Unlike this work which aims at optimizing throughput over time, we aim at optimizing costs of recovery under constraints on quality of service. Moreover, our algorithm also produces a routing solution that guarantees that the demand flows are actually accommodated.

The multi-commodity flow problem, addressed in a large amount of research work, aims at finding the routing of several multi-commodity flows in a supply network, so as to optimize the totally routed flow. This problem seems the most reasonable reduction of our problem to a classic problem. Nevertheless this approach has considerable limitations when applied to the problem of recovery. We discuss these aspects in detail in Section VI-A. Many heuristics have been proposed to solve several variants of the multi-commodity flow problem. Most of these works [17], [9] rely on the idea that a higher total flow can be obtained by balancing the load distributing the flow over many paths. This idea is opposite to what is needed in the recovery problem, where we want to maximize the flow traversing repaired paths, and concentrate the flow towards shared paths.

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. [24] and Fleischer et al. [15] 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, in their work [25], Lee et al. address 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 [8], [22] address the problem of recovery beyond the field of telecommunications with solutions tailored to the specific type of network being considered.

Finally, the work of Magnanti et al. [26] addresses the problem of network design under connectivity only requirements. It shows that the simplified version of our main problem in which every demand pair requires only to be connected regardless of the capacity of the interconnecting paths, is a specific instance of the Steiner Forest problem.

III. The Network Recovery Problem

In this section we formulate the Minimum Recovery (MinR) problem as a mixed integer linear optimization problem. MinR aims at minimizing the cost to repair broken nodes and links so as to restore the necessary network capacity to meet a given demand.

Table I summarizes the notation used throughout the paper. 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 consider a *demand*

Notations	**Descriptions**
$G=(V,E)$	supply graph
$G^{(n)}=(V^{(n)},E^{(n)})$	supply graph at iteration n
$H=(V_{\mathrm{H}},E_{\mathrm{H}})$	demand graph
$H^{(n)}=(V_{\mathrm{H}}^{(n)},E_{\mathrm{H}}^{(n)})$	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 $(s_h,t_h)\in E_H$
$c_{ij}^{(n)}$, $d_{s_h t_h}^{(n)}$	capacity of (i,j), demand of (s_h,t_h) at the n=th iteration
$V_{\mathrm{B}}\subseteq V$ and $E_{\mathrm{B}}\subseteq E$	broken vertices and edges
$V_{\mathrm{B}}^{(n)}$ and $E_{\mathrm{B}}^{(n)}$	V_{B} and E_{B} at iteration n
$h\in E_{\mathrm{H}}$,	demand pair $(s_h,t_h)\in E_{\mathrm{H}}$
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 $i\in V$
$\eta_{\mathtt{max}}$	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
$n(p)$	number of edges of p
$c(p)$	capacity of path p: $\min\limits_{e\in p} c_e$
$\mathscr{P}(i,j)$	paths in G between i and j
$\mathscr{P}^*(i,j)$	shortest paths necessary to route demand d_{ij}
$\mathscr{P}_{ij}^*\|_v$	set of paths in $\mathscr{P}_{ij}^*$ that include v
$c_d(v)$	demand based centrality, see (3)
$v_{\mathrm{BC}}^{(n)}$	node with highest centrality at iteration n
$\mathcal{C}^{(n)}(v_{\mathrm{BC}}^{(n)})\subseteq E_{\mathrm{H}}^{(n)}$	demand pairs that contributed to the centrality of $v_{\mathrm{BC}}^{(n)}$, updated at iteration n
$\mathcal{L}(n)$	list of repairs, updated at iteration n

TABLE I
NOMENCLATURE AND NOTATION

graph $H=(V_{\mathrm{H}},E_{\mathrm{H}})$, where $V_{\mathrm{H}}\subseteq V$, and $E_{\mathrm{H}}\subseteq V_{\mathrm{H}}\times V_{\mathrm{H}}$ is the set of pairs of nodes in V_{H} having a positive flow demand. Each pair $(s_h,t_h)\in E_{\mathrm{H}}$ 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 $h\in E_{\mathrm{H}}$, when $(s_h,t_h)\in E_{\mathrm{H}}$, and we shortly use the notation d_h for d_{s_h,t_h} when the context allows. Notice that the demand flows modeled by the sets V_{H} and E_{H} can take emergency related priorities into account. These sets define the endpoints of critical communication services and an estimate of the related demand flow, which may account for the increased needs due to the disaster [6].

In order to model the network failure, we define the sets $V_{\mathrm{B}}\subseteq V$ and $E_{\mathrm{B}}\subseteq E$ of damaged vertices and edges, respectively. We denote with k_i^v the cost of repairing vertex $i\in E_{\mathrm{B}}$ and with k_{ij}^e the cost of repairing the edge $(i,j)\in E_{\mathrm{B}}$[1]. The recovery costs are heterogeneous and dependent on the location and on the technology in use.

We then 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. Notice that other flows may traverse the same edge in the opposite direction.

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 $(i,j)\in E_{\mathrm{B}}$, 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 $i\in V_{\mathrm{B}}$.

The objective function of the MinR problem can be expressed as in 1(a), where we optimize the cost of repairing the only vertices and edges that are both used (the corresponding binary decision variable is 1) and that were initially broken (the related vertices and edges belong to V_{B} and to E_{B}, respectively).

The capacity constraint of our problem is expressed by 1(b). According to this constraint the total amount of flow traversing the edge (i,j) in both directions cannot 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 $\eta_{\mathtt{max}}$ of the network. Therefore the relationship between δ_i and δ_{ij} can be expressed by the constraint given by 1(c).

We consider a flow balance constraint, in the form expressed by 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. Finally, 1(e) shows that we are considering non negative, continuous decision variables for the flow assignment to edges, while 1(f) expresses the binary constraint for the decision variables which determines whether some vertices and edges are used in the solution of the problem.

The MinR problem can therefore be formulated in linear terms in the variables δ_{ij}, δ_i and f_{ij}^h as follows:

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

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

Proof. Let us consider a generic instance of the Steiner Forest problem [21], [26]. 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 $c_{\mathtt{sf}}:E\rightarrow\mathbb{R}^+$, 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 $E_{\mathrm{B}}=E$ and $V_{\mathrm{B}}=\emptyset$. 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

[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. The model can also be *extended* to the case of multiple choices for link technology and related capacity. For simplicity of presentation, in this paper we refer to the only case of recovery decisions.

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 therefore conclude the reducibility of the Steiner Forest problem to MinR, and consequently that the problem MinR is NP-Hard. □

IV. Iterative Split and Prune

The algorithm ISP (Iterative Split and Prune) works by iteratively selecting the best candidate nodes and links for repair, then simplifying 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. Notice that at the end of its execution the algorithm ISP will output both the set of repairing interventions and the corresponding routing of demands.

The pseudo-code of the algorithm is shown in Algorithm 1. More details on the single 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 $E_B$
1  while routability test fails do
2      while pruning condition do
3          Prune demands satisfying pruning condition;
4          Update G and H;
5      if there are repairable links then
6          Repair broken repairable links;
7          Update G and $E_B$;
8      else
9          Find best candidate $v_{BC}$ for split;
10         Repair $v_{BC}$ if broken;
11         Find best demand $d$ to split on $v_{BC}$;
12         Calculate the maximum splittable amount $d_x$;
13         Split amount $d_x$ of demand $d$ on $v_{BC}$;
14         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 should terminate 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 [31], 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 no less 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 [14], or a *bad-k4-pair* as discussed in the already mentioned work by Schijver [31].

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*:

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

If the constraint system given by the routability conditions of (2) 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 $V^{(n)} = V \setminus V_B^{(n)}$, and $E^{(n)} = (E \setminus E_B^{(n)}) \setminus \{(i,j) \text{ s.t. } |\{i,j\} \cap V_B^{(n)}| \geq 1\}$. 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 core actions of ISP rely on a ranking among nodes based on a novel *demand based centrality* metric. Unlike previous definitions of node centrality [16], [13], [10], [20], [30], our metric takes account of the ability of each node to accommodate flow throughout the network.

The metric extends the notion of *betweeness centrality* [16], [13] as follows. A path p in the graph G is hereby defined as a list of composing 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. Such a length can be defined in several ways, as discussed in Section IV-D. The capacity of a path is denoted by $c(p)$ and is equal to the minimum capacity of the links in p.

We denote with $\mathscr{P}(i,j)$ the set of acyclic paths in G connecting nodes $i,j \in V$. 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:

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

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), and therefore it does not affect the complexity of ISP. If otherwise, this pre-calculation is not available, the demand based centrality can be calculated at runtime, with some approximation, as follows. For each demand d_{ij}, with endpoints $(i,j) \in E_{\mathrm{H}}$, we calculate iteratively the set of shortest paths $\hat{\mathscr{P}}^*_{ij}$, which estimates the actual set $\mathscr{P}^*_{ij}$ to reduce the complexity. Initially, $\hat{\mathscr{P}}^*_{ij} = \emptyset$. We iteratively use Dijkstra's algorithm to find the shortest path p between i and j, and add p to $\hat{\mathscr{P}}^*_{ij}$. If the paths in $\hat{\mathscr{P}}^*_{ij}$ have enough capacity, i.e. $\sum_{q \in \hat{\mathscr{P}}^*_{ij}} c(q) \geq d_{ij}$, then these paths are 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 at the next iteration to satisfy a demand $d_{ij} - \sum_{q \in \hat{\mathscr{P}}^*_{ij}} c(q)$.

For each selected shortest path, 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, with an expression analogous to 3.

Notice that, the calculation of the centrality based ranking is performed at each iteration considering the supply graph G (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 $v^{(n)}_{\mathrm{BC}} \in V$ 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 $v^{(n)}_{\mathrm{BC}} \in V^{(n)}_{\mathrm{B}}$, then $v^{(n)}_{\mathrm{BC}}$ is virtually repaired at the current iteration, therefore it is removed from the set $V^{(n)}_{\mathrm{B}}$ and it is added to the *list of items to be repaired*, referred to with $\mathcal{L}(n)$. Notice that once an element is inserted in the list $\mathcal{L}(n)$ it is thereafter considered by the algorithm as if it were already repaired (more details on this list can be found in Section IV-E).

The next step of the algorithm ISP is the split of a demand flow over the node $v^{(n)}_{\mathrm{BC}}$. Let us consider a split action occurring at the n-th iteration. Let us consider also a demand pair $(s_h, t_h) \in E^{(n)}_{\mathrm{H}}$ of value $d^{(n)}_h$. *Splitting* d_x *units of the demand* $d^{(n)}_h$, with $d_x \leq d^{(n)}_h$ 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 $(s_h, v^{(n)}_{\mathrm{BC}})$ and $(v^{(n)}_{\mathrm{BC}}, t_h)$.

Fig. 1 illustrates the described split action.

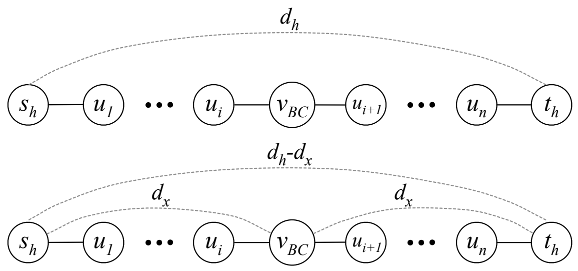


Fig. 1. Split of d_x units of demand

The set of demand couples $E^{(n)}_{\mathrm{H}}$ will be updated as follows

$$E^{(n+1)}_{\mathrm{H}} = \{(s_h, v^{(n)}_{\mathrm{BC}}), (v^{(n)}_{\mathrm{BC}}, t_h)\} \cup E^{(n)}_{\mathrm{H}}. \qquad (4)$$

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

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

while

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

and the new demand pairs have the following flows:

$$d^{(n+1)}_{zw} \quad = \quad d_x \text{ if } (z,w) = (s_h, v^{(n)}_{\mathrm{BC}}) | (v^{(n)}_{\mathrm{BC}}, t_h). \qquad (7)$$

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 $v^{(n)}_{\mathrm{BC}}$ 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 $v^{(n)}_{\mathrm{BC}}$ with d_x units of the original demand $d^{(n)}_h$. The algorithm ISP can be tuned to perform this action according to several criteria, to address two different aspects following the selection of the vertex $v^{(n)}_{\mathrm{BC}}$: (1) which demand should be split, and (2) the amount of flow to split.

Let $\mathcal{C}^{(n)}(v^{(n)}_{\mathrm{BC}}) \in E^{(n)}_{\mathrm{H}}$ be the set of demand pairs that positively contributed to the centrality value of the node $v^{(n)}_{\mathrm{BC}}$

at the current iteration, that is:

$$\mathcal{C}^{(n)}(v_{\text{BC}}^{(n)}) = \bigcup_{(i,j)\in E_{\text{H}}^{(n)}} \{(i,j) \text{ s.t. } \mathscr{P}^*(i,j)|_{v_{\text{BC}}^{(n)}} \neq \emptyset\}.$$

Decision (1): 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

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

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 the constraints of $d_x \leq d_h^{(n)}$ and to the flow conservation and capacity constraints defined by (2), where the set $E_{\text{H}}^{(n)}$ is defined according to (4), and the demand flows are defined according to (5), (6) and (7).

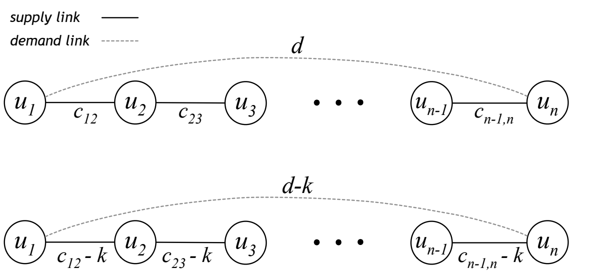


Fig. 2. Pruning of k units of demand

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 $l^{(n)}(e_{ij}) = [const + k_{ij}^e(n) + (k_i^v(n) + k_j^v(n))/2]/c_{ij}$, 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, allowing the concentration of demand flows on the repaired components.

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. Therefore ISP considers a *list of items to be repaired* $\mathcal{L}(n)$, 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 list of repairs, so $\mathcal{L}(n+1) = \mathcal{L}(n) \cup \{v_{\text{BC}}\}$, 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 link directly connects two endpoints of a demand, and such demand cannot be satisfied by the current repairs. Formally, if 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 $\mathcal{L}(n)$), 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 list of repairs, that is $\mathcal{L}(n+1) = \mathcal{L}(n) \cup \{(s_h, t_h)\}$. 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. Fig. 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 $V_{\mathrm{H}}^{(n)}$.

It must be noted that, like the splitting action, the pruning action implies a routing decision which may possibly lead to an unfeasible 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 2 (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 3 (Prune conditions)**.** *Consider a supply graph G and a demand graph H, which satisfy the routability conditions given by (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. □

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.

V. Properties of ISP

Theorem 4. *Algorithm ISP terminates in a finite number of steps which 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 2, with the updated demands. Every time such a problem is executed, 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* [29] of a set of vertices $U \subset V$ is defined as: $\sigma(U) = \sum_{(i,j)\in\delta_{\mathrm{G}}(U)} c_{ij} - \sum_{(i,j)\in\delta_{\mathrm{H}}(U)} d_{ij}$, where $\delta_{\mathrm{G}}(U) = \{(i, j) \in E,\ \text{s.t.}\ |\{i, j\} \cap U| = 1\}$ is a cut determined by U on the supply graph; similarly the cut on the demand is $\delta_{\mathrm{H}}(U) = \{(i, j) \in E_{\mathrm{H}},\ \text{s.t.}\ |\{i, j\} \cap U| = 1\}$. 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 [14] 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 splits 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 $\min\{d, c(p)\}$. Therefore, as the capacity of each edge is limited, the number of prune actions is also limited, as d is always finite. □

Theorem 5. *The computational complexity of ISP is polynomial.*

Proof (sketch). The proof is omitted due to space limitation. It follows from Theorem 4 which shows that ISP terminates in a polynomial number of iterations and from the analysis of the complexity of the algorithm activities at each iteration, which is also polynomial. □

VI. Heuristics

A. A multi-commodity based solution

One way to address the problem of finding the subset of broken components to be recovered is to minimize the amount of flow that makes use of broken links.

$$\begin{array}{lll} \min \sum_{(i,j)\in E_{\mathrm{B}}} k^e_{ij} \cdot \sum_{h\in E_{\mathrm{H}}} f^h_{ij} & & (a) \\ \sum_{h\in E_{\mathrm{H}}} (f^h_{ij} + f^h_{ji}) \le c_{ij} & \forall (i,j) \in E & (b) \\ \sum_{j\in V} f^h_{ij} = \sum_{k\in V} f^h_{ki} + b^h_i & \forall (i,h) \in V \times E_{\mathrm{H}} & (c) \\ f^h_{ij} \ge 0 & \forall (i,j) \in E, h \in E_{\mathrm{H}} & (d) \end{array} \quad (8)$$

In terms of recovery decisions, this approach repairs only those broken links and vertices that are actually used by the optimal solution. Notice that this new problem is a particular instance of the Multi-Commodity Flow problem.

Under this formulation, which is a relaxation of the problem of (1), the problem is no longer NP-hard, but has polynomial time complexity, being it solvable efficiently with LP methods such as the interior point method [18].

Nevertheless, the multi-commodity flow 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. Fig. 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 are obtained with the Bell-Canada topology [7], [23] by increasing the demand flow per pair under the experimental setting explained in Section VII. The results show that the multi-commodity approach has a wide solution space, which includes solutions close to the optimum as well as solutions equivalent to repairing all broken elements. Note that finding MCB among the wide set of solutions is NP-hard, being it an instance of MinR. For this reason we do not include the multi-commodity approach in our results.

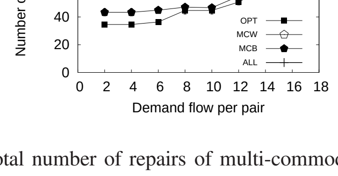


Fig. 3. Total number of repairs of multi-commodity solution

B. Shortest Path Heuristic (SRT)

This heuristic is based on a very intuitive approach to the MinR problem, that is to consider all the demand pairs (s_i, t_i, d_i) in decreasing order of demand d_i, and repair all the shortest paths that are necessary to meet the demand requirements. 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. According to SRT, for each demand d_i, all broken nodes and edges in S_i are repaired. The pseudo-code of SRT is shown in Algorithm SRT.

Algorithm SRT

```
   Input: $G$, $H$, $V_{\mathrm{B}}$ and $E_{\mathrm{B}}$
1  Sort demand pairs in $E_{\mathrm{H}}$ in decreasing order of $d_i$;
2  for $i = 1, \dots, |E_H|$ do
3      Calculate the set $S_i$ for the demand pair $(s_i, t_i, d_i)$;
4      Repair nodes and links of all paths in $S_i$;
```

Notice that the sets of shortest paths of different demands may overlap, 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 between a demand pair in H. The cost of repairing such path is the weight of the corresponding knapsack object, while the path capacity is the object value. Both heuristics make use of the set $P(H, G)$ of all simple paths between 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 Knapsack analogy, we can formulate two different heuristics based on the greedy approach to Knapsack [27].

Algorithm GRD-COM

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

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, d_i) the demand pair for which p was included in $P(H, G)$. 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. The pseudo code is shown in Algorithm GRD-COM.

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-NC

```
  Input: $G$, $H$, $V_B$, and $E_B$
1 Calculate (offline) $P(H, G)$;
2 for $p \in P(H, G)$ do $w(p) = \frac{cost(p)}{capacity(p)}$;
3 Sort paths according to their weight;
4 while routability test fails do
5     Repair the next path $p$;
```

The second heuristic is called *Greedy 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 unlike GRD-COM. 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. The pseudo code is shown in Algorithm GRD-NC.

Note that unlike GRD-COM, 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.

VII. Experiments

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

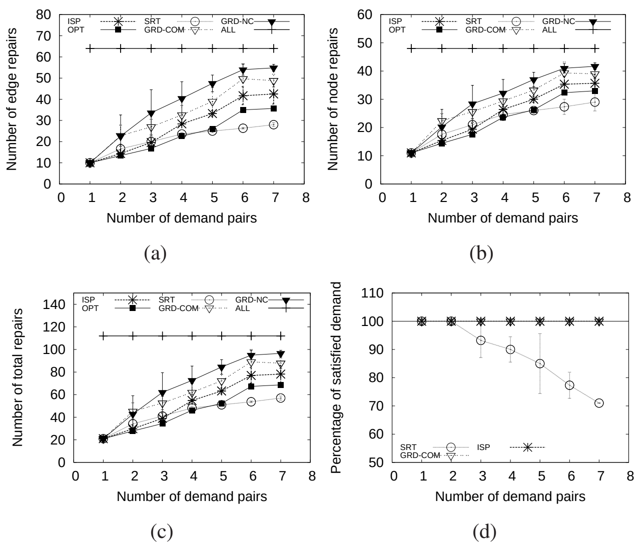


Fig. 4. Bell-Canada topology. Varying number of demand pairs (10 flow units/pair). Repaired edges (a), repaired nodes (b), total repairs (c) and demand loss (d).

We start the analysis with a real network topology of small size so that optimal solutions may be obtained in a reasonable time, and to provide a thorough experimental comparison of all the algorithms discussed in this paper. A second experimental scenario is based on synthetic topologies of varying complexity, to study the computational time of the proposed heuristics and of the optimal solution. We will evidence the poor scalability of the optimal approach, motivating the need to resort to heuristic solutions. In a last scenario we instead show the results on a real large size topology, to evidence the good approximation of ISP to the optimal solution even with a large problem size.

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

A. First scenario

In this set of experiments we consider the Bell-Canada topology, taken from the Internet Topology Zoo [7], [23] collection. This network has 48 nodes and 64 edges. The data set provides uniform edge capacities, which we manually altered to consider non homogeneous capacities. In particular we consider two backbones with capacity 30 and 50, respectively, while all remaining edges have capacity 20. 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 which have a hop distance greater than or equal to half the diameter of the network.

We perform three sets of experiments. 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 flow per pair. In both these experiments, we considered a complete destruction of the supply graph, in order to have the maximum range of potential solutions. On the contrary, in the third set of experiments (Section VII-A3) we consider different failure scenarios according to a geographically correlated failure model.

1) Variation of the number of demand pairs: In these experiments we increase the number of demand pairs from 1 to 7, and each demand pair has a requirement of 10 flow units. Fig. 4(a) and (b) show the number of edges and nodes repaired by the considered approaches, respectively. Fig. 4(c) shows the cumulative number of repairs. In the figures, the line ALL refers to the total number of destroyed nodes and links.

The experiments shown in Fig. 4(a)-(c) highlight that by linearly increasing the number of demand pairs, the number of repaired edges and vertices also grows.

ISP is the closest to the optimal among the considered heuristics. In the most critical setting, with 7 demand pairs, OPT repairs 37 edges, ISP repairs 42 edges, while GRD-COM repairs 49 edges and GRD-NC repairs 55 edges. The number of repaired vertices are consequently proportional, as in this experimental scenario the entire network is damaged by the destruction. We highlight that the greedy solutions 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 Fig. 4 (a)-(c), when the number of demand pairs is less than 4.

As the figures show, SRT results in the lowest number of repairs, however SRT, and similarly GRD-COM, does not ensure that all demand flows can be routed. In particular, SRT repairs the number of shortest paths up to the minimum necessary to satisfy each demand, treating demands independently. As the number of demand pairs increases, the paths selected by SRT are more likely to be shared. Therefore when these shared paths are saturated, the policy SRT is not able to satisfy all demands, as Fig. 4(d) shows. In these experiments, this occurs when the number of pairs grows from 2 to 3. This behavior reflects the fact that two pairs can be commonly routed on a path of capacity 20 units, but when 3 demand pairs require 30 units, the shortest paths may have some edge in common and a portion of demand is lost. These arguments explain the initial constant behavior of the demand loss shown in Fig. 4(d) and in the analogous figures of the following sets of experiments. Due to the similarity of the behavior of the policy SRT in all the following experiments, we will not comment on this policy any longer.

2) Variation of the demand intensity: In this section we introduce a dual experiment in which we fix the number of demand pairs to 4, and we vary the intensity of demand per pair. Fig. 5(a) and (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 experiments. Nevertheless there are some aspects worthy of note.

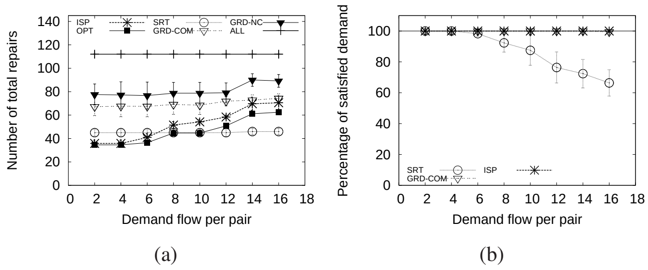


Fig. 5. Bell-Canada topology. Varying the intensity of demand flow (4 demand pairs). Total repairs (a), demand loss (b).

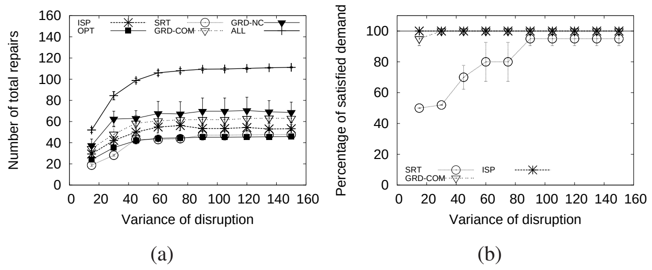


Fig. 6. Bell-Canada topology. Varying the extent of destruction (4 demand pairs, 10 flow units/pair). Total repairs (a) and demand loss (b).

Even if the global demand increase of this experiment is the same of the previous experiment, all policies tend to reveal a smoother increase in the number of repairs when the number of demand pairs is fixed. This is due to the need to repair damaged elements to at least connect the demand pairs, even when the demand intensity is low with respect to the link capacity. Such repairs are sufficient until the demand reaches an intensity for which more repairs are needed. This justifies the step-wise behavior of OPT and ISP.

The above reasoning helps understanding the trend of the greedy heuristics with respect to the intensity of the demand. These approaches blindly repair paths with high rank until all demands are satisfied. When the demand intensity is low, and basically only connectivity between demand pairs is needed, these heuristics still repair all paths in the list which have a higher rank than those required for connectivity. As the demand increases, this high number of paths is still sufficient to serve the demand, and hence further repairs are not needed. However, when the demand increases further, a bunch of additional paths are repaired, as shown in Fig. 5(a) in correspondence of the increase in demand intensity from 12 to 14 for the heuristic GRD-NC.

3) Variation of the extent of destruction: In this experiment we consider the impact of the extent of destruction. We consider a geographical failure model, to represent natural disasters and intentional attacks. We generated the disruption according to a bi-variate Gaussian distribution of the disruption probability of network components. We varied the variance of such a distribution and scaled the probability accordingly to obtain larger failures with larger variance.

In these experiments we consider 4 demand pairs, each with a demand intensity of 10. We consider an increase in the amount of disrupted components obtained by varying the variance of the disruption. We consider the epicenter at the barycenter of the nodes in the network, and same variance in both dimensions of the bi-variate distribution of failures.

Fig. 6(a) and (b) show the total number of repaired elements and the percentage of demand loss, respectively. The line labeled ALL shows how many edges or vertices are disrupted in the considered instance of the problem.

Even in this setting we observe similar behavior of the considered policies, which highlights the superiority of ISP. In particular, ISP performs close to the optimal, and when the network is almost completely destroyed (i.e. a variance equal to 150) ISP repairs only 53 elements, with respect to the 46 elements repaired by the optimal solution, whereas GRD-COM requires 63 repairs and GRD-NC requires 68 repairs.

B. Second scenario

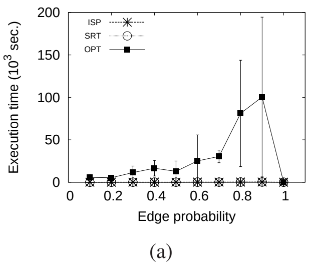
In this scenario, we analyze the scalability of ISP and OPT. We consider synthetic network topologies of increasing complexity and we evaluate the performance and the computation time of the two algorithms.

We considered an Erdos-Renyi topology [12] with 100 nodes. We recall that in an Erdos-Renyi graph, any two nodes are connected through an edge with probability p (*edge probability*). In the experiments of Fig. 7 we varied the parameter p.

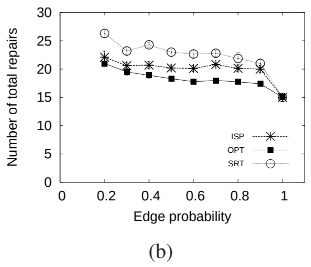
As the purpose of this set of experiments is to evaluate the algorithm scalability, we consider a relatively simple problem instance in which we have only a connectivity requirement, with a construction similar to the one of the proof of Theorem 1 (an instance of the Steiner Forest problem).

We modeled the link capacity and flow demand as follows: we considered 5 demand pairs, of one unit each, and we analyzed the case of a completely destroyed network, where each link has a capacity of 1,000 units of flow. Despite the relative simplicity of the problem formulation (only connectivity requirements), by growing p we increase the difficulty of the problem.

In Fig. 7(a) we focus on the execution time of ISP and OPT. For the calculation of the optimal solution we implemented problem 1 using Python and the Gurobi [19] library, which is known for its efficiency. For these experiments we used a 20



(a)



(b)

Fig. 7. Erdos-Renyi topology. Varying edge probability p. Execution time (a), number of total repairs (b).

Fig. 8. CAIDA topology AS28717, with 825 nodes and 1018 edges.

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 edges connecting them.

Notice that, when p grows, the graph becomes non planar and in the case of non-planar graphs, the Steiner Forest problem is known to be APX-hard [21], hence we do not expect a good approximation of the optimal solution.

In fact, Fig. 7(b) shows that the gap between ISP and OPT is much higher than in the other experiments which used real topologies. This is because, as observed in [11], real topologies are typically planar or mostly planar. Nevertheless, ISP is still repairing a number of elements close to the optimal, and lower than the number of repairs under SRT. Notice also that in the case of 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.

For these experiments, we do not show the demand loss, as under this scenario, the link capacity is so high that none of the heuristics has any 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 this knowledge would require $O(N!)$ steps.

C. Third scenario

For this final set of experiments, we consider the real topology AS28717 of Fig. 8, taken from the CAIDA (Center for Applied Internet Data Analysis) resource collection

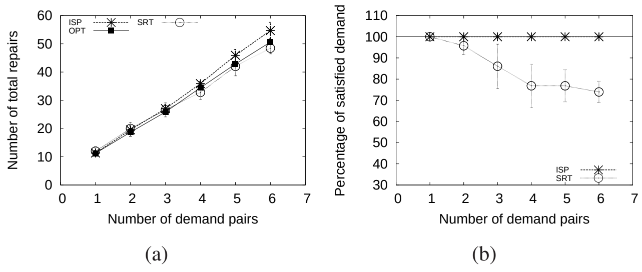


Fig. 9. CAIDA topology AS28717. Varying the number of demand pairs (22 flow units per pair). Total repairs (a), demand loss (b).

[33]. This topology represents IP-level connections between backbone/gateway routers of several ASs from major Internet Service Providers (ISPs) around the globe. Since CAIDA topologies are often disconnected, we selected the giant connected component, which has 825 nodes and 1018 edges. We consider 22 units of flow per demand and vary the number of demand pairs.

Fig. 9(a) shows the total number of repairs, while Fig. 9(b) shows the demand loss. ISP performs close to the optimal, and does not show any demand loss. The number of repairs under heuristic SRT is also comparable to the optimal, but the demand loss in this case is considerably high. We did not run the greedy heuristics in this experiment, as despite their simplicity, they do not scale to large topologies.

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 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 concentrate 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. 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.

Acknowledgments

The authors would like to thank Lisa Fleischer for a discussion on some details of the proofs. They also would like to thank Nicola Galesi for reading a preliminary version of the manuscript and suggesting ways to improve the presentation.

References

[1] D. Mendonça. Decision support for improvisation in response to extreme events. *Decision Support Systems*, 43(3), 2007.

[2] W. Leavitt, J. Kiefer. Infrastructure interdependency and the creation of a normal disaster the case of hurricane Katrina and the city of New Orleans. *Public works management & policy*, 10(4):306–314, 2006.

[3] R. Miller. Hurricane Katrina: Communications & Infrastructure Impacts. *Defense and Technical Information Center (DTIC) Document*, 2006.

[4] 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.

[5] Y. Nemoto, and K. Hamaguchi, Resilient ICT research based on lessons learned from the Great East Japan Earthquake. *Communications Magazine, IEEE*, 52 (3): 38 – 43, 2014.

[6] S. Bailey, Disaster Preparedness and Resiliency, in *Guide to Reliable Internet Services and Applications,* Editors C. Kalmanek, et al., Springer London, pp. 517 – 543, 2010.

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

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

[9] B. Awerbuch, and R. Khandekar. Greedy distributed optimization of multi-commodity flows. *Distributed Computing*, 21(5), 2009.

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

[11] R. Bowden, H. Nguyen, N. Falkner, S. Knight, and M. Roughan. Planarity of data networks. Teletraffic Congress (ITC), 2011.

[12] P. Erdos and A. Renyi, *On Random Graphs*, Publicationes Mathematicae, pp. 290-297, 1959.

[13] U. Brandes. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25(2), 2001.

[14] 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 Symposium on Theory of Computing (STOC)*, 2012.

[15] 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.

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

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

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

[19] Gurobi. http://gurobi.com. Last accessed on November 2015.

[20] P. Hage and F. Harary. Eccentricity and centrality in networks. *Social Networks*, 17(1):57 – 63, 1995.

[21] M. Hauptmann and M. Karpinski. A compendium on Steiner tree problems, *http://theory.cs.uni-bonn.de/info5/steinerkompendium/*. Last accessed on May 2015.

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

[23] 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):1765–1775, 2011.

[24] 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.

[25] 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.

[26] T. L. Magnanti and S. Raghavan. Strong formulations for network design problems with connectivity requirements. *Networks*, 45(2), 2005.

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

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

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

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

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

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

[33] The Cooperative Association for Internet Data Analysis (CAIDA). *Macroscopic Internet Topology Data Kit (ITDK)*, In http://www.caida.org/data/active/internet-topology-data-kit/, 2013.