Progressive Damage Assessment and Network Recovery after Massive Failures

S. Ciavarella, N. Bartolini, H. Khamfroush, and T. La Porta,

***Abstract*—After a massive scale failure, the assessment of damages to communication networks requires local interventions and remote monitoring. While previous works on network recovery require complete knowledge of damage extent, we address the problem of damage assessment and critical service restoration in a joint manner. We propose a polynomial algorithm called Centrality based Damage Assessment and Recovery (CeDAR) which performs a joint activity of failure monitoring and restoration of network components. CeDAR works under limited availability of recovery resources and optimizes service recovery over time. We modified two existing approaches to the problem of network recovery to make them also able to exploit incremental knowledge of the failure extent. Through simulations we show that CeDAR outperforms the previous approaches in terms of recovery resource utilization and accumulative flow over time of the critical services.**

I. Introduction

Major disruptions resulting from natural disasters such as hurricanes, floodings, earthquakes or designed malicious attacks can compromise critical infrastructures and hamper services critical for safety. In 2011, the "great east Japan earthquake" and subsequent tsunami hit the north-east of Japan causing enormous loss of life and an overall damage estimated around 309 billions of US dollars [1]. Almost all wired communication networks and emergency municipal radio communication systems were destroyed [2]. Recovery of the network infrastructure required months, during which there was no sufficient support to the most critical services, not to mention normal communications in the devastated areas.

Disaster management requires the restoration of at least the minimum necessary infrastructure to perform safety critical services, with the utmost urgency. These recovery efforts are constrained by the limited availability of human personnel and limited information available during the emergency outbreak.

In this paper we focus on the communication infrastructure, and more specifically, on the operative phases of damage assessment and network recovery. In this context, a complete and detailed damage assessment requires time for extensive monitoring and local inspections. It is therefore fundamental that recovery interventions start as soon as possible even if knowledge of the damage extension is incomplete. Network recovery should follow a progressive process of monitor placement, network probing and repair interventions, which is also necessary for coping with unpredicted variability and surges of the demand [3].

While previous work [4], [5] assumes perfect knowledge of the network status, we consider the more realistic problem of *network recovery under incomplete damage information*, where damage assessment and critical service restoration are performed in a joint manner. For the first time in the literature we formulate the problem of Progressive Damage Assessment and network Recovery (PDAR) which aims at progressively restoring critical services in the shortest possible time, under constraints on the availability of recovery resources. We show that the PDAR problem is NP-hard and may require an unsustainable computation time for large networks.

We propose a polynomial time algorithm called Centrality based Damage Assessment and Restoration (CeDAR) which dynamically schedules repair interventions, local inspections and remote probing of network components, with the objective to restore critical services in the shortest possible time with efficient use of recovery resources. CeDAR restores critical demand flows iteratively by planning repair schedules which are based on the current global view of the network. It schedules the repair of components that can be utilized immediately and with the highest advantage for the largest number of critical services first, maximizing the accumulative service flow during the recovery process.

Since none of the previous approaches is directly applicable to our problem setting, we modified two existing algorithms, originally designed to work under complete knowledge of the failure. In particular we modified the approach proposed by Wang et al. [4] which aims at optimizing the accumulative flow over time under recovery resource constraints. We also modified the approach of Bartolini et al. [5] which instead aims at minimizing restoration costs under quality of service constraints. The modified variants of these approaches work under incomplete and progressively available information so that their actions can be dynamically adjusted depending on the current view of the network status.

Through extensive simulations, we show that CeDAR recovers the network with minimum cost, minimum number of local inspections and highest flow over time, compared to the other approaches in all the experimental scenarios.

We summarize our contributions as follows:

- We model, for the first time, the progressive recovery problem under incomplete knowledge of the disruption and show its NP-hardness.
- We propose a polynomial time heuristic called Centrality based Damage Assessment and Recovery (CeDAR), that solves the problem of progressive network recovery under incomplete knowledge of the network disruption.

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

H. Khamroush and T. La Porta are with the Department of Computer Science and Engineering, Pennsylvania State University, USA. E-mail: {hxk5299,tlp}@cse.psu.edu

This work was supported by the Defense Threat Reduction Agency under the grant HDTRA1-10-1-0085.

- We analyze the properties of CeDAR and prove its correctness, termination, and polynomial time complexity.
- We modified two previous approaches by Wang et al. [4] and Bartolini et al. [5], to make them work under incomplete knowledge of the disruption.
- We evaluate the algorithms under various load settings and disruption scenarios, showing that CeDAR outperforms the other approaches in all the performance metrics.

II. Related work

The growing dependence of our society on communication networks motivates the increasing interest in the problem of network recovery after failures.

Numerous works address the case of sparse failures through the provision of alternative paths, provided either proactively, as in the work of Todimala et. al [6], or reactively, as in the work of Zheng et al. [7], whereas Suchara et. al [8] jointly adress recovery and traffic engineering, to minimize congestion after a failure.

A different line of research considers massive failures, for which no existing alternative path can provide sufficient quality of service. The restoration of communication services, under complete knowledge of the failure extent is addressed by the two works of Wang et al. [4] and Bartolini et al. [5] that are described in Sections VII-A and VII-B, respectively, and considered as baselines for comparisons with our proposal. Ferdousi et al. [9] tackle the problem of progressive datacenter recovery after a large-scale failure.

The problem of network recovery is also studied in the case of interdependent networks, as in the work of Lee et al. in [10]. Finally, other works from Arab et al. [11] and Ho et al. [12] address the problem of recovery in the case of non-communication networks, with specific solutions that are not applicable to our problem setting.

All the mentioned works assume perfect knowledge of the status (working or damaged) of network elements. By contrast, in this paper we consider the more realistic case of incomplete knowledge of the network status.

III. Nomenclature and notation

We model the network as an undirected *supply graph* $G = (V, E)$, where V and E represent nodes and links of the network, respectively. Each link $(i, j) \in E$ has capacity c_{ij}. We also consider a *demand* graph $H = (V_{\mathtt{H}}, E_{\mathtt{H}})$, where $V_{\mathtt{H}} \subseteq V$, and $E_{\mathtt{H}} \subseteq V_{\mathtt{H}} \times V_{\mathtt{H}}$. $E_{\mathtt{H}}$ is the set of *demand pairs*. Each pair (s_h, t_h) has a corresponding demand flow d_h. We consider partial knowledge of the network failures. Therefore the set V is partitioned into the three sets $V_{\mathtt{W}}$, $V_{\mathtt{B}}$, and V_u of working, broken and unknown-status nodes, respectively. The set E is likewise partitioned into the sets $E_{\mathtt{W}}$, $E_{\mathtt{B}}$, and E_u. We define the *working graph* $G_{\mathtt{W}} = (V_{\mathtt{W}}, E'_{\mathtt{W}})$, where $E'_{\mathtt{W}} = E_{\mathtt{W}} \setminus \{(i, j) \in E_{\mathtt{W}} | \{i, j\} \cap (V_{\mathtt{B}} \cup V_u) \neq \emptyset\}$. Namely the working graph is formed by the working nodes of the supply graph, and by the working edges that are not incident to broken nodes or nodes with unknown status. We denote with k^v_i the cost of a repair intervention on node $i \in V_{\mathtt{B}} \cup V_u$. Similarly, we denote with k^e_{ij} the cost of intervention on a link $(i, j) \in E_{\mathtt{B}} \cup E_u$. We consider a time based budget of repair resources, denoted with $B_{\mathtt{repairs}}$, for instance human personnel or vehicles, which determines the amount of repair interventions that can be performed in a same time period.

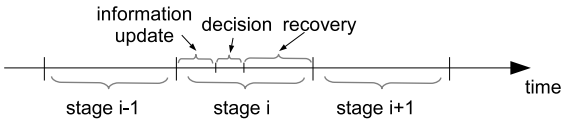


Fig. 1. Stages of PDAR

IV. Problem definition

The problem of progressive damage assessment and network recovery (PDAR) aims at finding a schedule of *repair interventions* to restore a set of critical demand flows as fast as possible, under constrained recovery resources.

PDAR works with partial and progressively available knowledge of the status of the network which is the result of network probing. As long as the repair interventions provided by PDAR are executed, monitoring probes can find new working paths to explore new areas of the network. Moreover, new working nodes can be used as monitors.

We consider subsequent stages of execution as illustrated in Figure 1. Whenever new information is available, the current stage ends and a new stage begins with the *information update* action. The new information is then used to determine a *decision* on the next schedule of repairs. A *recovery phase* follows, with repair interventions and monitor placement, until the next information update becomes available. Notice that, as the information available to PDAR is only partial, a repair intervention may be scheduled also on network elements with unknown status. At a local inspection, such elements may result to be properly working.

A. Assumptions on information update

In order to keep the problem formulation simple, we do not incorporate monitor placement actions in the decision problem, but we assume the following: 1) Software monitors are placed on all the nodes that are selected for a repair intervention (both in the case of broken nodes that have been repaired and in the case of nodes with unknown status that had been scheduled for repair but were found working after a local inspection); 2) Each new monitor node probes the surrounding network until it is able to determine its connected working component. In addition to probing, cable diagnostic devices, such as reflectometers, are used, when available, to determine the status of the adjacent lines of a monitor, if the next hop neighbors are unreachable; 3) The demand endpoints are the first nodes to be repaired, and to host network monitors.

Although for practical purposes it is often desirable to limit the monitoring activity to a given number of hops from the monitor nodes, we assume that a monitor obtains knowledge of its entire connected component in the working graph $G_{\mathtt{W}}$.

Notice that the monitoring activity and the consequent information update trigger the transition to a new stage of PDAR, because it may find a more efficient repair schedule. Nevertheless, if the monitoring activity does not provide any update, or the only unknown elements that are discovered are actually broken or are isolated elements, the current schedule is kept unchanged as PDAR would provide the same solution.

B. The PDAR optimization problem

Due to the constrained repair resource budget, only a limited set of repairs can be executed in parallel. For this reason, PDAR schedules repairs according to the time availability of repair resources. It considers the *stage* as a sequence of successive *steps* in which the maximum number of parallel repairs is bounded due to the budget constraint $B_{\texttt{repair}}$.

Therefore, the PDAR optimization problem works in a sequence of at most N repair steps to be performed at each stage, where N is the maximum number of steps that are necessary to repair all the broken elements. Nevertheless, the sequence of repairs, which will be executed in the recovery phase of the stage, will be terminated if a useful information update is determined. In such a case PDAR will move to the next stage, before reaching the N-th step.

At each step n, there is an update of the composition of the sets of working, broken and unknown network elements. We will add the argument (n) to the notation of these sets when we want to refer to the specific composition they have at stage n. Each stage s starts with $n = 0$, with input consisting of the supply graph G, the demand graph H, and the current stage estimate of the damages, represented by the sets of certainly broken elements $V_{\text{B}}^s(0)$ and $E_{\text{B}}^s(0)$ and by the sets $V_u^s(0)$ and $E_u^s(0)$ of elements whose state is unknown.

The purpose of PDAR is to find a step-based schedule of repairs, which determines the sequence of repair interventions within $V^* = V_{\text{B}}^s(0) \cup V_u^s(0)$ and $E^* = E_{\text{B}}^s(0) \cup E_u^s(0)$ that optimizes the *accumulative demand flow over N steps* $F^*(N)$. This value is defined as follows: $F^*(N) = \sum_{n=1}^{N} f(n)$, where $f(n) = \sum_{h \in E_{\text{H}}} d_h \cdot \alpha_h(n)$, and $\alpha_h(n) \in [0, 1]$ is a variable representing the percentage of the demand flow d_h that is routed at the n-th step.

Let the variables $f_{ij}^h(n) \in \mathbb{R}$, with $f_{ij}^h(n) \geq 0$, represent the fraction of the demand flow h that is routed through the link $(i, j) \in E$, going from vertex i to vertex j, at the completion of the n-th step. Notice that other flows may traverse the same edge in the opposite direction.

Also consider the binary variables $x_{ij}(n)$ and $y_i(n)$. The variable $x_{ij}(n) = 1$ if there is a recovery intervention on edge $(i, j) \in E$ exactly at step n, while $x_{ij}(n) = 0$ otherwise. The variable $y_i(n) = 1$ if node i is repaired at step n, and $y_i(n) = 0$ otherwise. For an edge that has been repaired at the k-th step, it is $x_{ij}(k) = 1$, and $x_{ij}(l) = 0$ for $l \neq k$. We consider working elements as repaired at the 0-th step. For instance, if the node $i \in V_{\text{W}}$ it is $y_i(0) = 1$ and $y_i(n) = 0$ for any other step $n \neq 0$.

The capacity constraint of the problem is expressed by Equation 1(a). If a link (i, j) is still broken at step n, its flow is null, otherwise the flow is bounded by c_{ij}. Notice that if an edge (i, j) is repaired, the corresponding nodes i and j must also be repaired if broken, which implies that $\sum_{k=0}^{n} y_i(k) \geq x_{ij}(n)$, $\forall i, j \in V, \forall n$ as in Equation 1(b).

The flow balance constraint is expressed by Equation 1(c). In this equation $b_i^h = d_h$ if i is the source of the demand flow h, $b_i^h = -d_h$ if i is the destination, and $b_i^h = 0$ otherwise to balance incoming and outgoing flow.

Finally, Equation 1(d) constrains the cost of repairs for each of the N stages, to be limited to the per step budget $B_{\texttt{repair}}$[1]. Equations 1(e-g) denote the domain of the variables of the problem, while Equations 1(h-i) initialize the values of the decision variables for the first stage.

We consider the optimization of the accumulative flow over a horizon of N stages. The PDAR optimization problem is therefore formulated in the variables $x_{ij}(n)$, $y_i(n)$, $\alpha_h(n)$ and $f_{ij}^h(n)$ as follows (we omit the statement $\forall n$ in all the constraints for clarity):

$$
\begin{array}{l}
\text{Max} \sum_{n=1}^{N} \sum_{h \in E_{\text{H}}} d_h \cdot \alpha_h(n) \\
\text{subject to, for all } n = 1, \ldots, N: \\
c_{ij} \cdot \sum_{k=0}^{n} x_{ij}(k) \geq \sum_{h=1}^{|E_{\text{H}}|} (f_{ij}^h(n) + f_{ji}^h(n)), \quad \forall (i,j) \qquad (a) \\
\sum_{k=0}^{n} y_i(k) \geq x_{ij}(n), \quad \forall (i,j) \qquad (b) \\
\sum_{j \in V} f_{ij}^h(n) = \sum_{k \in V} f_{ki}^h(n) + b_i^h \cdot \alpha_h(n), \quad \forall (i,h) \qquad (c) \\
\sum_{(i,j) \in E^*} x_{ij}(n) \cdot k_{ij}^e + \sum_{i \in V^*} y_i(n) \cdot k_i^v <= B_{\texttt{repair}} \qquad (d) \\
f_{ij}^h(n) \geq 0, \quad h \in E_{\text{H}} \qquad (e) \\
y_i(n), x_{ij}(n) \in \{0, 1\}, \quad \forall i \in V, \ (i,j) \in E \qquad (f) \\
\alpha_h(n) \in [0, 1], \quad h \in E_{\text{H}} \qquad (g) \\
y_i(0) = 0, \text{ if } i \in V^*; \quad y_i(0) = 1, \text{ if } i \in V_{\text{W}} \qquad (h) \\
x_{ij}(0) = 0, \text{ if } (i,j) \in E^*; \quad x_{ij}(0) = 1, \text{ if } (i,j) \in E_{\text{W}} \qquad (i)
\end{array}
\tag{1}
$$

As a simpler instance of the problem PDAR has been proven to be NP-hard in [4], the PDAR problem is also NP-hard.

V. The algorithm CeDAR

In this section, we propose a polynomial algorithm, called Centrality based Damage Assessment and Recovery (CeDAR), to solve the PDAR problem introduced in Section IV. We consider a progressive monitoring and network recovery in multiple stages, as in Figure 1. CeDAR aims at maximizing the accumulative flow over time, as follows: (1) prioritizing the repair of network components that can accommodate higher flow, by using a dynamic ranking of broken and unknown elements, based on their centrality with respect to the demand; (2) scheduling the repairs of the same-path elements all at once (or in an interrupted sequence, if not allowed by the time based constraint on repair resources) in order to make the repaired components immediately available for flow routing. For these reasons, CeDAR obtains a high accumulative flow throughout the entire execution period, even when the recovery and monitoring activities are still in progress.

A. Definitions and notation

Each iteration of CeDAR potentially provides an update of the current view of the status of the network. Hence, subsequent iterations correspond to different stages of the PDAR problem, according to the nomenclature introduced in section IV. Notice that some iterations provide long sequences of repair interventions, which may require several steps, within the time constraint on the available repair resources.

At each stage CeDAR performs new repairs and simplifies the problem instance by reducing demand and link capacities according to an operation called *demand pruning*, formalized in Definition V.1. With $G_{\text{W}}(n)$ we denote the composition of the working graph, and with $d_h(n)$, for $(s_h, t_h) \in E_H(n)$ and

[1]This formulation does not consider budget rollover from one repair step to the next in the case of partially depleted budget. This is because we want to use this model to represent limited repair resources, such as vehicles or human personnel.

$c_{kl}(n)$, for $(k,l) \in E$, we denote the demand and capacities updated at the n-th stage. With $d_h(0)$ and $c_{kl}(0)$ we denote the initial values of demand and capacity (before the disruption).

Depending on the needs of the discussion, the same path is equivalently described as an ordered list of links p, or as a subset of nodes and links, and denoted with $\hat{p}$.

Definition V.1 (Pruning of a demand)**.** *Let us consider a demand of $x \leq d_h(n)$ units of flow between the endpoints s_h and t_h, with $(s_h, t_h) \in E_H(n)$, at the current stage n. Let p be a path between s_h and t_h in $G_W(n)$, that is $\hat{p} \subset V_W(n) \cup E_W(n)$, for which $x \leq min_{(i,j)\in\hat{p}} c_{ij}(n)$, for all $(i,j) \in \hat{p}$.* Pruning x units of demand $d_h(n)$ on path p *consists in the decrease of demand $d_h(n)$, so that $d_h(n+1) = d_h(n) - x$, and in the corresponding update of the link capacities of p: $c_{ij}(n+1) = c_{ij}(n) - x$, for all $(i,j) \in \hat{p}$.*

The following notion of routable instance, constitutes the core of the termination condition of the CeDAR. When the current demand is routable on the current working graph, without the need of additional repairs, the algorithm CeDAR terminates.

Definition V.2 (Routable demand)**.** *Given a demand graph $H(n)$ at stage n, and the currently working graph $G_W(n)$, with currently updated capacities $c_{ij}(n)$, for any $(i,j) \in E_W(n)$, we say that $H(n)$ is* routable *on $G_W(n)$ if the capacity constraints and flow balance equations of the related flow routing problem are satisfied, that is:*

$$\begin{cases} \sum_{h\in E_{H(n)}} (f_{ij}^h(n) + f_{ji}^h(n)) \leq c_{ij}(n) \\ \sum_{j\in V_W(n)} f_{ij}^h(n) = \sum_{k\in V_W(n)} f_{ki}^h(n) + b_i^h(n) \\ f_{ij}^h(n) \geq 0, \ \forall i \in V_W(n), (i,j) \in E_W(n), h \in E_H(n) \end{cases} \tag{2}$$

Definition V.3 (Residual capacity graph)**.** *We denote with $G^{TOT}(n)$ the supply graph (i.e. containing all broken, working and unknown components), with residual capacities, considering all the pruning actions performed until stage $n-1$. Such a graph is shortly called the* residual capacity graph*. Notice that $G^{TOT}(n)$ can be obtained from $G_W(n)$ repairing all broken nodes and links.*

The following notion of feasible pruning establishes the necessary conditions for the feasibility of a pruning action. Informally, if a pruning action reduces the capacity of the current graph to the point that the current demand is no longer routable, even with complete repairs, then it would determine a non-feasible instance of the problem and therefore should be prohibited.

Definition V.4 (Feasible pruning)**.** *Given a demand graph $H(n)$ at stage n, and the currently working graph $G_W(n)$ with updated capacities $c_{ij}(n)$, for $(i,j) \in E_W(n)$, we say that the* pruning of x units of demand $d_h(n)$ on path p is feasible*, if after the pruning of x on p, $H(n+1)$ is routable on the residual capacity graph $G^{TOT}(n+1)$.*

Definition V.5 (Infeasible set)**.** *Let $\mathcal{P}$ be a set of paths in the residual capacity graph $G^{TOT}(n)$. $\mathcal{P}$ is an infeasible set for $H(n)$ if for all paths $p \in \mathcal{P}$, and for all the demands $d_h(n)$ in $H(n)$, there is no positive value $\epsilon > 0$ such that pruning of ϵ units of $d_h(n)$ is feasible in p.*

An example of infeasible set is shown in Figure 2. In this example, there are two pairs of demand $d_{xy} = \{x, y\}$ and $d_{st} = \{s, t\}$ with a demand of 1 and 0.5 unit of flow respectively, represented with green lines. The black and solid lines represent the working links and the red and dashed lines represents the broken links. The labels on each edge in the graph represent the residual capacity.

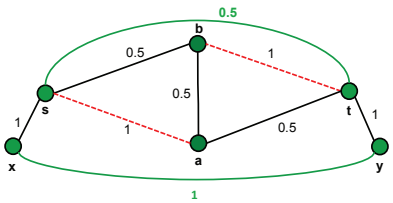


Fig. 2. Infeasible set of paths for demand flows

The entire demand is routable on $G^{\text{TOT}}(n)$, i.e., on the graph of Figure 2, after the repair of the links (s,a) and (b,t).

However, as both the deman pairs have a working path, it may seem intuitive to use it for at least one of them. Nevertheless the current working paths $p_{xy}^w =< x, s, b, a, t, y >$ and $p_{st}^w =< s, b, a, t >$ form an infeasible set. In fact, the pruning of a quantity $\epsilon > 0$ of any of the two demands on its related path, precludes the routability of the remaining demand on $G^{\text{TOT}}(n+1)$, compromising the solution of the problem.

The following definition generalizes a notion of dynamic path length introduced in [5].

Definition V.6 (Cost based path length)**.** *Let p be a path in G, that is $\hat{p} \subset V \cup E$. Let $V_{B|u}^p(n)$ and $E_{B|u}^p(n)$ be the sets of nodes and links traversed by p which at the n-th stage are still broken or unknown. We define the* cost based path length *of p, at the current stage n as follows:*

$$l^{(n)}(p) \triangleq \sum_{(i,j)\in\hat{p}\setminus E_{B|u}^p(n)} \frac{a}{c_{ij}(n)} + \sum_{(i,j)\in E_{B|u}^p(n)} b \cdot \frac{k_{ij}^e}{c_{ij}(n)} + \sum_{i\in V_{B|u}^p(n)} c \cdot k_i^v.$$

For simplicity we consider unitary values of the constants a, b and c, and uniform costs of repair for broken elements $k_{ij}^e = k^e$, and $k_i^v = k^v$ for $(i,j) \in E_{B|u}^p(n)$ and $i \in V_{B|u}^p(n)$, respectively, with $k^e, k^v \gg 1$, for all $(i,j) \in E$ and $i \in V$.

With Definition V.6, the length of a path depends on the number of broken elements, hence varies from stage to stage. Thanks to this dynamic notion of path length, a shortest path selection tends to prioritize paths with fewer broken elements, and links with higher capacities.

We now recall the notion of demand based centrality, previously introduced in [5]. This notion is an extension of the classic definition of betweenness centrality to consider the problem of flow routing.

Definition V.7 (Demand based centrality [5])**.** *The* demand based centrality *$c_d(v)$ of a node $v \in V$ is defined as:*

$$c_d(v) \triangleq \sum_{(ij)\in E_H} \left(\frac{\sum_{p\in\mathscr{P}_{ij}^*|_v} c(p)}{\sum_{p\in\mathscr{P}_{ij}^*} c(p)} \cdot d_{ij} \right) \tag{3}$$

where $\mathscr{P}^*(i,j)$ is the set of the first shortest paths necessary to route the demand (i,j) when considered independently of the other demands, $\mathscr{P}_{ij}^*|_v$ is the set of the paths in $\mathscr{P}^*(i,j)$ traversing v, $c(p)$ is the capacity of path $p \in \mathscr{P}^*(i,j)$, and d_{ij} is the demand flow of the pair $(i,j) \in E_H$.

Notice that, when used by CeDAR the centrality of a node is calculated at each stage to determine how likely the routing of

the demand would benefit from the repair of the node. Hence we calculate the value of $c_d(v)$ by considering the instance of the problem at the current stage n. To this purpose, we consider the current demand graph $H^{(n)}$, while the set of paths $\mathscr{P}^*(i,j)$ is calculated in $G^{\text{TOT}(n)}$, and the length of the paths takes account of the current composition of the sets of broken, unknown, and working elements $V_{\text{B}}(n)$, $E_{\text{B}}(n)$, $V_{\text{u}}(n)$, $E_{\text{u}}(n)$ and $V_{\mathbb{W}}(n)$ and $E_{\mathbb{W}}(n)$.

B. CeDAR in details

In Algorithm 1 we show the details of CeDAR.

We assume that the algorithm has no initial knowledge of the disruption and accumulates information iteratively, through network monitoring.

Initially, in **lines 4-6**, CeDAR repairs the demand endpoints if necessary, and places a software monitor in all of them, to determine their connected working component. CeDAR builds its current view of the working graph $G_{\mathbb{W}}(0)$ with all the nodes and links that were found to be working, and with link capacities as in the original supply graph (before the disruption). If $G_{\mathbb{W}}(0)$ is not sufficient to route all the existing demand flows, CeDAR proceeds with a progressive repair and monitoring of the network, as described in **lines 7-23**.

At stage n of this progressive recovery, in **line 8**, CeDAR computes the set $\mathcal{P}$ that contains, for each demand $d_i \in H(n)$ the corresponding shortest path p_i on $G^{\text{TOT}}(n)$, according to the distance metric given in Definition V.6. Nevertheless $\mathcal{P}$ may constitute an infeasible set for the current demand $H(n)$, according to Definition V.5, tested in **line 9**. In such a case, none of the paths in $\mathcal{P}$ can be used for routing and CeDAR, in **line 10**, resorts to the Equations (2) calculated in $G^{\text{TOT}}(n)$, to determine a set of feasible paths $\mathcal{P}_f$.

As $\mathcal{P}_f$ may contain more than one path for each demand pair, in **line 11**, CeDAR builds the new set $\mathcal{P}$ by choosing, for each demand d_i, the shortest path in $\mathcal{P}_f$.

CeDAR only schedules path repairs when the status of all the elements of the path is known. This is meant to *keep unnecessary local interventions at a minimum*. Therefore, in **line 12**, CeDAR looks for the paths $p_i \in \mathcal{P}$, such that $\hat{p_i} \cap (V_u \cup E_u) = \emptyset$ and, with these, it builds the set of paths with known status $\mathcal{P}^k$.

If there is more than one path in $\mathcal{P}^k$ (**line 13**), then in **line 14** CeDAR chooses the path p_i such that $p_i = \arg\max_{p_i \in \mathcal{P}} min_{(k,l)\in p_i} c_{kl}$, namely, the path of maximum capacity, where the capacity of a path is defined as the capacity of the link with minimum capacity. Further ties are addressed by choosing the path of shortest length (not detailed in the pseudocode). CeDAR then schedules the repair of the entire set of broken elements in p_i, which is $\hat{p_i} \cap (V_{\text{B}}(n) \cup E_{\text{B}}(n))$ in **line 15**, and then the pruning, in **line 16**, of the maximum feasible quantity x of $d_i(n)$, on p_i. In **line 17** CeDAR updates the graphs $G_{\mathbb{W}}(n+1), G^{\text{TOT}}(n+1)$ and $H(n+1)$, to keep track of the scheduled repairs and of the updates in the demands and capacities due to the occurred pruning actions.

If all the selected paths of $\mathcal{P}$ contain at least an unknown element (**line 13**), which implies that $\mathcal{P}^k = \emptyset$, in **line 18** CeDAR selects a new node v_{BC} in which to place a new monitor. To optimize the chance to obtain new information on the area of the network that is of interest for routing the demand flows, CeDAR selects the node v_{BC} in the set $V_m(n) \triangleq \{v \in V | v \in V_u(n) \vee \exists w \in V, \text{ s.t. } (v,w) \in E_u(n)\}$ of nodes that are either unknown or have an incident unknown link. Among the nodes of $V_m(n)$, it selects the one with highest demand based centrality: $v_{\text{BC}} = \arg\max_{v \in V_m(n)} c(v)$, according to Definition V.7.

If at the time of the local intervention, the node v_{BC} is discovered to be broken, it is scheduled for repair in **line 20**, then CeDAR places a monitor in v_{BC}, in **line 21**.

The new repairs and the monitor activity from v_{BC} require an update of the graphs $G_{\mathbb{W}}(n+1)$ and $G^{\text{TOT}}(n+1)$ and the transition to a new stage.

The algorithm terminates with **line 7** as soon as CeDAR determines that the current demand $H(n)$ is routable over the known working graph $G_{\mathbb{W}}(n)$.

Algorithm 1: CeDAR

Input: Supply graph G, demand graph H, broken sets $V_{\text{B}}, E_{\text{B}}$, unknown sets V_u, E_u
Output: Schedule of repairs R

1 Initialize $V_{\text{B}}(0), E_{\text{B}}(0), V_u(0), E_u(0), H(0), R(0)$
2 Build current graphs $G_{\mathbb{W}}(0)$ and $G^{\text{TOT}}(0)$
3 $n \leftarrow 0$
4 **for** $x \in V_H$ **do**
5 If x if broken, append x to $R(n)$ and repair it
6 Monitor from x
7 **while** $H(n)$ *is not routable on* $G_{\mathbb{W}}(n)$ **do**
8 Build the set $\mathcal{P}$ of shortest paths p_i in $G^{\text{TOT}}(n)$, $\forall d_i(n) > 0$
9 **if** $\mathcal{P}$ *is an infeasible set for* $H(n)$ **then**
10 Solve Equations (2) in $G^{\text{TOT}}(n)$ to obtain feasible paths $\mathcal{P}_f$
11 Build $\mathcal{P}$ with the shortest path $p_i \in \mathcal{P}_f$, $\forall d_i \in H(n)$
12 $\mathcal{P}^k = \{p_i | p_i \in \mathcal{P} \text{ and } \hat{p_i} \cap (V_u \cup E_u) = \emptyset\}$
13 **if** $\mathcal{P}^k \neq \emptyset$ **then**
14 Choose $p_i = \arg\max_{p_i \in \mathcal{P}^k} min_{(k,l)\in p_i} c_{kl}(n)$
15 Append elements of $\hat{p_i}$ to $R(n)$ and repair them
16 Prune the max feasible x of $d_i(n)$ over p_i in $G(n)$
17 Build the new sets $G_{\mathbb{W}}(n+1), G^{\text{TOT}}(n+1)$ and $H(n+1)$
18 **else**
19 Choose $v_{\text{BC}} = \arg\max_{v \in V_m(n)} c(v)$
20 If v_{BC} is found broken, append v_{BC} to $R(n)$ and repair it
21 Deploy a monitor in v_{BC}
22 Build the new graphs $G_{\mathbb{W}}(n+1)$ and $G^{\text{TOT}}(n+1)$
23 $n \leftarrow n+1$

VI. Properties of CeDAR

In this section we show the properties of the algorithm. In particular, we focus on the termination, correctness and time complexity of CeDAR.

Theorem VI.1 (Termination and correctness of CeDAR)**.** *Let us consider a demand graph $H = (V_H, E_H)$ and a supply graph $G = (V, E)$, which is partially disrupted, such that V_B, E_B are the sets of broken nodes and links, and V_u, E_u are nodes and links of unknown status, and $V_{\mathbb{W}}$, $E_{\mathbb{W}}$ are the working elements. In a finite number of stages N_{CeDAR}, CeDAR produces a repair schedule R such that the demand H is routable on the* repaired graph $G^R = (V^R, E^R)$*, where $V^R = V_{\mathbb{W}} \cup (R \cap V)$ and $E^R = E_{\mathbb{W}} \cup (R \cap E)$.*

Proof. We first prove that CeDAR terminates in a finite number of stages (termination), then we prove that the demand is routable on the repaired graph (correctness).

Termination. At each stage n, CeDAR selects a set of paths $\mathcal{P}$. If there is at least a path $p \in \mathcal{P}$ such that the status of all the elements of $\hat{p}$ is known, the algorithm enters **lines 14-17**. In this case CeDAR prunes the maximum portion x of a demand d_i on the path p_i, preserving the feasibility of the instance. This requires the solution of an optimization problem with a new variable x. The set of constraints will be the same as in Equations 2, on the graph $G^{\text{TOT}}(n)$, with the additional equality constraints requiring that the demand d_i be routed for a quantity equal to x on the edges of $\hat{p}$ and for the remaining quantity $d_i - x$ in any other edges, possibly including those of $\hat{p}$. Notice that since the only inequality constraints of this optimization problem are those related to link capacities, every time such optimization is executed, there is a capacity constraint which acts as a *binding constraint* [13]. Given a demand, new pruning decisions will create new binding constraints while previous binding constraints will remain binding. As the number of capacity constraints is equal to the number of links in $G^{\text{TOT}}(n)$ it follows that the number of pruning operations for each demand is bounded by $|E^{\text{TOT}}(n)|$.

Let us consider instead the case in which none of the paths in $\mathcal{P}$ is completely known, and for each path $p \in \mathcal{P}$ there is at least one unknown status element, so $\forall p \in \mathcal{P}$, $\hat{p} \cap (V_u \cup E_u) \neq \emptyset$. In such a case, the algorithm actions are provided by **lines 18-22**. Every time this happens a new node v_{BC} is selected from $V_m(n)$ which is the set of nodes that are either unknown or have adjacent unknown links. By placing a monitor in v_{BC}, according to the assumptions detailed in Section IV-A, we assess the status of (at least) v_{BC} and of all its adjacent links, so the number of elements of $V_m(n)$ gradually decreases at each stage. Since this number is lower bounded by 0, the number of monitoring actions is limited by the initial size of V_m.

Correctness. At each stage, CeDAR may either prune a demand, or it may place a monitor and explore its connected component. The first case (**lines 14-17**) CeDAR gradually reduces the total demand preserving the feasibility of the instance, by means of repair and pruning actions, but it requires knowledge of the status of entire paths. In the second case (**lines 18-22**) CeDAR gradually decreases the size of the unknown sets $V_u(n)$ and $E_u(n)$, so it progressively enables more actions of the first kind. Therefore, at each stage new portions of the network are discovered, or a non-infinitesimal demand portion is pruned preserving the feasibility of the problem. As the instance of the problem is feasible by assumption, CeDAR will eventually prune enough demands and repair enough network elements to meet the routability of the demand on the currently repaired graph $G^R = (V^R, E^R)$, where $V^R = V_{\text{W}}(n) \cup (R(n) \cap V)$ and $E^R = E_{\text{W}}(n) \cup (R(n) \cap E)$. □

Theorem VI.2 (Time Complexity of CeDAR)**.** *CeDAR has polynomial time complexity.*

Sketch of the proof. As we discussed in the proof of Theorem VI.1, the number of stages is finite and polynomial, in particular $O(|E_{\text{H}}| \times |E|)$. The proof follows from the observation that the individual activities performed at each stage also have polynomial complexity. □

VII. Comparisons with other approaches

To the best of our knowledge there is no previous work in the literature that addresses the problem of recovery in the case of incomplete knowledge of the failure extent. In this section we introduce two previous approaches. As both of them assume perfect knowledge it would be unfair to compare them to CeDAR in a setting with incomplete information, for which CeDAR is specifically designed. For this reason, we modify these approaches to make them able to determine a progressive recovery schedule, where network monitoring is performed in parallel to repairs, and the recovery plan can be progressively adjusted.

A. Shadow Price Progressive Recovery (ShP)

The work of Wang et al. [4] introduces a progressive recovery approach, which we hereby call the *Shadow Price* (ShP) approach. ShP assumes complete knowledge of the failure which can only affect links and not nodes, and considers limited resource availability to perform simultaneous repairs in a massively disrupted network. The purpose of ShP is to schedule the repairs of the broken network components so as to optimize the weighted sum over time of the flow of every demand pairs. The ShP approach considers the progressive recovery problem as an MILP problem. By recognizing the NP-hardness of the approach, the authors suggest to use an LP relaxation of the problem and suggest to schedule link repairs according to a decreasing order of the shadow prices of the link capacity constraints.

To make the comparison with CeDAR more fair, we modified ShP as follows. First, as ShP cannot work with broken nodes, we let it assume that all nodes are working, and whenever it selects an edge for repair, its endpoint nodes are also repaired if broken, and a monitor is placed on one of them. Second, we consider a progressive execution of ShP, in which ShP is executed iteratively as a single stage process of repair, and a monitoring activity is performed from the newly repaired nodes at each iteration. Finally, we observe that since ShP aims at maximizing flow, and not at meeting specific flow requirements, it may find solutions in which a large flow of one demand compensates for an insufficient flow of another. We modified the LP problem used by ShP, to include an upper bound on each demand flow equal to its requirement, and stop the algorithm execution as soon as all demand requirements are satisfied.

With these three modifications we allow ShP to work also under incomplete knowledge of the failed area and make it more appropriate to meet specific demand requirements. Notice that ShP requires that broken edges have a small residual capacity, to avoid scenarios where all shadow prices are null. This is not realistic, as broken links have null capacity, but is a requirement for the algorithm to work. The values of these residual capacities influence the schedule of repairs. As a consequence ShP does not perform the repair of the components of a same path in an interrupted sequence, which is critical to have high accumulative flow. In the experiments of Section VIII we set the residual capacities of broken links to random values as suggested by the authors [4].

B. Progressive ISP (P-ISP)

Bartolini et al. [5], propose a polynomial heuristic called Iterative Split and Prune (ISP), to solve the problem of minimizing the cost of repair, while restoring critical demand flows. ISP works only with perfect knowledge of the status of nodes and links.

The original version of ISP works by iteratively selecting the next node to repair, called *best candidate*, according to a centrality ranking on the basis of the notion of centrality given in Definition V.7. After the repair of this node, ISP selects a demand to *split*, thus creating two smaller demands with the best candidate as a new end-point for both. The algorithm also provides a *pruning* operation, which is similar to the one performed by CeDAR, but works under different enabling conditions based on structural properties of the demand and supply graph.

To make ISP able to work in the case of partial knowledge of the disruption, we designed a progressive variant, hereby called Progressive ISP (P-ISP) as follows. First, we assume that network elements of unknown status are broken, and let P-ISP consider them as high cost repair elements. Second, we consider a progressive execution in stages, where at every stage P-ISP executes both repair interventions, monitor deployment, and network probing, according to a stage model similar to the one of Figure 1.

Despite the modifications, P-ISP could still make wrong decisions due to the assumption that components with unknown status are broken. This may cause P-ISP to split a demand on a best candidate node which may result an inefficient choice when more knowledge becomes available. Moreover, the split action determines an irreversible routing decision that may compromise the entire solution of the problem. In addition to this, IPS performs repairs one at a time, potentially scheduling successive repairs in distant and unrelated portions of the network, resulting in a low accumulative flow over time.

VIII. Experiments

In this section we study the behavior of the discussed approaches by means of simulations. We consider a real network topology, taken from the CAIDA (Center for Applied Internet Data Analysis) dataset [14]. This dataset includes real topologies describing the connections between backbone/gateway routers of several autonomous sytems. We used the topology AS28717, of which we extracted the giant connected component with 825 nodes and 1018 edges, where we set the edge capacities randomly in a range between 20 and 50 units.

In the following experiments we considered three different scenarios in which we varied the number of demands, the amount of flow for each demand, and the extent of the disruption, randomizing the results for a minimum of 20 runs for each experiment.

In all the experiments, with the only exception being the optimal (OPT) solution, we assume that the initial knowledge of the network state is only partial, and determined by monitoring the network from the demand endpoints. The OPT solution instead is obtained by using complete knowledge of the disruption and solving the NP-hard optimization problem PDAR of Section IV-B. Therefore, we underline that OPT is an ideal solution and is considered only as a baseline for comparisons, to evidence the margin of improvement that any algorithm can provide with respect to existing solutions. For this reason we show the comparisons with OPT only in the first scenario.

A. Scenario A: Varying demand intensity

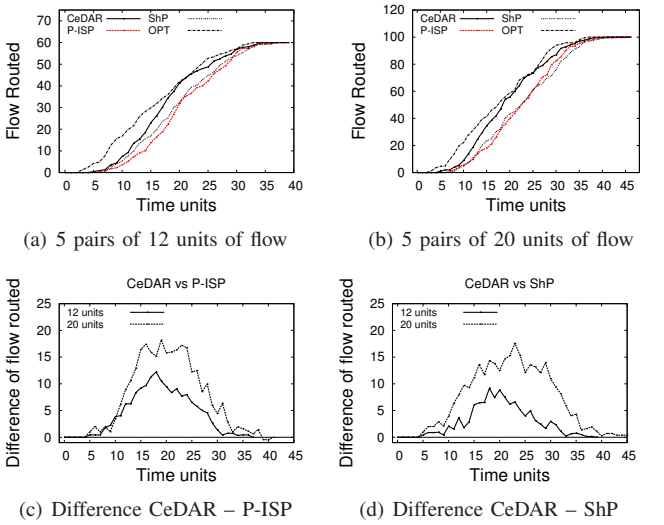
In this scenario we increase the load on the network by varying the amount of flow of 5 uniform demand pairs, with randomly selected endpoints. We generate the network disruption so as to form multiple disconnected portions. To this purpose we generate a geographic distribution of the probability of failure, in the form of a composition of two bi-variate Gaussian distributions, representing two epicenters of maximum disruption probability. The disruption probability gradually decreases with the distance from the epicenters. The extent of the disruption is such that 60% of the network components are broken.

In Figure 3 we show the effects of the progressive recovery actions of the three algorithms CeDAR, P-ISP and ShP and of the optimal solution OPT. The figure shows the trend with time of the maximum amount of critical flow that can be routed on the currently repaired supply network. In the figure, the number of repairs grows in proportion with time as we consider that all the algorithms repair one network element at each time step, to mimic a scenario with limited resources.

We consider two different load settings: a case with moderate flow in Figure 3(a) corresponding to 12 flow units for each of the 5 demand pairs, and a case with high flow in Figure 3(b), corresponding to 5 demand pairs of 20 flow units each. The figure shows that CeDAR outperforms P-ISP and ShP by routing more flow at each time step, with peaks of about 18 flow units of difference, corresponding to the 30% of the total demand in the case of moderate flow, and to the 18% in the case of high flow. Compared with OPT, CeDAR shows a good approximation in the initial phase, with a difference between the two within the 15% of the total demand, that gradually becomes even lower, until the recovery process is halfway, when the difference between CeDAR and OPT becomes negligible.

Figures 3(c) and 3(d) emphasize the difference between CeDAR and the other two algorithms by showing how much more flow CeDAR routes in both the considered load settings. For instance, in the case of high load, corresponding to the dashed lines of Figures 3(c) and 3(d), after about 20 rounds of repairs CeDAR routes an amount of flow that is 20 units higher than P-ISP (see Figure 3(c)), and 15 units higher than ShP (see Figure 3(d)). In the entire execution period, CeDAR routes more flow than ShP, despite the fact that ShP targets cumulative flow as main objective function.

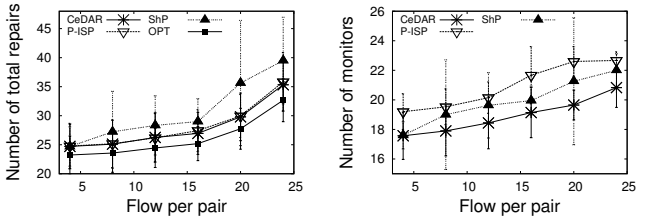
Figure 4 considers an experiment where we increased the amount of flow of each of the 5 demand pairs from 4 to 24 flow units. Figure 4(a) shows the number of repairs needed to route the entire flow demands. With respect to the number of repairs, CeDAR outperforms ShP and performs the same as P-ISP which instead is specifically meant to optimize repair cost. Notice that the number of repairs performed by CeDAR



(a) 5 pairs of 12 units of flow (b) 5 pairs of 20 units of flow

(c) Difference CeDAR – P-ISP (d) Difference CeDAR – ShP

Fig. 3. Scenario A. 5 demand pairs with varying demand intensity. Flow routed, 12 (a) and 20 (b) flow units per pair. Flow difference: CeDAR vs. P-ISP (c), CeDAR vs. ShP (d)



(a) (b)

Fig. 4. Scenario A. 5 demand pairs, with varying demand intensity: total repairs (a), monitors (b)

and P-ISP is close to the optimal OPT, which assumes full knowledge of the status of the network nodes and links. By contrast, ShP needs to repair more network elements than the other algorithms to route the same amount of flow. This is due to the fact that ShP aims at optimizing cumulative flow at each iteration, so it may decide to sacrifice cost, by repairing more elements than strictly necessary for the purpose of satisfying the demand requirements.

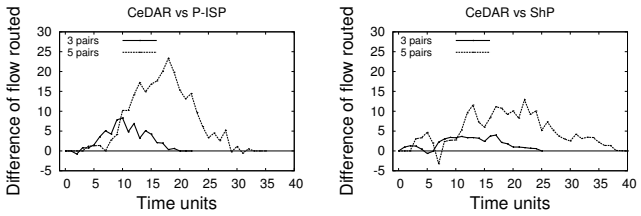
Figure 4(b) shows that CeDAR deploys a lower number of monitors than ShP and P-ISP. This means that CeDAR is able to perform the necessary monitoring activity with a lower number of monitors, thanks to more focused monitor deployment decisions that aim at obtaining information on portions of the network that are more relevant to the demand requirements.

B. Scenario B: Varying number of demand pairs

In this set of experiments we considered the effect of the demand load by varying the number of critical demand flows in the range from 1 to 6. We consider critical demands of 22 flow units. Also in this scenario, the disruption is generated according to the composition of two bi-variate Gaussian distributions so that 40% of the network components are broken.

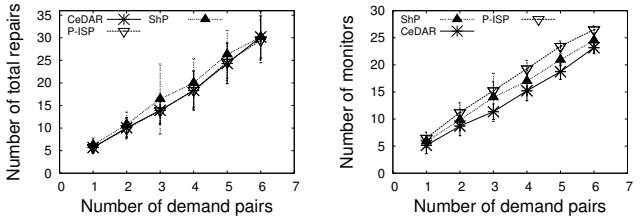
For space limitations we no longer show the trend of the flow routed with time. Figure 5 shows instead the difference between the flow routed by CeDAR with respect to P-ISP (Figure 5(a)) and ShP (Figure 5(b)). For instance, with 5 pairs of demand, after about 18 rounds of repairs, CeDAR routes about 25 more units of flow than P-ISP, corresponding to the 23% of the total demand, and about 11 more units than ShP, which is the 10% of the total demand.

With the last two figures, by varying the number of demand pairs from 1 to 6, we show that CeDAR routes the entire demand with a similar number of repairs as P-ISP, lower than ShP, as shown in Figure 6(a) and with a lower number of monitors, as shown in Figure 6(b). We conclude that in this scenario, with a number of repairs close to those of P-ISP and lower than ShP, CeDAR achieves a higher cumulative flow than both the other algorithms.



(a) Difference CeDAR – P-ISP (b) Difference CeDAR – ShP

Fig. 5. Scen. B. Flow difference: CeDAR vs. P-ISP (a), CeDAR vs. ShP (b)



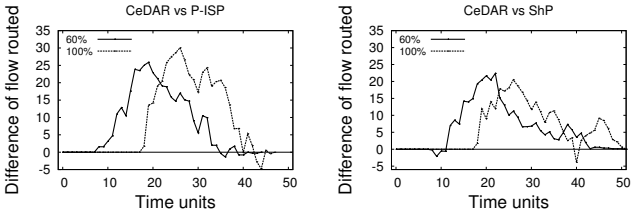
(a) (b)

Fig. 6. Scenario B. Varying demand pairs: repairs (a), monitors (b)

C. Scenario C: Varying disruption extent

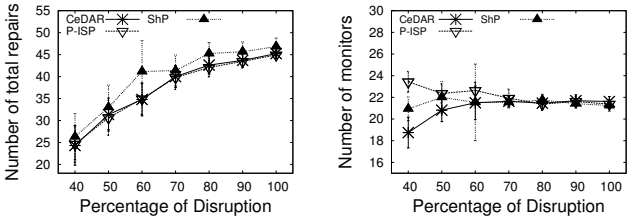
In this last scenario, we investigate the behavior of the three algorithms by varying the extent of the disruption. Similar to the previous experiments we consider two epicenters of bi-variate Gaussian failure distribution. It must be noted that when the network disruption is sparse, the monitoring activity is particularly beneficial. Indeed by placing monitors on nodes of the unknown area of the network, it is likely that these can discover large connected working components of the network, and increase the speed of the recovery process. By contrast, if the extent of the disruption is very large, the monitoring activity is less effective as it is more likely that nodes in the unknown area have broken adjacent links and therefore cannot send monitoring probes to perform the exploration of the surrounding network.

We first consider two different settings, with moderate and complete disruption. The extent is such that 60% of the network elements are broken in the first case, and 100% in the second. We consider 5 demand pairs with a demand of 22 flow units each. The difference between the amount of flow routed by CeDAR and the other algorithms is particularly remarkable in both the considered scenarios. Figure 7(a) shows the difference between the total flow routed by CeDAR and P-ISP. In the case of complete disruption (dashed line), after 28 time units, the flow routed by CeDAR is 30 units higher than



(a) Difference CeDAR – P-ISP (b) Difference CeDAR – ShP

Fig. 7. Scen. C. Flow difference: CeDAR vs. P-ISP (a), CeDAR vs. ShP (b)



(a) (b)

Fig. 8. Scen. C. Varying disruption: repairs (a), monitors (b)

with P-ISP, corresponding to about 27% of the total demand. Similar to these results, Figure 7(b) shows that in the case of complete disruption, after about 20 time units, CeDAR routes 23 more units of flow than P-ISP, corresponding to about the 21% of the total demand.

The figure shows analogous results for the case of moderate disruption (solid line), where the accumulative flow of CeDAR is even higher as it allows to route flow sooner, i.e., after about 7 time units, compared to the case of complete disruption (dashed line), in which it requires 18 time units before we start seeing a positive flow routed. It is evident that in the case of moderate disruption there is more room for prioritizing the repair of the network elements that can ensure a higher value of the cumulative flow over time, while in the case of large disruption it is more likely that the first recovery interventions will not be sufficient to accommodate any demand flow nor to create enough working paths for monitoring.

The study of this scenario confirms the results discussed for Scenario A and B. The higher cumulative flow routed by CeDAR is obtained through a better scheduling of repairs and monitor placement.

A more detailed study, conducted by varying the disruption from 40% to 100% evidences that CeDAR performs a number of repairs close to P-ISP and lower than ShP, as shown in Figure 8(a) while Figure 8(b) shows that CeDAR also requires a lower number of monitors. Notice that in the case of large disruption the number of monitors coincides with the number of repaired nodes as most of the nodes selected to host monitors in the unknown area, are found to be broken.

D. Discussion on execution time

Due to space limitation we do not include graphs of the comparisons among the three approaches in terms of execution time. We performed experiments in which we varied the problem size, both in terms of number of demands and amount of flow for each demand, and the disruption extent. We also tested the three algorithms on different topologies. We noticed that average node degree affects the computation time of the algorithms more than other aspects. For example, on a relatively small artificial network with 100 nodes, average node degree of about 30, low demands and large capacities (only connectivity requirements), and complete knowledge, the optimal solution of the PDAR problem requires more than 5 hours of execution time, while ShP takes about 30 minutes, ISP 15 minutes and CeDAR less than 5 minutes. For a degree of 10, the computation time drops significantly to about one hour for the optimal, and to order of seconds for the heuristics. The difference between the optimal and the heuristics becomes much more evident when the algorithms need to be executed several times, for multiple stages, to determine adjustments of the solution in the case of partial knowledge, with monitoring, and for larger networks.

IX. Conclusions

In this work we studied for the first time, the problem of progressive recovery of a communication network after large scale failure under incomplete knowledge of the damage extent. We model the problem of Progressive Damage Assessment and network Recovery (PDAR), which is shown to be NP-Hard. We propose CeDAR, an efficient heuristic to solve PDAR, that performs joint repair and monitor interventions, to progressively restore critical services. We compared CeDAR with previous approaches modified to deal with incomplete knowledge. Experimental results on real topologies show that CeDAR outperforms the previous approaches with a significantly higher accumulative flow over time and comparable repair cost.

References

[1] E. Neumayer, T. Plumper, and F. Barthel, "The political economy of natural disaster damage," *Elsevier Global Environmental Change*, January 2014.

[2] Y. Nemoto and K. Hamaguchi, "Resilient ict research based on lessons learned from the great east japan earthquake," *IEEE Communications Magazine*, no. 3, 2014.

[3] S. Bailey, "Disaster preparedness and resiliency," *Guide to Reliable Internet Services and Applications, Springer*, 2010.

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

[5] N. Bartolini, S. Ciavarella, T. La Porta, and S. Silvestri, "Network recovery after massive failures," *IEEE DSN*, June 2016.

[6] A. Todimala and B. Ramamurthy, "Approximation algorithms for survivable multicommodity flow problems with applications to network design," *IEEE INFOCOM*, April 2006.

[7] Q. Zheng, G. Cao, T. La Porta, and A. Swami, "Optimal recovery from large-scale failures in ip networks," *IEEE ICDCS*, 2012.

[8] M. Suchara, D. Xu, R. Doverspike, D. Johnson, and J. Rexford, "Network architecture for joint failure recovery and traffic engineering," *ACM SIGMETRICS*, June 2011.

[9] S. Ferdousi, F. Dikbiyik, M. Tornatore, and B. Mukherjee, "Progressive datacenter recovery over optical core networks after a large-scale disaster." *IEEE DRCN*, 2016.

[10] J. E. M. E. E. Lee and W. A. Wallace, "Restoration of services in interdependent infrastructure systems: A network flows approach." *IEEE Trans. on Systems, Man, and Cybernetics, Part C*, vol. 37, no. 6, 2007.

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

[12] H. Ho and A. Sumalee, "Optimal recovery plan after disaster," *Journal of Transportation Engineering*, vol. 140, no. 8, 2014.

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

[14] "The cooperative association for internet data analysis (CAIDA)," Macroscopic Internet Topology Data Kit (ITDK), 2013. [Online]. Available: http://www.caida.org/data/active/internet-topology-data-kit/