On Progressive Network Recovery from Massive Failures under Uncertainty

Diman Zad Tootaghaj, *Student Member, IEEE*, Novella Bartolini, *Senior Member, IEEE*, Hana Khamfroush, *Member, IEEE*, Thomas La Porta, *Fellow Member, IEEE*

***Abstract*—Network recovery after large-scale failures has tremendous cost implications. While numerous approaches have been proposed to restore critical services after large-scale failures, they mostly assume having full knowledge of failure location, which cannot be achieved in real failure scenarios. Making restoration decisions under uncertainty is often further complicated in a large-scale failure. This paper addresses progressive network recovery under the uncertain knowledge of damages. We formulate the problem as a mixed integer linear programming (MILP) and show that it is NP-Hard. We propose an iterative stochastic recovery algorithm (ISR) to recover the network in a progressive manner to satisfy the critical services. At each optimization step, we make a decision to repair a part of the network and gather more information iteratively, until critical services are completely restored. We propose three different approaches: 1) an iterative shortest path algorithm (ISR-SRT), 2) an approximate branch and bound (ISR-BB) and 3) an iterative multi-commodity LP relaxation (ISR-MULT). Further, we compared our approach with the state-of-the-art Centrality based Damage Assessment and Recovery (CeDAR) and iterative split and prune (ISP) algorithms. Our results show that ISR-BB and ISR-MULT outperform the state-of-the-art ISP and CeDAR algorithms while we can configure our choice of trade-off between the execution time, the number of repairs (cost) and the demand loss. We show that our recovery algorithm, on average, can reduce the total number of repairs by a factor of about 3 with respect to ISP, while satisfying all critical demands.**

***Index Terms*—Network Recovery, Massive Disruption, Stochastic Optimization, Uncertainty.**

I. Introduction

LARGE-SCALE failures in communication networks due to natural disasters can severely affect critical communications and threaten the lives of people in that area. In 2005, Hurricane Katrina led to an outage of over 2.5 million lines in the BellSouth (now AT&T) network [2]. In 2017, more than 7 million subscribers to cable or wire-line telecommunication services lost service due to Hurricane Irma [3]. In the absence of a proper communication infrastructure, rescue operation becomes extremely difficult. Progressive and timely network recovery is, therefore, a key to minimizing losses and facilitating rescue missions. Many prior works on failure detection and recovery assume full knowledge of failures and use a deterministic approach for the recovery phase, e.g., [4, 5]. In real-world scenarios, however, the failure pattern might be unknown or only partially known. Therefore, classic recovery approaches may not work, as they should. To this end, we focus on network recovery assuming partial and uncertain knowledge of the failure pattern.

We propose a multi-stage stochastic recovery algorithm, that uses three optimization techniques to repair a part of the network at each iteration To clarify the discussion, we consider different states of network components. Depending on the available knowledge, we consider the network to be partitioned into three areas: 1) a green area where all nodes/edges are known to be working, 2) a red area where the status of nodes/edges is known to be failed, and 3) a gray area where the status of nodes/edges is unknown. We improve the knowledge of the network state by installing monitors on top of the nodes repaired at each iteration. A monitor is a piece of software, which can be installed on a working node to discover the reachable nodes. Monitor nodes provide additional information about the status of the network, which can be used to revise and improve the recovery plan. The **contributions** of this work are the following:

- We tackle for the first time, the problem of network recovery after massive disruption under uncertainty of the exact location of the disrupted nodes/links.
- We formulate the *minimum expected recovery* (MINER) problem as a mixed integer linear programming and show that it is NP-Hard. MINER aims at satisfying the critical demand flows while minimizing the proposed expected recovery cost (*ERC*) function under network capacity constraints.
- We propose a multi-stage iterative stochastic recovery (ISR) algorithm, that is presented in three different versions (depending on the optimization algorithm that is used), namely, *Iterative shortest path* (ISR-SRT), *Iterative Branch and Bound* (ISR-BB), and *iterative multi-commodity LP relaxation* (ISR-MULT) to find a feasible solution and solve the MINER problem.
- In order to provide a fair comparison with our approach, we modified the state-of-the-art *iterative split and prune* (ISP) algorithm [5], presented as *progressive ISP* to work

D. Z. Tootaghaj was with the Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802 USA. She is now with Networking and Mobility group, Hewlett Packard Enterprise Labs, Palo Alto, CA 94304 USA (email: {diman.zad-tootaghaj}@hpe.com). T. La Porta is with the School of Electrical Engineering and Computer Science in the Pennsylvania State University, University Park, PA 16802 USA (email: {tlp}@cse.psu.edu), N. Bartolini is with the Department of Computer Science, Sapienza University of Rome, 00185 Rome, Italy (email: {bartolini}@di.uniroma1.it), H. Khamfroush is with the Department of Computer Science in University of Kentucky, Lexington, KY 40506 USA (email: {khamfroush}@cs.uky.edu). A partial and preliminary version appeared in Proc. IEEE IFIP Networking'17 [1].

under uncertainty, by allowing a progressive approach and adding a discovery phase at each iteration. We show that since ISP does not consider uncertain failures and makes routing decisions at each iteration step, it may lead to incorrect routing decisions due to uncertainty which leads to higher repair cost compared to our algorithms.
- Further, we compare our algorithms with the state-of-the-art *Centrality based Damage Assessment and Recovery* (CeDAR) algorithm [6]. CeDAR works under the incomplete knowledge of failure but aims at maximizing the total satisfied flow during the recovery process, while we aim at minimizing the recovery cost. Further, CeDAR does not use a probabilistic knowledge of failures. Our results show that ISR-BB and ISR-MULT outperform the state-of-the-art ISP and CeDAR algorithms in terms of recovery cost.

We observed that since the algorithms in [5] have been designed for known network failure patterns, poor decisions in the first iterations of the algorithm propagate through the entire execution, while our algorithm can correct previous decisions after each iteration as more information becomes available and therefore, reduces the recovery cost. Different configurations of our algorithm can significantly improve the total number of repairs over other heuristics while performing close to the optimal in terms of recovery cost.

The remainder of this paper is organized as follows. Section II discusses the background and motivation behind this work. In Section III, we explain the minimum expected recovery (*MINER*) problem and show it is NP-Hard. Section IV describes our approach to minimize the expected recovery cost. Section VI shows our evaluation methodology and experimental results and Section VII concludes the paper with a summary.

II. Background and Motivation

A. Background

Large-scale network failure detection and recovery have been studied when full knowledge of the failure pattern is available in the system [7–12]. To the best of our knowledge, network recovery has not been extensively studied under uncertainty.

In the absence of complete knowledge of disrupted network components, prior works propose network tomography techniques to localize node failures from binary states of end-to-end paths or infer the performance degradation of links [13–16]. Our paper differs significantly from the literature work in the area of network tomography, as it uses progressive monitoring with the purpose of providing information necessary to perform the recovery activities. The problem of recovery is not considered in tomography studies which address the problem of localizing sparse degradation and failures, and are not suitable for massive failure scenarios.

In a different line of research the problem of network recovery has been studied in the case of interdependent networks [17–21]. Dependent failures in interdependent networks between a power grid and a communication network have been studied in [20]. Tootaghaj et al. propose a progressive recovery approach in an interdependent network consisting of a power grid and a communication network [17, 18]. They assume a limited amount of resources available at each iteration step and propose a progressive recovery heuristic that restores the power lines while maximizing the total load served during the recovery intervention.

Wang et al. and Ciavarella et al. studied progressive network recovery for large-scale failures. They proposed a progressive recovery approach to maximize the weighted sum of total flow over the entire steps of recovery [4, 6]. While both Wang et al. and Ciavarella et al.'s work and our work aim to design a progressive recovery approach, the objective is different. In [4] and [6], the objective is maximizing the throughput over time, whereas we aim to minimize the total cost of repair under link capacity constraints, which is closer to the work of Bartolini et al. [5, 22]. In addition, both [4] and [5], assume that full knowledge of failure is available in the system while our work and [6] do not make this assumption. We assume the availability of a probabilistic estimate of the failure scenario, while [6] assumes lack of knowledge.

The problem of minimizing the recovery cost to satisfy multiple demand flows under network capacity or quality of service constraint has been proven to be NP-hard and several heuristics have been proposed in the literature to reduce the complexity. Bartolini et al. propose a polynomial-time heuristic, called ISP, to break the problem into smaller subproblems using iterative split and prune [5]. While this approach performs very close to the optimal when full knowledge of network failure is available, its performance has not been investigated under uncertain failure patterns.

We propose a progressive version of ISP in Section V-A and show that the lack of detailed information regarding the status of the network components causes a considerable amount of additional repairs with respect to the ideal case of complete knowledge. Then, we show that by running our multi-stage recovery approach, we can reduce the total number of repairs compared to ISP and avoid unnecessary repairs. Furthermore, we show that single-stage optimization techniques or iterative algorithms, which do not update the initial beliefs, do not perform well for uncertain failures. This is due to the fact that a small mistake at the beginning of the single-stage optimization algorithms propagates through the following steps and no corrective actions can be taken.

We design a novel iterative algorithm to have an approximate solution to the problem of network recovery under uncertainty of the status of network components. Unlike previous algorithms, our approach provides an iterative monitoring activity, which allows more informed decisions and corrective actions as long as more information becomes available.

B. Motivation

In this section, we show the gap between optimal recovery and ISP [5] when we do not have perfect information.

Consider a network in which a large-scale failure has

TABLE I: Number of repaired nodes/edges in optimal and ISP with full information compared to ISP with gray area and uncertain-info, and progressive ISP.

Network Name	# of network elements (nodes/edges)	OPT full-info	ISP full-info	ISP uncertain-info	Progressive ISP
BellCanada	112	28	34.23	79	45.39
Deltacom	274	36.94	43.26	112	55.5
KDL	1649	55.2	63.2	165.65	83.55

occurred. Due to the failures, the state of the entire network is not visible to the network manager. Instead, the network manager knows that some nodes and links have failed, some continue to work, and the fate of others is uncertain (gray), meaning that their state is only known with some probability. If we use an algorithm like ISP to determine the required repairs, it is likely that mistakes will be made due to the uncertain knowledge. ISP determines all the required repairs in a one-shot, meaning that once the algorithm is run, all repairs are determined. Therefore, it does not have an opportunity to learn the state of the uncertain nodes. We performed a set of simulations on three different network topologies to illustrate the gap.

Table I shows the average total number of repairs for the three algorithms on the three topologies for 10 random runs. As is shown, ISP with full information performs relatively close to the optimal in each case, while ISP with uncertain information requires more than three times the number of repairs as optimal in some cases.

We then modified ISP, as described in Section V-A, to run in iterations, where at each iteration a repair is made. Information about network state is then gathered for any new portions of the network now visible due to the repair, and the algorithm is run again with the new information until all demands are met. We call this algorithm progressive ISP. As can be seen in the Table, progressive ISP drastically reduces the repairs compared to ISP with uncertain information. However, the gap between progressive ISP and OPT full-info is still large which motivates us to develop our iterative stochastic recovery (ISR) algorithm that progressively repairs network elements and updates its knowledge of the state of the network.

III. Problem Definition

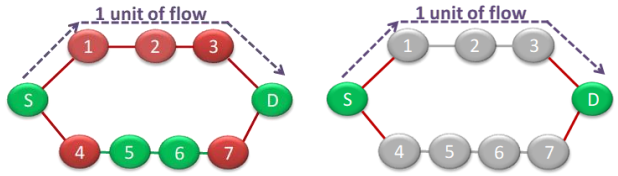
We consider the problem of restoring critical services in a network subject to a large-scale failure, under the uncertain knowledge of the failure extent. We are interested in gradual recovery of the network such that the total cost of repaired nodes/edges during all steps of the recovery is minimized.

A. Network Model

Given an undirected graph $G = (V, E)$ and a set of demand pairs $E_H = \{(s_1, t_1), ..., (s_k, t_k)\}$, where $E_H \subseteq V \times V$ and each demand pair $(s_h, t_h) \in E_H$ has a source s_h, a destination t_h and a positive demand flow d_h, the goal is to minimize the expected recovery cost (ERC) to satisfy the demands while having capacity constraint c_{ij} for every edge in the graph.

TABLE II: Summary of notations.

Notation	Explanation
$G = (V, E)$	undirected graph modeling the communication network. V is the set of nodes and E is the set of links.
E_H	set of (source, destination) demand pairs.
$E_B \subseteq E$	the set of broken edges in the red area.
$V_B \subseteq V$	the set of broken nodes in the red area.
$E_U \subseteq E$	the set of edges in the gray area whose failure pattern is unknown.
$V_U \subseteq V$	the set of nodes in the gray area whose failure patterns is unknown
$E_W \subseteq E$	the set of edges in the green area which are known to be working correctly.
$V_W \subseteq V$	the set of nodes in the green area which are known to be working correctly.
$\zeta_i^v(n)$	the failure probability of node v_i in the network at the n^{th} iteration.
$\zeta_{ij}^e(n)$	the failure probability of edge e_{ij} in the network at the n^{th} iteration.
$k_{ij}^e(\zeta_{ij}^e(n))$	the cost of repairing edge e_{ij} in the network.
$k_i^v(\zeta_i^v(n))$	the cost of repairing vertex v_i in the network.
c_{ij}	the capacity of edge (i, j).
$f_{ij}^h(n)$	the fraction of flow h that will be routed through link (i, j).
η_{max}	the maximum degree of the network.
$b_i^h(n)$	the flow h generated at node i.
δ_{ij}^e	the decision to repair link $(i, j) \in E$.
δ_i^v	the decision to repair node $i \in V$.



(a) Complete Information. (b) Uncertain Information.

Fig. 1: An example of a failure in a network topology under (a) complete information and (b) uncertain information.

Table II shows the notation used in this paper.

The nodes and edges in the graph $G = (V, E)$ belong to three different categories:

1) the sets $E_B \subseteq E$ and $V_B \subseteq V$ are the set of **broken** edges and nodes in the red area which we know for sure have failed,
2) the sets $E_U \subseteq E$ and $V_U \subseteq V$ are the sets of edges and nodes in the gray area whose failure patterns is **unknown**,
3) the sets $E_W \subseteq E$ and $V_W \subseteq V$ are the sets of nodes and edges in the green area which are known to be **working** correctly in the system.

To clarify the discussion, consider the network shown in Figure 2 where all links have a capacity of one. Suppose that there exists one source-destination pair $(S - D)$ that has one unit of demand. Assume there exists a massive failure in the red area of figure 1a. At the beginning of the recovery process, we do not have complete knowledge about the failures and the network is partitioned into a green, red and gray areas as in figure 1b. Assuming the same recovery cost for all nodes and

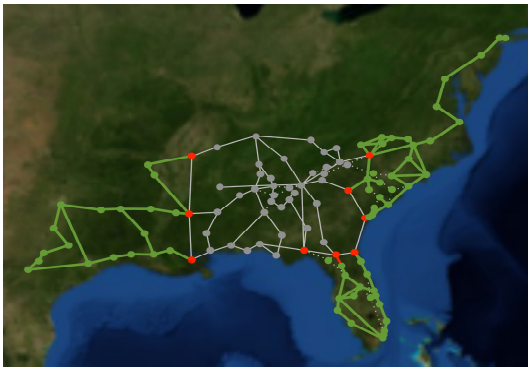


Fig. 2: An example of a failure in a real network topology from the internet topology zoo [24].

edges in the network, it is obvious that repairing nodes $\{4, 7\}$ and edges $\{S-4, 7-D\}$ is the optimal recovery approach with a total recovery cost of 4. However, under uncertain knowledge of failures we might end up repairing nodes $\{1, 2, 3\}$ and edges $\{S-1, 1-2, 2-3, 3-D\}$ with a total recovery cost of 7.

Figure 2, shows a large-scale failure in a real network topology (DeltaCom) taken from the internet topology zoo [23]. At the beginning of the recovery process, we do not have complete knowledge about the failures and the network is partitioned into a green, red and gray area. It is possible that the nodes in the middle of the gray area have not failed and therefore, recovery of a few nodes in the gray area may lead the whole graph of the network to be connected. Figure 3 shows different steps of our recovery approach. At each iteration step, based on the current knowledge of the network state, we repair some of the damaged network elements, perform a monitoring step and gain more information and iterate this procedure until all critical services are restored. At the beginning of the iterative recovery process, we assume that all the working demand endpoints are endowed with software monitors which can discover the status of up to K-hops in the network. Later, as we repair more nodes/edges in the network, we exploit the repaired nodes as monitors to discover the gray area and adjust the initial estimate $\zeta(0)$ about the failure probability distribution. Any technology specific methodology, that can provide incremental information on the status of the network components, can be integrated in the monitoring phase of the proposed algorithm.

B. Recovery Problem

To model our optimization problem as a decision-making process which includes uncertainty, we model the cost of repair as a function of the failure probability for each node/edge in the network. Our estimate of the probability of failure in the location of the considered network elements at the n^{th} iteration is $\zeta(n) = \{\zeta^e_{ij}(n) \quad \forall e_{ij} \in E, \zeta^v_i(n) \quad \forall v_i \in V\}$, where $\zeta^v_i(n)$ and $\zeta^e_{ij}(n)$ are representing our estimate about the failure probability of node v_i and edge e_{ij} in the network at the n^{th} iteration. We use heterogeneous non-uniform cost function in our evaluation, where k^v_i and k^e_{ij} is the cost of repairing

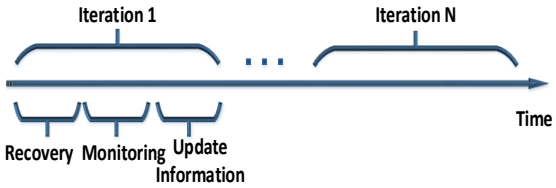


Fig. 3: Different steps of our progressive recovery approach.

each vertex v_i and edge e_{ij} in the network. We note that the cost of repairing each node/edge also depends on its location. For example, it is more difficult to access and repair a node on a mountain or an edge that crosses an ocean. Therefore, our objective function is to minimize the expected recovery cost (ERC) given the information from the monitoring nodes to satisfy the given demand. We assume that at each iteration step we have enough resources to repair one node and its adjacent edges.

The MINER problem to find a feasible solution set at the n^{th} iteration can be formulated as follows:

$$\text{Min} \sum_{(i,j)\in E_U \cup E_B} k^e_{ij} \cdot \zeta^e_{ij}(n)\delta^e_{ij}(n) + \sum_{i \in V_U \cup V_B} k^v_i \cdot \zeta^v_i(n)\delta^v_i(n)$$

$$\text{s.t. } c_{ij}.\delta^e_{ij}(n) \geqslant \sum_{h=1}^{|E_H|} f^h_{ij}(n) + f^h_{ji}(n), \qquad \forall (i,j) \in E \tag{1a}$$

$$\delta^v_i(n) \cdot \eta_{max} \geqslant \sum\nolimits_{(i,j)\in E_B} \delta^e_{ij}(n), \qquad \forall i \in V \tag{1b}$$

$$\sum\nolimits_{j\in V} f^h_{ij}(n) = \sum\nolimits_{k\in V} f^h_{ki}(n) + b^h_i(n), \quad \forall (i,h) \in V \times E_H \tag{1c}$$

$$f^h_{ij}(n) \geqslant 0, \qquad \forall (i,j) \in E, h \in E_H \tag{1d}$$

$$\delta^v_i(n), \delta^e_{i,j}(n) \in \{0,1\}, \quad \forall (i,j) \in E, \quad \forall i \in V \tag{1e}$$

where the binary variables $\delta^e_{ij}(n)$ and $\delta^v_i(n)$ represent the decision to use link $(i,j) \in E$ and node $i \in V$ in the routing at iteration n, c_{ij} is the capacity of edge (i,j), $f^h_{ij}(n)$ is the fraction of flow h that will be routed through link (i,j), η_{max} is the maximum degree of the network, and $b^h_i(n)$ is the flow h generated at node i which is positive if i is the source of the flow ($b^h_i(n) = d_h$) and negative if i is the destination of the flow ($b^h_i(n) = -d_h$); k^e_{ij} and k^v_i are the repair cost of edge (i,j) and vertex i. The recovery cost is heterogeneous and depends on the location of the nodes/edges.

Constraint 1a specifies that the fraction of flow that will be routed through link (i,j) has to be smaller than or equal to the capacity of that edge; constraint 1b specifies that the degree of each node is smaller than or equal to the maximum degree of the network; 1c shows the flow balance constraint, i.e. the total flow out of a node is equal to the summation of total flow that comes into a node and the net flow generated/consumed at the node; 1d states that we consider non-negative assignment of flows and finally 1e shows the binary variables representing the decisions to use nodes and edges at iteration n. Our goal here is to minimize the expected recovery cost. Since our initial estimate about the failure in the system is not always correct, it may happen that we try to repair a gray node/edge which is not

failed, but simply isolated from the working components. This is unavoidable in some cases. Nevertheless, it is an unwanted event. For this reason, we distinguish between necessary and unnecessary recovery interventions. We associate a cost to the intervention on an unknown but working network element, to take account of the cost to send personnel to make a local inspection of the device. In the evaluation, we consider a metric called *unnecessary repairs* which corresponds to the total number of nodes/edges in the gray area, $n_i \in V_U, e_{i,j} \in E_U$, which we decide to use, $\delta^e_{ij}, \delta^v_i = 1$, and are found to be properly functional after a local inspection. On the other hand, *necessary repairs* are the total number of nodes/edges in the gray or red area, $n_i \in \{V_U \cup V_B\}, e_{i,j} \in \{E_U \cup E_B\}$ which we decide to use, $\delta^e_{ij}, \delta^v_i = 1$ and are found to be broken.

Proof of NP-Hardness. The Steiner Forest problem which is NP-Hard and APX-Hard in general graphs [25–27], is a special case of our optimization problem. To reduce the Steiner Forest problem to an instance of our problem, we create one unit of demand flow for each demand pair in our supply graph. We assume all source/destination pairs in the supply graph are the set of node pairs in the Steiner Forest problem for which we want to find a forest with minimum cost. Furthermore, we assume there exist no broken/gray nodes in the supply graph and all edges are broken with expected repair cost of $k^e_{ij}\zeta^e_{ij}$. We also assume that the capacity of edges is large enough to accommodate the sum of all demand flows. Since there exists no broken node, this instance of MINER returns a set of edges to recover, and since the capacity of links are large enough to accommodate the sum of all flows, a single path between any source/destination pair suffices to accommodate the demand. Therefore, the union of repaired edges generates a Steiner forest because any cycle implies unnecessary repairs. Also, since MINER minimized the repair cost, the forest is the one with minimum cost. Therefore, our problem is also NP-Hard. □

In this work we do not consider cascading failures. In particular, when defining the status of the network elements, we assume that any cascading process is terminated and the failure has reached its full extent. Notice also that ISR assumes that, after recovery, routing will be performed according to a centralized scheme, based on the algorithm decisions so that no overload will be produced on healthy links producing further cascading failures. The problem of cascading failures in a communication network under uncontrolled routing is itself an interesting problem and can be further studied in future works. Notice also that such a centralized approach is required during an emergency and is made possible through the cooperation of multiple governmental and private entities. In the USA, for instance, the Federal Emergency Management Agency (FEMA) is in charge, according to the Stafford act, of providing disaster relief and emergency assistance in the territory of the USA. The Agency recognizes the communication infrastructure are critical for the community and includes it in the list of the infrastructures to be repaired to restore critical communication services during an emergency, with utmost urgency. Despite the fact that multiple private businesses may own different parts of the communication infrastructure, FEMA promotes a holistic approach to disaster recovery providing financial and physical assistance. The example of FEMA holds for the USA, but almost every country that recognizes the relevance of the communication network as a critical infrastructure adopts identical policies. For the same reason, in this paper we assume that all nodes can be exploited as monitors to gain more information and can ping their neighbors to gather more information.

C. Estimating the Probability Distribution of Network Failure

In the absence of detailed knowledge of which are the failed components of the network, we assume the availability of a probabilistic estimate of the failure scenario. To this purpose, we assume knowledge of the probability of failure of each node/link in the gray area, which can be found using machine learning algorithms [28, 29], reinforcement learning approaches [30], seismography analysis in case of an earthquake [31], or understanding the robustness of different parts of the network. This is typical of other works in this area. In [28], Bent et al. use historical sampling and machine learning techniques to learn the distribution online. Tati et al. tackle the problem of unknown failure distribution from the perspective of reinforcement learning and propose an algorithm that learns path availabilities through probing [30]. Guikema et al. use a statistical learning theory approach to analyze risk and reliability of infrastructure systems [29].

Failures of network elements in the gray area may be geographically correlated or independent [32]. For example, if a router in a building has failed, then it is more likely that other nodes/links in that building are also failed due to building collapse. We model the intensity of the disruption according to a geographic failure distribution. For example, in case of an earthquake with known epicenter coordinates and strength, we model the geographical distribution of failures according to a bi-variate Gaussian distribution centered at the epicenter of the earthquake, and with variance set in proportion to its strength. Instead, when failures are equally likely to occur in any network device of an area, we assume that they are uniformly distributed in a region wide enough to include all the known failures. To evaluate the performance of our recovery approach when our estimate of the probability distribution of failure is not perfect, we performed a sensitivity analysis in section VI.

IV. Iterative Stochastic Recovery Algorithms

In this section, we propose the *Iterative Stochastic Recovery* (ISR) algorithm, in its three variants, namely, Iterative shortest path (ISR-SRT), Iterative branch and bound (ISR-BB), and iterative multi-commodity (ISR-MULT). The skeleton of these versions follow the same structure and only differ in terms of the approximate algorithm they use. We summarize ISR algorithm in six main steps shown in Figure 4 and Algorithm 1.

Initially, ISR starts by estimating the probability distribution of the network failure (Step 1). At each iteration, ISR uses

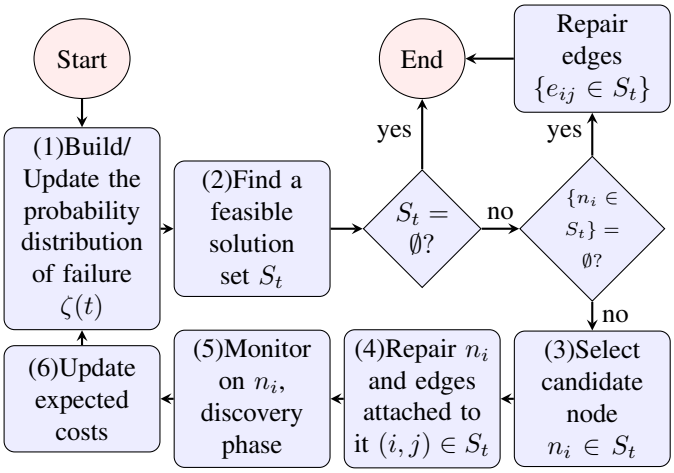


Fig. 4: Different steps of our iterative stochastic recovery (ISR).

an approximate algorithm to build a partial solution set of candidate network components to repair, $S_t = \{(i \in V_U \cup V_B \quad |\delta_i = 1), ((i,j) \in E_U \cup E_B \quad |\delta_{ij} = 1)\}$ (Step 2). In our evaluation, we do not consider infeasible problems, i.e., there exists at least one feasible solution which can satisfy all critical services.

We use three different optimization techniques explained in Section IV-A to build the partial solution set. The partial solution minimizes the MINER problem based on the current estimated costs which can change as we gain more knowledge about the gray area. In step 3, the nodes in the partial solution set S_t, are ranked based on the amount of flow in critical services that they are likely to route, and a node with the maximum value is selected as a candidate node (Steps 3 and 4). We repair the candidate node, and use it to monitor (Step 5) the surrounding network and obtain more information about the status of the network. In step 6, the algorithm updates the previous estimate of the costs after the discovery. The procedure is repeated until all the demands are satisfied or no more repairs are possible although there is a demand loss.

A. An approximate feasible solution

This section describes our three different approaches to find an approximate feasible solution, step (2) in Figure 4, of the MINER problem. We use this approximate solution set (S_t) to select a candidate node to repair and gain information in our ISR algorithm. The first alternative is to use an iterative shortest path algorithm, which has lower time complexity compared to the other approaches but may not satisfy all the demands. The second alternative, is to use an iterative branch and bound, which has high complexity due to large space exploration but gives a solution very close to the optimal in terms of repair cost; and finally, we use an iterative multi-commodity relaxation of the problem to reduce the execution time but with higher repair cost with respect to the iterative branch and bound solution.

Algorithm 1: Iterative Stochastic Recovery (ISR)

Data: The supply graph G, demand graph H, E_U, V_U, E_B, V_B, E_W, V_W, initial belief about the failure pattern $\zeta(0)$
Result: Set of nodes/edges to be recovered to satisfy the demand

```
DemandSatisfied= False;
t= 0;
Solution = ∅ ;
while DemandSatisfied !=True do
    Find an approximate solution set of nodes/edges to repair from
      the MINER problem that satisfy the demand:
      S_t = {V_s(t) ∈ (V_B ∪ V_U), E_s(t) ∈ (E_B ∪ E_U)} using
      ISR-SRT, ISR-BB, or ISR-MULT.;
    if S_t == ∅ then
        DemandSatisfied = True;
        break;
    else
        SelectedNode = Select a node with highest flow in the
          current solution S_t ;
        if |SelectedNode| > 1 then
            SelectedNode = Select the node with maximum failure
              probability ;
        Repair the SelectedNode, n_i and edges attached to it,
          e_{n_i j} ∈ S_t ;
        Solution = Solution ∪ n_i ∪ e_{n_i j ∈ S_t} ;
        Put a monitor on the selected node and run K-hop
          discovery phase;
        t = t + 1 ;
        Update our belief ζ(t) from failure probability distribution
          from the discovered nodes/edges ;
return Solution
```

1) Iterative Shortest Path (ISR-SRT): This intuitive heuristic first sorts all the demand pairs in decreasing order of demand flows, and repairs all the shortest paths that are necessary to satisfy each demand separately, without considering potential conflict among them. To account for the impact of uncertainty, we use a new notion of path length. For a path at the n^{th} iteration, the length of each link $e_{ij} \in E$ is defined as $l^{(n)}(e_{ij}) = k^e_{ij} \cdot \zeta^e_{ij}(n) + (k^v_i \cdot \zeta^v_i(n) + k^v_j \cdot \zeta^v_j(n))/2$, where $k^e_{ij} \cdot \zeta^e_{ij}(n)$, $k^v_i \cdot \zeta^v_i(n)$ and $k^v_j \cdot \zeta^v_j(n)$ are the expected cost of repair for edge e_{ij} and nodes i and j based on the estimated probability distribution at the n^{th} iteration. Therefore, the algorithm finds the shortest expected repair cost paths for each demand pair to repair independently. We run the full optimization based on the current estimated costs each time, repair one node and put a monitor on the repaired node, and then run the optimization with the updated cost again. Since the algorithm does not consider potential conflicts among demand pairs, it is possible that only a portion of demand pairs will be satisfied in the network. The advantage of this algorithm is its polynomial time complexity since it only needs to find the shortest cost paths of all demand pairs which makes it a good candidate for situations, where a small amount of critical demands needs to be satisfied in short period of time.

2) An Approximate Iterative Branch and Bound (ISR-BB): As a second option to determine a more accurate estimate solution of the problem, we use an iterative branch and bound optimization [33]. The algorithm starts by finding a solution of the problem by removing the integrality restrictions. The resulting linear programming relaxation of MINER gives a solution for the *Multi-Commodity Flow* relaxation of the problem [34]. The multi-commodity relaxation has a

polynomial time complexity and gives a lower bound (LB) for the minimization. If the solution satisfies all integrality restrictions, then we have the optimal solution, otherwise, we pick a fractional variable, δ_x, and make two branches by creating two more constraints in the optimization ($\delta_x = 0$ and $\delta_x = 1$). We continue this procedure by making more branches to get closer to optimal. The smallest branch that satisfies all integrality constraints is called an *incumbent*. We stop branching once the gap between the incumbent's objective function and the lower bound in the first iteration on the objective function is smaller than a threshold (Gap), or we can stop branching after passing a given time limit. In the first case, the algorithm gives a solution with an objective function within $(100 * Gap)/LB$ percentage of the optimal. In the second case, there is no guarantee on the bound but we have a guarantee on the execution time of the algorithm. In the worst-case scenario, we need to put all fractional variables from the LP-relaxation of MINER in the solution set. At each iteration, we run the optimization with the current estimation of the costs, repair one node and run the discovery phase, and then run the optimization with the updated costs again.

The advantage of this algorithm is its low recovery cost. Although the execution time is high due to the exploration of all possible branches, we can trade-off recovery cost to reduce the execution time.

3) An iterative multicommodity (ISR-MULT): Since the approximate branch and bound algorithm has high execution time due to large space exploration of branches, we propose a new iterative multicommodity solution. In this algorithm, we do not explore all possible branches, but only select the branch which is more likely to stay in the final solution (best candidate node selection). We first start by constructing a linear programming (LP) relaxation of the *MINER* problem which can be solved in polynomial time providing non-integer solution for $0 \leq \delta_i \leq 1$ and $0 \leq \delta_{i,j} \leq 1$. The LP relaxation gives a lower bound on the objective function of MINER, but it can result in many repairs if we repair all fractional variables. To reduce the number of repairs, we select the best candidate node from the current non-integer solution to repair and run the discovery phase and update the cost functions and failure probability distribution accordingly. We iterate the algorithm until all the demand pairs are satisfied in the network. Therefore, the iterative multicommodity solution works the same as a branch and bound technique except that, at each iteration of the algorithm we only select one of the branches and do not explore other possible branches. At each iteration of the algorithm, we repair the node or the edge which contributes the maximum flow. In case of ties, we choose the network element with maximum failure probability.

B. Complexity analysis

In this section, we compare the complexity of the three proposed approaches.

We note that all versions of ISR repair one node at a time and therefore in the worst case scenario, the optimization problem needs $O(|V|)$ iterations. In ISR-SRT, for each demand pair ($O|E_H|$) we calculate the shortest paths between the endpoints in $O(|E|) \times O(|E| \times log|V|)$, using the Dijkstra algorithm repeatedly on residual graphs until the set of shortest paths obtained is sufficient to meet the demand requirements. Notice that at each iteration, the residual graph will have at least one edge less than the original graph. This iteration, for calculating a number of shortest paths to satisfy the demand can therefore be repeated up to $O(|E|)$ times. Hence the ISR-SRT has a complexity of $O(E_H) \times O(|E|^2 \times log|V|)$. Since it is executed in the algorithm of Figure 4 for up to $O(|V|)$ times, the overall complexity of ISR-SRT is $O(|V||E_H||E|^2 \times log|V|)$. Assuming that $|E| = O(|V|)$, ISR-SRT has complexity $O(|E_H||E|^3 \times log|E|)$.

The bottleneck of the ISR-MULT algorithm is solving the LP relaxation. The relaxed problem has $O(|E|+|V|+|E||E_H|) = O(|V|+|E||E_H|)$ decision variables, and $O(|E|+|V|+|V||E_H|) = O(|E|+|V||E_H|)$ constraints. The complexity of solving LP relaxation using Karmarkar's interior point method [35] is $O(n^{3.5} \times L^2)$, where n is the number of variables and L is the number of bits in the input. Therefore, under the assumption that $|E| = O(|V|)$, the LP relaxation takes $O(|E|^{7.5} \times |E_H|^{7.5})$ and ISR-MULT takes $O(|E|^{8.5} \times |E_H|^{7.5})$ to run. Assuming ISR-BB runs B branches to stop (depending on the time budget), the complexity of the algorithm is $O(|B| \times |E|^{8.5} \times |E_H|^{7}.5)$.

C. Best candidate node selection

The choice of the best candidate node, step (3) in Figure 4, is performed based on a centrality ranking, where we use a new notion of centrality which generalizes the classic concept of betweenness centrality, to consider flow routing. Assuming the total set of paths in the current solution (S_t), is P^* and $P^*_{n_i}$ be the total set of paths in the current solution (S_t) that contain n_i, then the candidate node, N^*_i, is chosen as follows:

$$N^*_i = argmax_{n_i \in S_t} \frac{\sum_{p \in P^*_{n_i}} f(p)}{\sum_{p \in P^*} f(p)} \tag{2}$$

The numerator is the total amount of flow which can be satisfied in the current solution set (S_t) and passes through n_i and the denominator is the total amount of satisfied flow in the current solution. We also tried a different criterion for selecting the best candidate node. In particular, we tried to select the node with maximum failure probability. Extensive simulation analysis shows that in most scenarios, selecting the node with maximum centrality gives better results and reduces the cost of repair. Therefore, we choose centrality as the main metric and whenever this metric is the same for several nodes, we choose the node with maximum failure probability, $argmax_{n_i \in S_t} \zeta^v_{n_i}(t)$, to reduce unnecessary repairs, where $\zeta^v_{n_i}(t)$ represents the estimation of failure probability of node n_i, at time t.

D. Monitoring nodes

This section describes how monitor nodes probe the surrounding network to derive more information on the status

of the reachable nodes and links. We assume that at the beginning of the algorithm, a monitor is deployed on each demand endpoint. Each monitor is able to identify other nodes that are located within a distance of K-hops, for example by using traceroutes or other probing methods.

Monitors adopt a breadth-first search algorithm to explore the network, and truncate the visit at K-hops. Whenever a monitor determines that a node v is not able to forward the probe to one of its neighbors w, the monitor marks both the link (v, w) and the node w as gray as the monitor is not able to assess whether the failure is located in the node w or in the link (v, w). Note that a monitor node can only detect its adjacent link failures.

E. When to iterate the optimization

In order to reduce the complexity of the algorithm, when the solution of the current iteration of the approximate does not change after the discovery phase, we propose the *Heuristic trigger for solution update*. Assuming S^* is the total set of nodes/edges in the gray area and $S(t)$ is the total set of gray nodes/edges which have to be repaired to satisfy the demands in the current iteration of the algorithm, it is possible that after running the discovery phase of our algorithm the next solution set $S(t+1)$ remains the same and therefore we do not need to iterate the optimization.

Heuristic trigger for solution update. *Before running the discovery phase, if the cost function for the current solution $S(t)$ was X, and it changes to X' after K-hops discovery, and the cost function of the set outside the current solution $S^* - S(t)$ was Y and changes to Y' after K-hops discovery, then we only need to re-run the optimization if $X - X' < Y - Y'$ because there exists a possibility that there is a better solution other than the current solution.*

V. Comparison with prior works

In this section, we introduce two prior works, ISP [5] and CeDAR [6] that aim at recovering the network progressively after large-scale disruption. Since the state-of-the-art ISP algorithm assumes perfect knowledge of the failed components, we modify it to work under uncertainty and call it progressive ISP.

A. Progressive ISP

This section describes progressive ISP which is our extension of the state-of-the-art iterative split and prune (ISP) [5]. The basic ISP algorithm starts iteratively by ranking the nodes based on a new centrality metric, called *demand based centrality*, and reducing the demands by either pruning or splitting the demand on the best candidate node. The demand pair which is least likely to be routed elsewhere is split over the repaired node to break the problem into two smaller sub-problems. The demand can be pruned once we find a working path that can satisfy a portion of the demand.

While it has been shown that ISP, in terms of recovery cost, performs very close to optimal compared to other greedy approaches when full knowledge of the failure is known, it performs poorly under uncertain failure distributions (see Table I). Therefore, we adapted the algorithm to accommodate uncertain failures in a gray area, and iterate at each step to discover the status of gray nodes/edges by putting monitoring nodes on the repaired nodes. We use an uncertain estimation of failure distribution in the first iteration of the algorithm and change the length of the edge $e_{ij} \in E$ at the n^{th} iteration to $l^{(n)}(e_{ij})/c_{ij}$ where $l^{(n)}(e_{ij})$ is the expected cost of e_{ij} based on the estimated probability distribution at the n^{th} iteration defined in Section IV-A1 (ISR-SRT), and c_{ij} is capacity of e_{ij}. The edge cost is divided by c_{ij} to give higher cost to the paths which have smaller capacity. Further, we put monitoring software on the node which is chosen to split the demand at each iteration to discover the gray area. However, once the demand splits over a candidate node, a routing decision is made on the selected node. Therefore, as we will see in Section VI, even with the help of monitoring nodes, progressive ISP does not perform well in terms of the total number of repairs under uncertain failures. In the remainder of the paper, we use the terms "progressive ISP" and "ISP" interchangeably.

B. CEDAR

Ciavarella et al. [6] propose a polynomial-time heuristic called *Centrality based Damage Assessment and Recovery* (CeDAR) that progressively recovers the network under the incomplete knowledge of failure. CeDAR aims at maximizing the total satisfied flow during the recovery process. At each iteration step of CeDAR, a limited amount of budget is available to repair nodes/edges and the repairs are scheduled according to the availability of the resources at each time step. While our progressive recovery approach and CeDAR differ in the objective function, they both propose a progressive recovery under the incomplete knowledge of failures and therefore, we compare our result with CeDAR in Section VI in terms of number of repaired nodes/edges and monitors placed.

At each iteration stage, CeDAR makes a repair decision based on the available resources and simplifies the problem by reducing the demand according to the pruning operation. Also, CeDAR updates the current status of the network by putting a monitor on the repaired node and progressively updates the incomplete knowledge of the disrupted area. However, CeDAR does not make any assumption of the probabilistic failure distribution of the disrupted area.

VI. Evaluation

In this section, we compare ISR algorithms, presented in IV, to the modified version of ISP introduced in Sections V-A. We use different network topologies including planar and non-planar real topologies taken from the Internet Topology Zoo [23, 24]. In particular considering the topologies BellCanada, Deltacom, and KDL. In addition to these ISP topologies, we

TABLE III: Network characteristics used in our evaluation.

Network Name	# of nodes	# of edges	Average Node degree
BellCanada	48	64	2.62
Deltacom	113	161	2.85
KDL	754	895	2.37
Minnesota	681	921	2.7

considered a physical layer topology, using the map of the fiber network of Minnesota made available by Aurora Fiber Optic Network [36], which results from the collaboration of more than 50 carriers in the state.

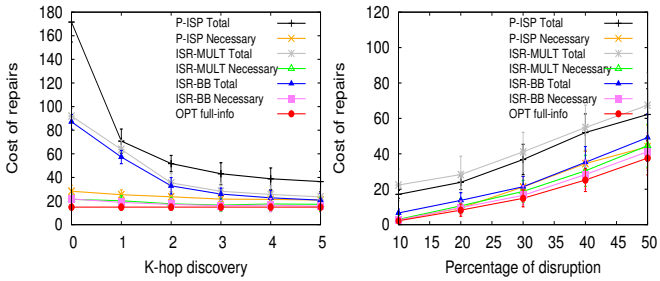
Table III shows the characteristics of the topologies used for the evaluation. In addition to the real network topologies, we use synthetic Erdos-Renyi graphs with 100 nodes, where we varied the probability of having an edge between any two different nodes, to investigate the behavior of the algorithms in scenarios of increasing complexity.

In the following experiments, we consider several scenarios, in which we vary different aspects, such as the number of demand pairs, the amount of flow demand for each pair, and the parameters defining the geographical extent of the disrupted area. For each scenario we randomize the results running 20 different trials, in which, depending on the scenario, we vary the random selection of source/destination pairs and the random disruption of network elements. We implement our recovery algorithms in python and used the *Gurobi* optimization toolkit, on a 24-core, 2.6 GHz, 32G RAM cluster [37].

A. K-hop discovery impact

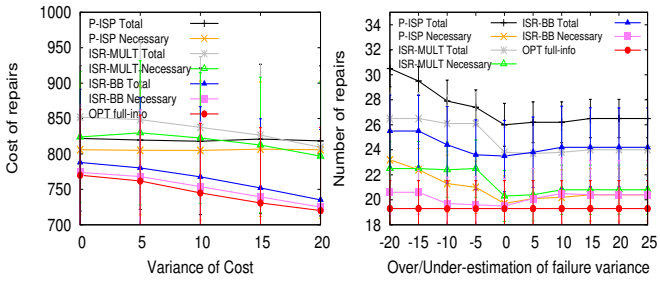
In this section, we investigate the impact of the depth of the discovery phase on the performance of the proposed algorithm. We change the number of discovered hops for the monitoring nodes from 1 to 5. We use the Minnesota fiber network topology with 6 demand pairs and 5 units of flow per demand. The link capacity is set randomly in the interval $[20, 30]$. We use a unitary repair cost for each node and edge. From Figure 5a, we can see that increasing the number of discovered hops improves the restoring performance of our algorithms in terms of total repair cost. We performed similar experiments with different topologies, which we do not show in this paper, due to space limitations, and obtained similar results. As Figure 5a highlights, the number of monitoring hops affects the number of unnecessary repairs (which we recall are the interventions performed on actually working, though, unreachable nodes) significantly, while it has a moderate effect on the number of necessary repairs. In fact, an inspection performed on a broken element with unknown status, always results in a necessary intervention. By contrast, additional knowledge resulting from a higher setting of the parameter K, helps in reducing the number of unnecessary interventions.

From this example it is clear that ISP works worse than ISR. In fact, ISP is not designed to work with uncertain knowledge of failure locations. Indeed ISP associates repair interventions with routing decisions, and never reverts a decision made at a past stage using information made available at later stages. By contrast, all variants of SRT are able to adjust routing decisions based on the most updated available knowledge, which results in lower recovery costs with respect to ISP.



(a) Increasing K-hop. (b) Increasing disruption.

Fig. 5: K-hop discovery and disruption variance (K-hop=2), (Minnesota fiber network topology).



(a) Heterogeneous cost. (b) Sensitivity analysis.

Fig. 6: a) The impact of heterogeneous repair cost variation on total cost of repair, b) Over/under-estimation of the disruption by adding an error between -20 to 25 to the variance (BellCanada, K-hop=2).

B. Percentage of Disruption

In this scenario, addressed in Figure 5b, we change the amount of disruption in the network to evaluate the performance of the algorithms. We use the Minnesota fiber network topology with 6 demand pairs and 5 units of flow per demand pair. The link capacity is set randomly in the interval $[20, 30]$. We used a Gaussian failure distribution and changed the percentage of disruption from 10 to 50. Figure 5b shows the simulation results for this scenario.

We observe that the difference from the optimal is higher for small amount of disruption, and all the algorithms perform close to each other when the percentage of disruption is higher. This is due to the fact that, as we increase the percentage of disruption, the total number of repairs increases until the whole network get disrupted. Therefore, the uncertainty in the gray area has less impact on the restoration performance of the algorithms because the whole gray area is failed. Furthermore, the number of necessary and unnecessary repairs is the same for dense disruptions since most of the nodes in the network are failed and the discovery phase does not help to reduce the number of unnecessary repairs by a large amount.

C. Heterogeneous repair cost

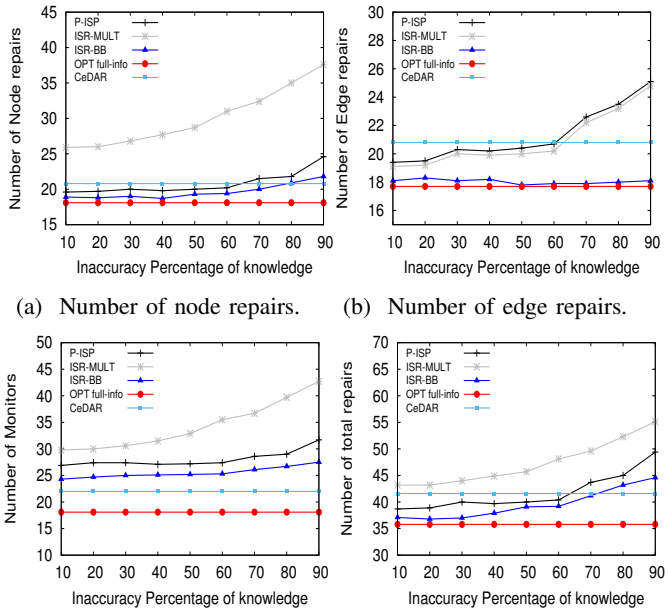
In this scenario, we analyze the impact of heterogeneous repair cost. We considered BellCanada topology with 5 demand pairs and 5 units of flow per demand pair. The link capacity is set randomly in the interval [20, 50]. We considered a scenario where the whole network is disrupted and used heterogeneous repair cost with the average of 20 derived from a uniform distribution, and changed the variance of cost from 0 to 20. Figure 6a shows the total and necessary repair cost for this scenario. As shown, our recovery algorithms perform better in terms of the total cost of repairs compared to the state-of-the-art ISP algorithm when the variance of heterogeneity is higher. This is due to the fact that when the variance of heterogeneity is higher, the algorithm has a higher solution space to choose the repair schedule and therefore, the algorithm performs better compared to a homogeneous repair cost for all nodes/edges. Therefore, in the next set of experiments, we consider a homogeneous repair cost.

D. Sensitivity analysis

In our next set of experiments, we study the sensitivity of the proposed algorithm with respect to the correctness of the initial failure estimation. We use the BellCanada topology where the link capacity is set randomly in the interval $[20, 50]$. The network disruption is randomly generated according to a Gaussian geographic distribution with a variance of 50 that destroys 50% of network components on average. We consider a varying error in the estimate of the disruption extent, and we overestimate/underestimate the disruption by adding an error between -20 to 25 to the variance of the disruption. Figure 6b shows the simulation results for this scenario, where an error of 0 means that the estimate is generated according to the same distribution that is used to generate the failures. We observe that when we underestimate the disruption, the algorithms try to route the critical demands through a part of the network, which is more likely to be failed. Overestimating the disruption assumes that more nodes/edges have been failed than the real disruption. Thus the algorithm attempts to repair a node/edge which was not really destroyed, therefore, there is a higher number of unnecessary repairs. Furthermore, the number of repairs does not change beyond a specific overestimation, because with higher disruption, we are assuming that the whole network is disrupted and the Gaussian distribution does not give much information about the disruption. ISR-BB performs better than other algorithms in overestimation or perfect estimation scenarios, but its restoring performance decreases for underestimation scenarios. ISR-MULT is more robust in underestimation scenarios and in perfect/overestimation scenarios its performance is close to ISP.

E. Impact of the accuracy of estimate in the initial probability distribution

In this set of experiments, we use the DeltaCom topology with 6 demand pairs and 5 units of flow per demand pair,



(a) Number of node repairs. (b) Number of edge repairs.

(c) Number of monitors. (d) Number of total repairs.

Fig. 7: Impact of imprecision in the accuracy of the probability of failed nodes and edges, 50% disruption.

where the link capacity is set randomly in the interval $[20, 50]$. The network disruption is randomly generated according to a geographic failure, where 68% of network elements reside inside a circle within which the network elements fail with a probability of 0.9 and the rest of the network elements fail with a probability of 0.1. We vary the accuracy of the estimate of the failure probability from 10% to 90%. For example, if the accuracy measure on the x-axis is 10, then the assumption on the failure probability is off by 10%, i.e. we estimate a failure of 0.8 or 0.2 for the network elements within or outside the circle, respectively.

Figure 7 shows the number of nodes repaired (Fig 7a); number of edges repaired (Fig 7b); and monitors placed (Fig 7c) as we decrease the accuracy of our assumed probability distribution of failures. It is shown that CeDAR places fewer monitors and has higher total number of repairs compared to the different versions of our ISR algorithms. Also, since CeDAR does not consider failure probabilities, its performance does not change by changing the inaccuracy percentage of knowledge. We also observe that all versions of our ISR algorithms perform worse than CeDAR when the inaccuracy percentage of knowledge is higher than 70%. Both ISR-MULT and P-ISP have smaller number of edge repairs when the inaccuracy percentage of knowledge is small and repair more edge repairs in higher inaccuracy percentage of knowledge. Also, even though ISR-BB repairs fewer edges, the total number of repairs (including nodes and edges) increases as we increase the inaccuracy percentage of knowledge.

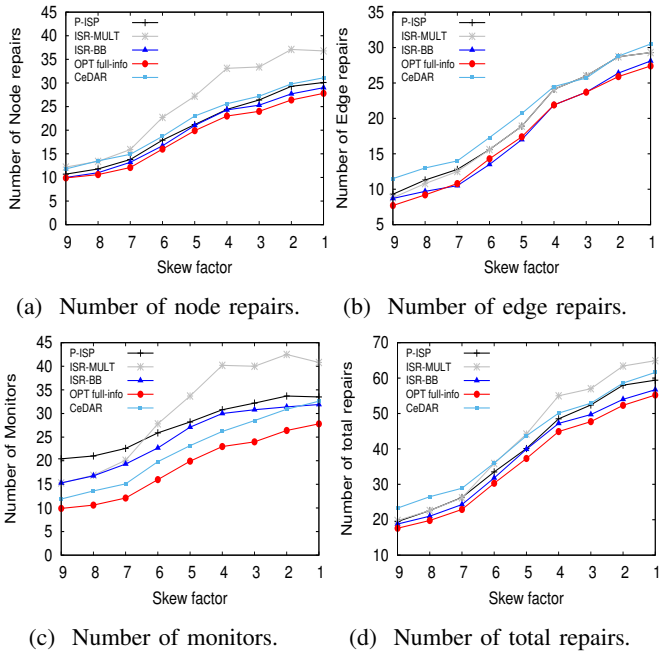
F. Skew factor

In this section, we use the DeltaCom topology with 6 demand pairs and 5 units of flow per demand pair, where

the link capacity is set randomly in the interval $[20, 50]$. The network disruption is randomly generated according to a geographic failure, where 50% of network elements reside inside a circle within which the network elements fail with a probability of 0.1 and the rest of the network elements fail with a probability of 0.9. This allows us to evaluate the impact of the certainty of the failure on our algorithm. We expect that the higher the certainty we have of the status of a gray network element, the better the results.

We then increase the probability of failure inside the circle from 0.1 to 0.9. The skew factor is 9 in the beginning and decreases to 1 as we increase the failure probability of the circle. Figure 8 shows the number of nodes repaired (Fig 8a); number of edges repaired (Fig 8b); and monitors placed (Fig 8c) as we increase the imprecision in our initial the probability distribution of failures.

We note that, although the algorithms might repair slightly higher/lower number of nodes/edges, but the number of total repairs (nodes and edges), compared to the optimal, is higher for smaller skew factors. We observe that, CeDAR places fewer monitors again in this scenario and has a higher total number of repairs when the skew factor is high, but when the skew factor is smaller its performance is close to our ISR approaches. We underline that since CeDAR's objective function is different from ours, we only compare CeDAR with our algorithms in the scenarios where we want to evaluate the inaccuracy in the evaluation of the disruption percentage and skew factor in Figures 7 and 8.



(a) Number of node repairs. (b) Number of edge repairs.

(c) Number of monitors. (d) Number of total repairs.

Fig. 8: Impact of skew factor in the accuracy of the probability of failed nodes and edges, 50% disruption.

G. Trade-off on demand loss, time complexity, and number of repairs

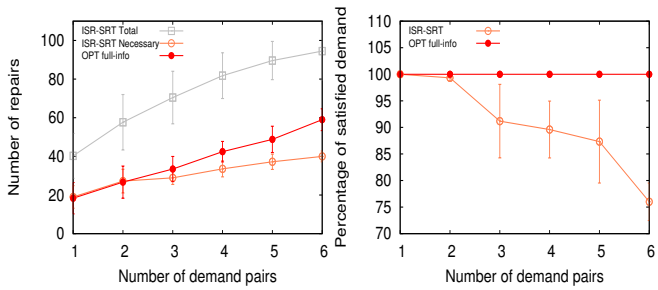
The recovery problem can be addressed by giving different priority to performance aspects such as: 1) demand loss, 2) execution time and 3) number of repairs (cost). These aspects are in conflict with each other; therefore, we study the trade-off between them.

In this scenario, addressed in Figure 9, we considered the Deltacom topology, where we set the link capacity randomly in the interval $[20, 30]$. We compare ISR-SRT to OPT to determine the amount of demand flow loss in ISR-SRT. We vary the number of critical demand flows from 1 to 6. Each demand pair has a requirement of 22 units of flow. The network disruption is randomly generated according to a Gaussian geographic distribution that causes the disruption of 43% of the network components on average.

Figure 9a shows that ISR-SRT performs a smaller number of necessary repairs than OPT but a much higher number of total repairs, meaning that ISR-SRT schedules repairs for nodes that are found to be working. Figure 9b also shows that ISR-SRT does not meet the demand requirements. The percentage of satisfied demands drops to 75% when the number of demand pairs grows to 6.

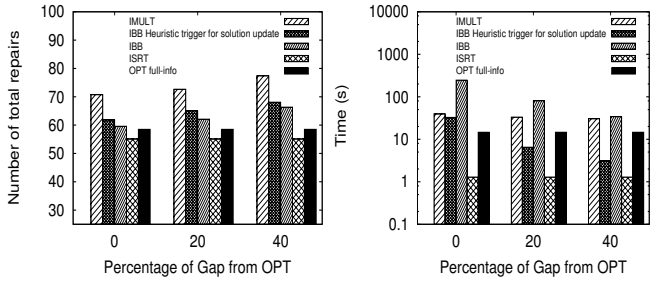
The reason for demand loss is due to the fact that ISR-SRT does not consider potential conflicts among different demands, and the decision on the nodes/links to be repaired is made separately for every demand pair without considering other demands of the network. This has two effects. First, it may lead to the wrong decisions, and therefore increases the number of unnecessary repairs. Second, the algorithm might make a routing decision in one iteration for a specific demand pair which turns to be in conflict with another demand pair in the next iteration and make it impossible for the second demand pair to be satisfied. Therefore, the repairs that are required to route the second demand pair are not performed due to the conflict, and the demand is not satisfied. This implies that the number of necessary repairs would be less w.r.t the optimal solution. Since ISR-SRT has demand loss, we do not consider the performance of ISR-SRT approach in other experiments. We underline that the other algorithms, namely OPT, ISR-BB and ISR-MULT, repair nodes/edges until all demand pairs are satisfied. In these algorithms, no routing decision is made before finding a feasible solution for all demand pairs. For this reason, they never show a demand loss. Since our goal is to restore all critical services, we do not further evaluate ISR-SRT. However, due to its low computational complexity, the algorithm can be used in scenarios where the demand load is low and a short computation time is required.

In the next experiment we used the same topology, under a larger disruption, corresponding to 75% of network elements on average. We consider 5 demand pairs, of 17 flow units each. In order to evidence the tradeoff between the number of repairs and computation time, in Figure 10 we vary the gap between the lower bound of the objective function and the solution of iterative ISR-BB and ISR-MULT algorithms from 0 to 40%. We recall that by increasing this gap, we decrease the number of iterations of the optimization algorithms, and therefore we obtain an approximation of the solution that is farther from the optimal. Nevertheless, the increase in the gap has the



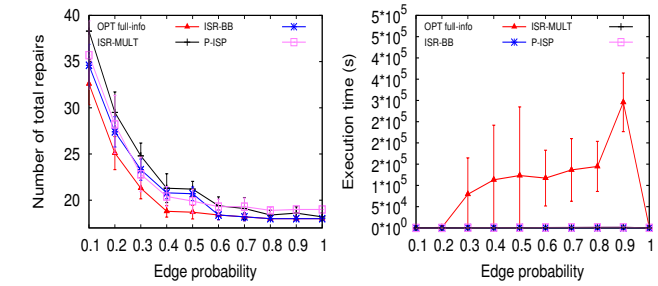
(a) Repairs. (b) Demand loss.

Fig. 9: Trade-off between number of repairs and demand loss.



(a) Repairs. (b) Time.

Fig. 10: Trade-off between execution time and repairs.



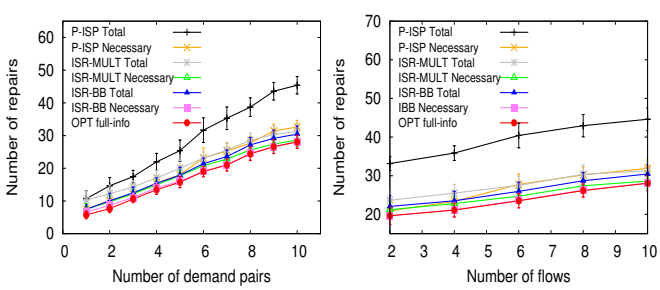
(a) Repairs. (b) Time.

Fig. 11: Synthetic Erdos-Renyi topology with 100 nodes.

advantage of reducing the computation time remarkably.

Figure 10a shows that, when we increase the gap from 0 to 40%, the difference between the total number of repairs of ISR-MULT with respect to optimal increases by a factor of 1.6, while this factor is 3.4 for ISR-BB. Furthermore, we observe that by running the Heuristic trigger for solution update, introduced in Section IV-E, the total execution time on average decreases by a factor of 10.5, while the total number of repairs increases of only 3.8%. This is mainly due to the fact that most of the time, after running the first optimization step, the solution is still valid by using Heuristic IV-E. We did not include the execution time results for progressive ISP since its performance has not been optimized to run on multi-core machines.

In the next scenario, we used synthetic Erdos-Renyi non-planar graphs. In an Erdos-Renyi graph, any two nodes are connected through an edge with probability p. We considered an Erdos-Renyi topology with 100 nodes where each link



(a) Increasing demand. (b) Increasing flows.

Fig. 12: Increasing demand pairs and flows (BellCanada).

has a capacity of 1,000 units of flow. We set the number of critical demand pairs to 6, of one unit each. Notice that with this set of demand flows and capacities, the problem reduces to establishing connectivity between the endpoints of the demand pairs. The complexity of the problem increases as we increase the parameter p and the graph becomes gradually non-planar. We compare the behavior of ISR-MULT, ISR-BB and progressive ISP with the optimal solution that would be obtained in the ideal setting of complete knowledge. In ISR-MULT and ISR-BB, we set the gap between the lower bound of the objective function and the solution of iterative ISR-BB and ISR-MULT algorithms to 50%. Once the gap is satisfied, we put all fractional variables in the solution and select a node to repair and continue this procedure until all critical services are restored. Figure 11b shows the execution time of the approximate solutions with respect to optimal as we increase the value of p. We observe that, since MINER is NP-Hard, the optimal recovery with full information has a very high execution time, while if we stop the algorithm when the objective is within 50% of the lower bound, the number of repairs is still close to optimal in ISR-MULT and ISR-BB, and the execution time with respect to OPT reduces by a factor of 200 in ISR-BB and 630 in ISR-MULT.

H. Increasing number of demand pairs and amount of flow

In this section, we investigate the impact of the number of demand pairs and of the amount of demand flow of each pair, on the number of necessary repairs. We consider the BellCanada topology, where we set random link capacity with values in the interval $[20, 50]$. We increased the number of demand pairs from 1 to 10, where each demand has a requirement of 10 units of flow. Figure 12a shows the simulation results for this scenario. We used a Gaussian disruption with disruption variance of 20, which destroys around 40% of the network. As we increase the number of demand pairs, the gap between necessary and unnecessary repairs increases in progressive ISP, while the number of necessary repairs is still close to optimal. This is mainly due to the fact that ISP was not designed for uncertain failures. ISR-IBB has the smallest number of repairs and ISR-MULT's number of repairs is between ISP and ISR-IBB.

In the next scenario, we consider the same network topology and same disruption parameters. We set the number of critical

TABLE IV: Potential Implication of the proposed algorithms.

Algorithm	Cons	Pros
ISR-SRT	Demand loss, cannot satisfy all demands	Low complexity, easy to implement. Can be used to satisfy small critical demands in short time.
ISP	High number of unnecessary repairs in high demand load	Low time complexity compared to ISR-BB and ISR-MULT, works better than ISR-MULT in low demand load
ISR-BB	High time complexity due to large space exploration	Low number of repairs, best for small topologies. Can be configured to reduce the execution time with higher number of repairs
ISR-MULT	Moderate time complexity, high number of repairs in smaller traffics (can be combined with ISP to have advantage of both)	Smaller number of repairs compared to ISP, higher than ISR-BB. Better restoring performance for large number of demand flow/pair.

demand pairs to 5 and increased the units of flow per demand pair from 2 to 10. Figure 12b shows the simulation results in this scenario for our iterative algorithms and optimal recovery with full knowledge. We observe that for less than 4 units of flow, ISP performs slightly better than the ISR-MULT solution in terms of the number of necessary repairs. However, as we increase the amount of flow per pair, ISR-MULT and ISR-BB perform better mainly because ISR-MULT and ISR-BB can refine their incorrect decisions due to lack of knowledge from the beginning of the algorithm while ISP is not able to adjust its solution after initial wrong decisions. For small number of flows/demand pair, both ISP and ISR-MULT are close to optimal. We observe that in larger topologies, ISP performs better than ISR-MULT when the total demand load (sum of all the demand flow requirements for all the demand pairs) is lower than 40% of the network capacity. This opens up the opportunity to have a hybrid scenario for low flow/pair and high flow/pair scenarios where one can get advantage of progressive ISP under low demand load and the ISR-MULT for higher demand load.

Table IV shows the comparison between progressive ISP, ISR-MULT, ISR-BB and ISR-SRT. We observed that each of the proposed algorithms has pros and cons, which makes them suitable for scenarios where we need short execution time, or higher restoring performance or small number of critical demand pairs.

VII. Conclusion

While there have been several works on timely network recovery algorithms, far less progress has been seen in the context of uncertain network failure patterns. This paper considers, for the first time, a progressive network recovery algorithm under uncertainty. We use a multi-stage stochastic optimization technique, called ISR to guess the best feasible solution set at each iteration using an estimated distribution of failure. ISR finds a feasible solution using three different approaches namely ISR-SRT, ISR-BB, and ISR-MULT. From the elements of this solution, we select the one with the highest centrality, at each iteration step to repair and exploit it as a monitor to discover the gray area, until all critical services are restored. We iterate the process, alternating monitoring, and repair activities until all critical services are restored. Our simulation results show that ISR reduces the total cost of repair significantly with respect to the state-of-the-art ISP algorithm. We also observed that we could configure our choice of trade-off between the demand loss, total number of repairs and execution time.

Acknowledgement

This research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes not with standing any copyright notation here on.

References

[1] D. Z. Tootaghaj, H. Khamfroush, N. Bartolini, S. Ciavarella, S. Hayes, and T. La Porta. Network recovery from massive failures under uncertain knowledge of damages. In *IFIP Networking*, 2017.

[2] A. Kwasinski, W. W. Weaver, P. L. Chapman, and P. T. Krein. Telecommunications power plant damage assessment for hurricane katrina–site survey and follow-up results. *IEEE Systems Journal*, 2009.

[3] Communications status report for areas impacted by hurricane irma. *Federal Communications Commission (FCC)*, September 13, 2017.

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

[5] N. Bartolini, S. Ciavarella, T. La Porta, and S. Silvestri. Network recovery after massive failures. In *Dependable Systems and Networks (DSN)*, 2016.

[6] S. Ciavarella, N. Bartolini, H. Khamfroush, and T. La Porta. Progressive damage assessment and network recovery after massive failures. In *Proceedings IEEE INFOCOM*, 2017.

[7] S. Tati, B. J. Ko, G. Cao, A. Swami, and T. La Porta. Adaptive algorithms for diagnosing large-scale failures in computer networks. In *Dependable Systems and Networks (DSN)*, 2012.

[8] T. Horie, G. Hasegawa, S. Kamei, and M. Murata. A new method of proactive recovery mechanism for large-scale network failures. In *AINA'09. International Conference on Advanced Information Networking and Applications.* IEEE, 2009.

[9] G. Yu and X. Qi. *Disruption management: framework, models and applications*. World Scientific, 2004.

[10] K. Al Sabeh, M. Tornatore, and F. Dikbiyik. Progressive network recovery in optical core networks. In *2015 7th International Workshop on Reliable Networks Design and Modeling (RNDM)*. IEEE, 2015.

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

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

[13] L. Ma, T. He, A. Swami, D. Towsley, and K. K. Leung. Network capability in localizing node failures via end-to-end path measurements. *IEEE/ACM Transactions on Networking (TON)*, 2017.

[14] P. Barford, N. Duffield, A. Ron, and J. Sommers. Network performance anomaly detection and localization. In *Proceedings IEEE INFOCOM*, 2009.

[15] N. Bartolini, T. He, and H. Khamfroush. Fundamental limits of failure identifiability by boolean network tomography. In *Proceedings IEEE INFOCOM*, 2017.

[16] D. Z. Tootaghaj, T. He, and T. La Porta. Parsimonious tomography: Optimizing cost-identifiability trade-off for probing-based network monitoring. *IFIP Performance*, 2017.

[17] D. Z. Tootaghaj, N. Bartolini, H. Khamfroush, and T. La Porta. Controlling cascading failures in interdependent networks under incomplete knowledge. In *SRDS*. IEEE, 2017.

[18] D. Z. Tootaghaj, N. Bartolini, H. Khamfroush, T. He, N. R. Chaudhuri, and T. La Porta. Mitigation and recovery from cascading failures in interdependent networks under uncertainty. *IEEE Transactions on Control of Network Systems*, 2018.

[19] S. Soltan, D. Mazauric, and G. Zussman. Cascading failures in power grids: analysis and algorithms. In *Proceedings of the 5th international conference on Future energy systems*. ACM, 2014.

[20] M. Parandehgheibi and E. Modiano. Robustness of interdependent networks: The case of communication networks and the power grid. In *Global Communications Conference (GLOBECOM), 2013 IEEE*.

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

[22] N. Bartolini, S. Ciavarella, T. La Porta, and S. Silvestri. On critical service recovery after massive network failures. *IEEE/ACM Transactions on Networking (TON)*, 2017.

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

[24] The internet topology zoo. `http://www.topology-zoo.org/`, accessed in May, 2015.

[25] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 1995.

[26] M. Bateni, M. Hajiaghayi, and D. Marx. Approximation schemes for steiner forest on planar graphs and graphs of bounded treewidth. *Journal of the ACM (JACM)*, 2011.

[27] M. Hauptmann and M. Karpiński. *A compendium on steiner tree problems*. Inst. für Informatik, 2013. `http://theory.cs.uni-bonn.de/`.

[28] R. Bent and P. Van Hentenryck. Online stochastic optimization without distributions. In *ICAPS*, 2005.

[29] S. D. Guikema. Natural disaster risk analysis for critical infrastructure systems: An approach based on statistical learning theory. *Reliability Engineering & System Safety*, 94(4):855–860, 2009.

[30] S. Tati, S. Silvestri, T. He, and T. La Porta. Robust network tomography in the presence of failures. In *ICDCS*. IEEE, 2014.

[31] Y. Bozorgnia and V. V. Bertero. *Earthquake engineering: from engineering seismology to performance-based engineering*. CRC press, 2004.

[32] S. Neumayer and E. Modiano. Network reliability with geographically correlated failures. In *Proceedings IEEE INFOCOM*, 2010.

[33] G. L. Nemhauser and L. A. Wolsey. Integer programming and combinatorial optimization. *Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin*, 20:8–12, 1988.

[34] T. C. Hu. Multi-commodity network flows. *Operations research*, 1963.

[35] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. ACM, 1984.

[36] Aurora fiber optic networks. `http://www.aurorafonet.com`, Last access on Aug. 2018.

[37] Gurobi optimization, inc. gurobi optimizer reference manual. `http://www.gurobi.com/`, accessed in, 2012.

Diman Zad Tootaghaj received her Ph.D. degree in the department of computer science and engineering at the Pennsylvania State University, University Park, PA. She received B.S. and M.S. degrees in Electrical Engineering from Sharif University of Technology, Iran in 2008 and 2011 and an M.S. degree in Computer Science and Engineering from the Pennsylvania State University in 2015. She is a Postdoc Researcher at Hewlett Packard Labs in Palo Alto, California. Her current research interests include computer network, recovery approaches, distributed systems, and stochastic analysis.

Novella Bartolini (SM '16) graduated with honors in 1997 and received her PhD in computer engineering in 2001 from the University of Rome, Italy. She is now associate professor at Sapienza University of Rome. She was visiting professor at Penn State University for three years from 2014 to 2017. Previously, she was visiting scholar at the University of Texas at Dallas for one year in 2000 and research assistant at the University of Rome 'Tor Vergata' in 2001-2002. She was program chair and program committee member of several international conferences. She has served on the editorial board of Elsevier Computer Networks and ACM/Springer Wireless Networks. Her research interests lie in the area of wireless networks and network management.

Hana Khamfroush Hana Khamfroush is an assistant professor in the Department of Computer Science, University of Kentucky. Before joining the University of Kentucky, she was a research associate in the Department of Computer Science and Engineering, Penn State University. She received her Ph.D. degree with honor in telecommunication engineering from the University of Porto, Portugal. She was named a rising star in EECS by MIT and CMU in 2015 and 2016. She was selected as one of the 200 young researchers for the Heidelberg Laureate Forum, Germany.

Thomas La Porta is the Director of the School of Electrical Engineering and Computer Science at Penn State University. He is an Evan Pugh Professor and the William E. Leonhard Chair Professor in the Computer Science and Engineering Department. He received his B.S.E.E. and M.S.E.E. degrees from The Cooper Union, New York, NY, and his Ph.D. degree in Electrical Engineering from Columbia University, New York, NY. He joined Penn State in 2002. He was the founding Director of the Institute of Networking and Security Research at Penn State. Prior to joining Penn State, Dr. La Porta was with Bell Laboratories for 17 years. He was the Director of the Mobile Networking Research Department in Bell Laboratories, Lucent Technologies where he led various projects in wireless and mobile networking. He is an IEEE Fellow and Bell Labs Fellow. Dr. La Porta was the founding Editor-in-Chief of the IEEE Transactions on Mobile Computing. He was the Director of Magazines for the IEEE Communications Society and was on its Board of Governors for three years. He has published numerous papers and holds 39 patents. He was an adjunct member of faculty at Columbia University for 7 years where he taught courses on mobile networking and protocol design.