Network Recovery from Massive Failures under Uncertain Knowledge of Damages

Diman Zad Tootaghaj†, Hana Khamfroush†, Novella Bartolini*
Stefano Ciavarella*, Seamus Hayes†, Thomas La Porta†
†The Pennsylvania State University (USA), *Sapienza University (Italy)
{*dxz149, hkham, sih5349, tlp*}*@cse.psu.edu*, {bartolini, ciavarella}@di.uniroma1.it

***Abstract*—This paper addresses progressive network recovery under 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. Three different algorithms are used to find a feasible set and determine which node to repair, namely, 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 have modified the state-of-the-art iterative split and prune (ISP) algorithm to incorporate the uncertain failures. Our results show that ISR-BB and ISR-MULT outperform the state-of-the-art "progressive ISP" algorithm while we can configure our choice of trade-off between the execution time, 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.**

I. Introduction

Large-scale failures in communication networks due to natural disasters or malicious attacks can severely affect critical communications and threaten lives of people in that area. In 2005, Hurricane Katrina led to outage of over 2.5 million lines in the BellSouth (now AT&T) network [1]. 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., [2, 3]. 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 assuming partial knowledge of failures until critical services are restored. To clarify the discussion, we consider different states of network components. Depending on the available knowledge, we consider the network to be partitioned in 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 repaired nodes 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.
- To compare with previous works, we modified a previously proposed algorithm called iterative split and prune (ISP) [3] to work under uncertainty. We refer to the modified variant of ISP as progressive ISP, as it allows a progressive approach with incremental discovery at each iteration. We show that since ISP is not designed to 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.

II. Background and Motivation

A. Background

Large-scale network failure detection and recovery has been studied when full knowledge of the failure pattern is available

ISBN 978-3-901882-94-4

in the system [4, 5, 6, 7, 8, 9]. To the best of our knowledge, network recovery has not been extensively studied under uncertainty. Wang 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 [2]. While both Wang et al.'s work and our work aim to design a progressive recovery approach, the objective is different. In [2], 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. [3]. In addition, both [2] and [3], assume that full knowledge of failure is available in the system while our work does not make this assumption.

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 to break the problem into smaller sub-problems using iterative split and prune [3]. 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 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 progressive 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 [3] when we do not have perfect information.

Consider a network in which a large-scale failure has 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, 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 one-shot, and once the algorithm is run, all repairs are determined.

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	OPT full-info	ISP full-info	ISP uncertain-info	Progressive ISP
BellCanada	28	34.23	79	45.39
Deltacom	36.94	43.26	112	55.5
KDL	55.2	63.2	165.65	83.55

Therefore, it does not have an opportunity to learn the state of the uncertain nodes. We performed a set of simulations on three network topologies which are described in Section VI to illustrate the gap in performance between ISP with full knowledge vs. uncertain knowledge. We first give the full information of the failure pattern to the system and solve the optimal NP-Hard recovery with full knowledge (OPT full-info) and the state-of-the-art ISP with full knowledge (ISP full-info). Next, we assume that the state of all the network components is uncertain, with an estimated failure distribution (uncertain-info), and run the state-of-the-art ISP algorithm where the cost of repair for each node/edge is proportional to its failure probability.

Table I shows the average total number of repairs for the three algorithms on the three topologies for 10 random selection of source/destination pairs and disruption of network elements. 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 performed by the optimal in some cases.

We then modified ISP, as described in Section V, to run iteratively, where in each iteration a repair is made. Information about network state is then gathered concerning the status of the 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, motivating the need to develop our iterative stochastic recovery (ISR) algorithm that also progressively repairs network elements and updates its knowledge of the state of the network, and makes repair decisions based on the known failure probabilities.

III. Problem Definition

We consider the problem of restoring critical services in a network subject to a large-scale failure, under uncertain knowledge of the failure extent. More specifically, in the absence of detailed knowledge of which are the failed components of the network, we assume availability of a probabilistic estimate of the failure scenario, given in the form of a geographical distribution of the failure probability of the network devices, namely nodes and links. We hereby call the *grey area* the portion of the network whose state is uncertain. We assume knowledge of the probability of failure of each node/link in the gray area, which can be found using machine learning

algorithms, seismography analysis in case of an earthquake [10], or stability analysis of different parts of the network. The network elements in the gray area might have geographical or independent correlation of failures [11]. We model the intensity of the disruption on a geographic coordinate system using a bi-variate Gaussian function, whose variance determines the extent of the disruption. 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.

Given an undirected graph $G = (V, E)$ and a set of demand pairs $D = \{(s_1, t_1), ..., (s_k, t_k)\}$, 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.

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 belief about the failure distribution 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 belief 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 $k^v_i(\zeta^v_i(n))$ and $k^e_{ij}(\zeta^e_{ij}(n))$ is the cost of repairing each vertex v_i and edge e_{ij} in the network, and is a function of the uncertainty of the failure status of the node/edge. Therefore, our objective function is to minimize the expected recovery cost (ERC) given the information from the monitoring nodes to satisfy the given demand. The nodes and edges in the graph 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 are known to be 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.

At the beginning of iterative recovery process, we assume that all the working demand endpoints are monitoring nodes which can discover the status of up to m 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 belief $\zeta(0)$ about the failure probability distribution.

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

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

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

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

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

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

$$\delta^v_i, \delta^e_{i,j} \in \{0, 1\} \qquad (1e)$$

where the binary variables δ^e_{ij} and δ^v_i represent the decision to repair link $(i, j) \in E$ and node $i \in V$, 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 and negative if i is the destination of the flow; $k^e_{ij}(\zeta^e_{ij}(n))$ and $k^v_i(\zeta^v_i(n))$ 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 or equal than the capacity of that edge; 1b specifies that the degree of each node is smaller or equal than 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 binary decision variable of repairing a node/edge.

Our goal here is to minimize the expected recovery cost. Since our initial belief 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 repair, $\delta^e_{ij}, \delta^v_i = 1$, but which 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 repair, $\delta^e_{ij}, \delta^v_i = 1$ and are actually broken.

NP-Hardness: The Steiner Forest problem which is NP-Hard and APX-Hard in general graphs [12, 13], is a special case of our optimization problem. The deterministic version of our problem is shown to be NP-Hard in [3]. Therefore, our problem is also NP-Hard. We therefore, consider a polynomial time heuristic (ISR-SRT) an LP-relaxation of the problem (ISR-IMULT) and an approximate solution of our problem (ISR-BB) to reduce the time complexity.

IV. Iterative Stochastic Recovery Algorithms

In this section, we propose the *Iterative Stochastic Recovery* (ISR) algorithm, and its three versions, 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 1 and Algorithm 1.

Initially, ISR starts by estimating the probability distribution

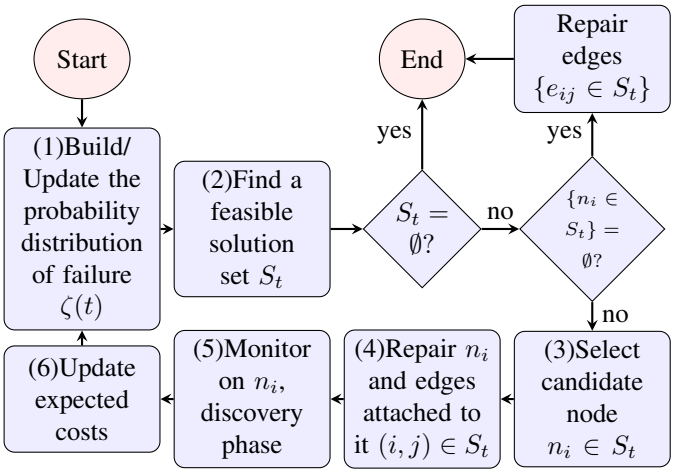


Fig. 1: Different steps of ISR.

of the network failure (Step 1). At each iteration, ISR uses 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 belief after the discovery. The procedure is repeated until demands are satisfied or there are no more repairs to satisfy the remaining demand pairs, even though the problem is feasible. The remaining demand pairs, which are not satisfied, are counted as demand loss percentage.

A. An approximate feasible solution

This section describes our three different approaches to find an approximate feasible solution, step (2) in Figure 1, 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 w.r.t the ISR-BB.

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 ∈ S_t ;
        if |SelectedNode|> 1 then
            SelectedNode = 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 m-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 $l^{(n)}(e_{ij}) = k^e_{ij}(\zeta^e_{ij}(n))\zeta^e_{ij}(n) + (k^v_i(\zeta^v_i(n))\zeta^v_i(n) + k^v_j(\zeta^v_j(n))\zeta^v_j(n))/2$, where $k^e_{ij}(\zeta^e_{ij}(n))\zeta^e_{ij}(n)$, $k^v_i(\zeta^v_i(n))\zeta^v_i(n)$ and $k^v_j(\zeta^v_j(n))\zeta^v_j(n)$ 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 [14]. 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 [15]. 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 restriction 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 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 (the node with maximum centrality). 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 has 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, we repair a node with maximum flow as described in section IV-B.

B. Best candidate node selection

The choice of the best candidate node, step (3) in Figure 1, 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:

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

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. Whenever this metric was 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.

C. 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 m-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 m 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.

D. When to iterate the optimization

In order to reduce the complexity of the algorithm, when the current iteration of the approximate optimization does not change after 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 m hops discovery, and the cost function of the set outside the current solution $S^* - S(t)$ was Y and changes to Y' after m 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. Progressive ISP (P-ISP)

This section describes progressive ISP which is our extension of the state-of-the-art iterative split and prune (ISP) [3].

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

TABLE II: 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

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 to break the flow into two. Therefore, as we will see in Section VI, even with the help of monitoring nodes, progressive ISP does not perform well in terms of total number of repairs under uncertain failures. In the remainder of the paper we use the terms "progressive ISP" and "P-ISP" interchangeably.

VI. Evaluation

In this section, we compare ISR algorithms, presented in IV, to the modified version of ISP introduced in Sections V. We use different network topologies including planar and non-planar real topologies taken from the Internet Topology Zoo [16, 17]. Table II 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 [18].

A. m-hop discovery impact

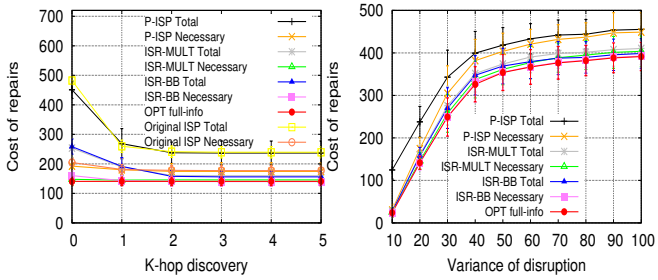
In this section, we investigate the impact of the depth of discovery phase on the performance of the proposed algorithm. We changed the number of discovered hops for the monitoring nodes from 1 to 5. We used the BellCanada topology with 10 demand pairs and 4 units of flow per demand. The link capacity is set randomly in the interval $[20, 50]$. We used heterogeneous repair cost for each node randomly from a uniform distribution in $[0, 10]$ and for each edge from a uniform distribution in $[0, 20]$. We used higher repair cost for edges due to difficulty in locating the failure and accessibility. From Figure 2a, 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. These experiments highlighted that the impact of the parameter m varies significantly with the size of network topology.

B. Disruption variation

In this scenario, addressed in Figure 2b, we changed the amount of disruption in the network to evaluate the performance of the algorithms. We used the BellCanada topology with 5 demand pairs and 4 units of flow per demand pair. The link capacity is set randomly in the interval $[20, 50]$. We used a Gaussian failure distribution and changed the disruption variance from 10 to 100. On average, 20% of the network is disrupted when the disruption variance is 10 and increases to 94% when the variance is 100. Figure 2b shows the simulation results for this scenario. We observed that, the difference from the optimal is higher for small disruption variation, and all the algorithms perform close to each other when the variance is higher. This is due to the fact that, as we increase the disruption variation, the total number of repairs increases until it gets saturated and the whole network gets disrupted. Therefore, the uncertainty in the gray area has less impact on the restoring 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 in higher disruption variations and the discovery phase does not help to reduce the number of unnecessary repairs by a large amount.

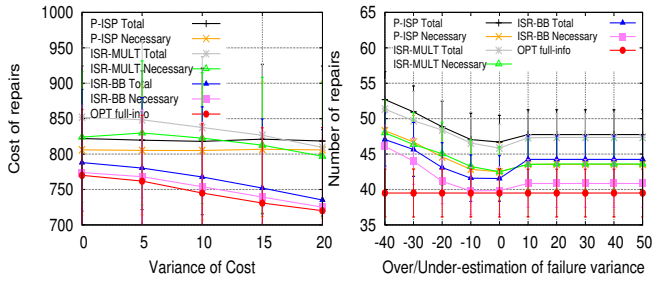
C. Heterogeneous repair cost

In this scenario, we analyze the impact of heterogeneous repair cost. We considered the 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 3a shows the total and necessary repair cost for this scenario. As shown, our recovery algorithms perform better in terms of total cost of repairs compared to the state-of-the-art P-ISP algorithm when the variance of heterogeneity is



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

Fig. 2: m-hop discovery and disruption variance, (BellCanada, m-hop=2).



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

Fig. 3: 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 -40 to 50 to the variance (BellCanada, m-hop=2).

higher. Therefore, in the next set of experiments we consider homogenous 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 variance of 50 that destroys 85% 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 -40 to 50 to the variance of the disruption. Figure 3b 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 network, which is more likely to be failed. Overestimating the disruption assumes that more nodes/edges have been failed than the real disruption and the algorithm tries 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 variance, 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 P-ISP.

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

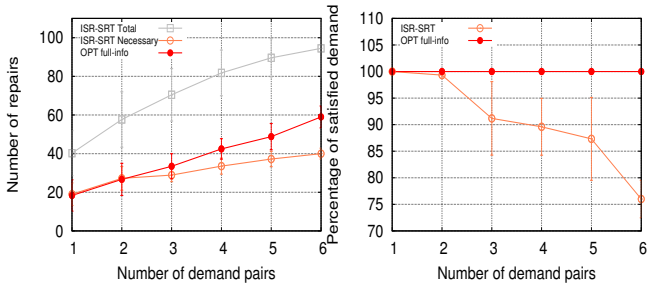
The recovery problem can be addressed by giving different priority to performance aspects such as: 1) demand loss, 2) execution time and 3) number of repairs (cost). These aspects are in conflict with each other; therefore, we study the trade-off among them.

In this scenario, addressed in Figure 4, 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 4a 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 4b 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.

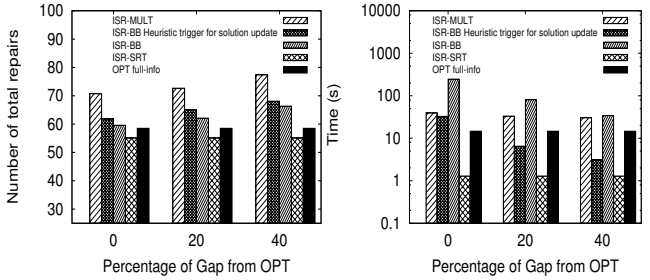
The reason for demand loss is due to the fact that ISR-SRT does not consider potential conflicts among different demands, and the decision on the nodes/links to be repaired is made separately for every demand pair without considering other demands of the network. This has two effects. First, it may lead 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. 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.



(a) repairs. (b) demand loss.

Fig. 4: Trade-off between number of repairs and demand loss.



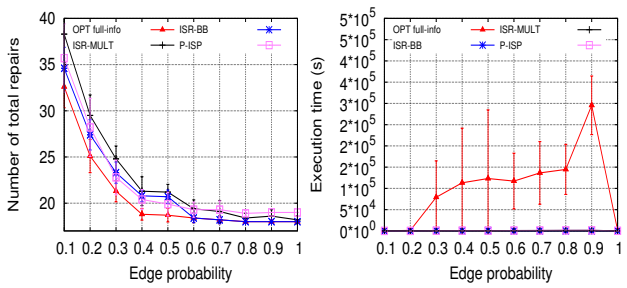
(a) repairs. (b) time.

Fig. 5: Trade-off between execution time and repairs.

In order to evidence the tradeoff between the number of repairs and computation time, in Figure 5 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 advantage of reducing the computation time remarkably.

Figure 5a 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-D, 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-D. 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 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 setting 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



(a) repairs. (b) time.

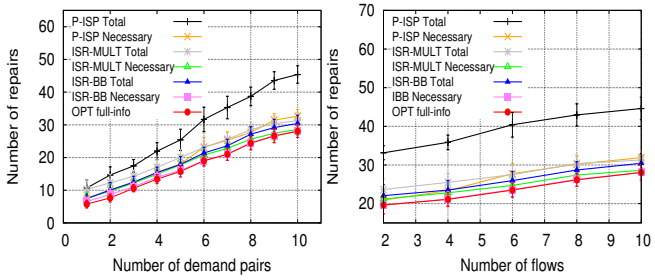
Fig. 6: Synthetic Erdos-Renyi topology with 100 nodes.

we increase the parameter p and the graph becomes gradually non planar. We compare the behavior of ISR-MULT, ISR-BB and P-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 till all critical services are restored. Figure 6b 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.

F. 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 7a 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 P-ISP, while the number of necessary repairs is still close to optimal. This is mainly due to the fact that P-ISP was not designed for uncertain failures. ISR-IBB has the smallest number of repairs and ISR-MULT's number of repairs is between P-ISP and ISR-IBB.

In the next scenario, we consider the same network topology, and same disruption parameters. We set the number of critical demand pairs to 5 and increased the units of flow per demand pair from 2 to 10. Figure 7b 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, P-ISP performs slightly better than the ISR-



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

Fig. 7: Increasing demand pairs and flows (BellCanada).

TABLE III: 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.
P-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 P-ISP to have advantage of both)	Smaller number of repairs compared to P-ISP, higher than ISR-BB. Better restoring performance for large number of demand flow/pair.

MULT solution in terms of number of necessary repairs. However, as we increase the amount of flows 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 P-ISP is not able to adjust its solution after initial wrong decisions. For small number of flows/demand pair, both P-ISP and ISR-MULT are close to optimal. We observe that in larger topologies, P-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 P-ISP under low demand load and the ISR-MULT for higher demand load.

Table III shows the comparison between P-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, 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 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.

Acknowledgments

This research is supported in part by the Defense Threat Reduction Agency under Grant HDTRA1-10-1-0085 and in part by the U.S. Army Research Laboratory under Agreement W911NF-14-0610.

References

[1] A. Kwasinski et al. Telecommunications power plant damage assessment for hurricane katrina–site survey and follow-up results. *IEEE Systems Journal*, 2009.

[2] J. Wang et al. On progressive network recovery after a major disruption. In *Proceedings IEEE INFOCOM*, 2011.

[3] N. Bartolini et al. Network recovery after massive failures. In *Dependable Systems and Networks (DSN)*, 2016.

[4] S. Tati et al. Adaptive algorithms for diagnosing large-scale failures in computer networks. In *Dependable Systems and Networks (DSN)*, 2012.

[5] T. Horie et al. A new method of proactive recovery mechanism for large-scale network failures. In *AINA'09*. IEEE, 2009.

[6] G. Yu et al. *Disruption management: framework, models and applications*. World Scientific, 2004.

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

[8] F. Farhat et al. Locally multipath adaptive routing protocol resilient to selfishness and wormholes. In *ISPEC*, 2010.

[9] D. Z. Tootaghaj et al. Game-theoretic approach to mitigate packet dropping in wireless ad-hoc networks. In *CCNC*, 2011.

[10] Y. Bozorgnia et al. *Earthquake engineering: from engineering seismology to performance-based engineering*. CRC press, 2004.

[11] S. Neumayer et al. Network reliability with geographically correlated failures. In *Proceedings IEEE INFOCOM*, 2010.

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

[13] M. Bateni et al. Approximation schemes for steiner forest on planar graphs and graphs of bounded treewidth. *Journal of the ACM (JACM)*, 2011.

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

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

[16] S. Knight et al. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 2011.

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

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